Imperfections & corrections

November, 2016. Thessaloniki

R. Tomás

The CERN Accelerator School

Contents

- Introduction to accelerator optics and imperfections
- Measurement algorithms
  - Free and forced oscillations
  - Singular Value Decomposition
  - Data cleaning: denoising and removal of outlayers
  - Spectral analysis
  - β from phase / β from amplitude
- Correction algorithms
  - Orthogonal knobs
  - Local: segment-by-segment
  - Best N corrector problem: MICADO and friends
  - Response matrix approach

Code examples

Code examples require the following.

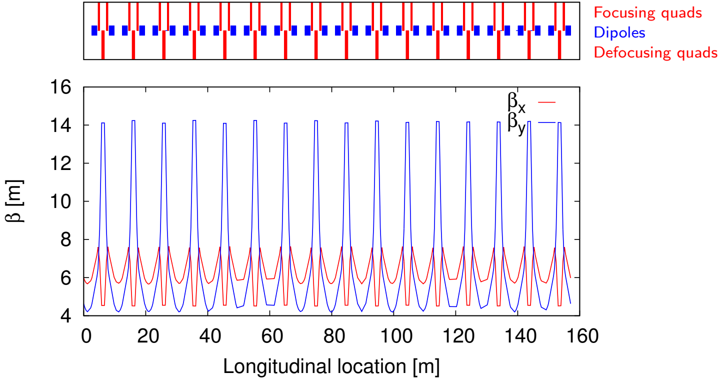
Install python™ (ubuntu® has it by default)

Install plotting and numerical libraries:

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

Or download ANACONDA. which is a very complete Python free distribution with scientific packages.

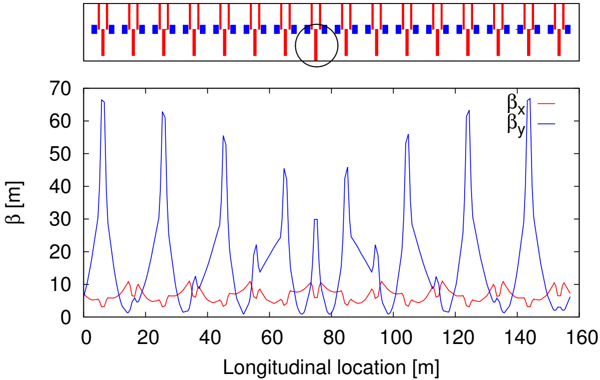
β function in a triplet lattice



Ideal CERN Proton Synchrotron Booster lattice.

Quadrupole strength error & β-beating

$\Delta k / k = 0.1$



β functions change (β-beating$=\frac{\Delta\beta}{\beta} = \frac{\beta_{pert} - \beta_0}{\beta_0}$).
Tunes change too (ΔQ).

E.D. Courant and H.S. Snyder 1957

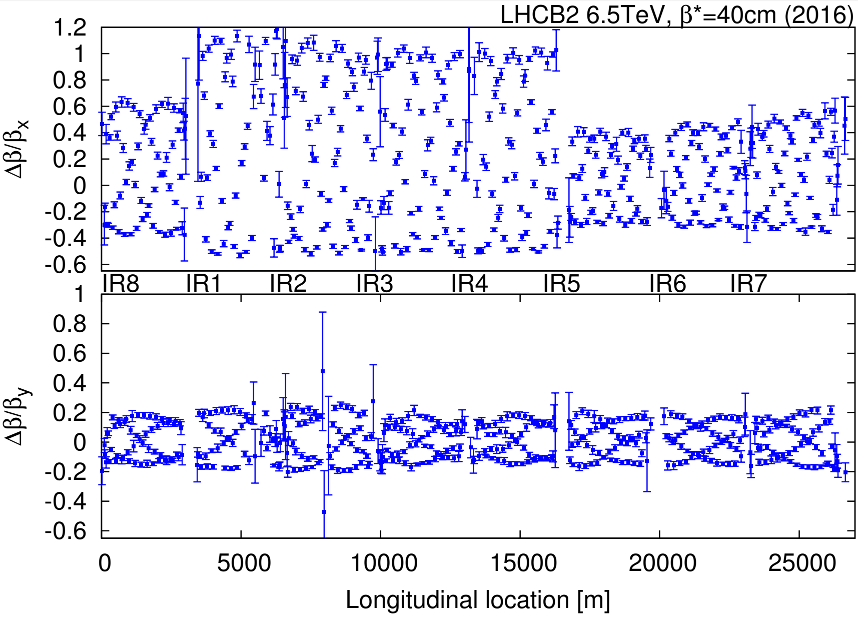
Theory of the Alternating-Gradient Synchrotron [1]:

$$\left(\frac{\Delta\beta}{\beta}\right)_{\rm max} = 4.0\left(\frac{\Delta k}{k}\right)_{\rm rms}$$

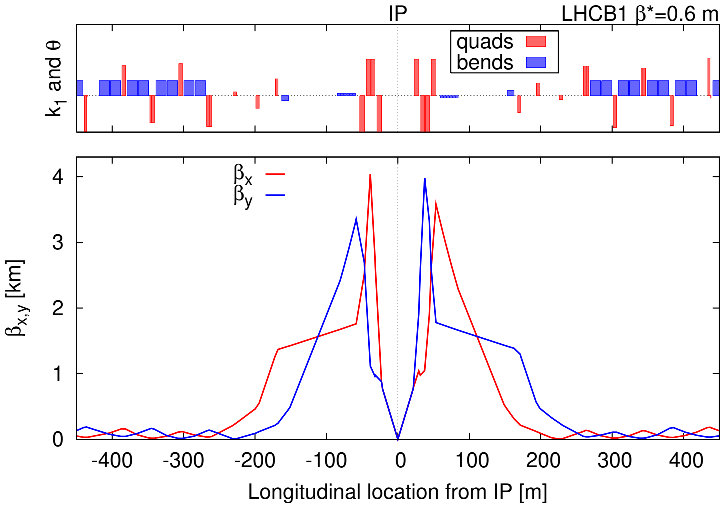
"Thus if the variation in k from magnet to magnet were 1% (...) we would have a β-**beating of 4%**. Any particular machine (...) would be unlikely to be worse by more than factor of 2."

$\rightarrow$ Expected β-beating below 8% for *any machine*

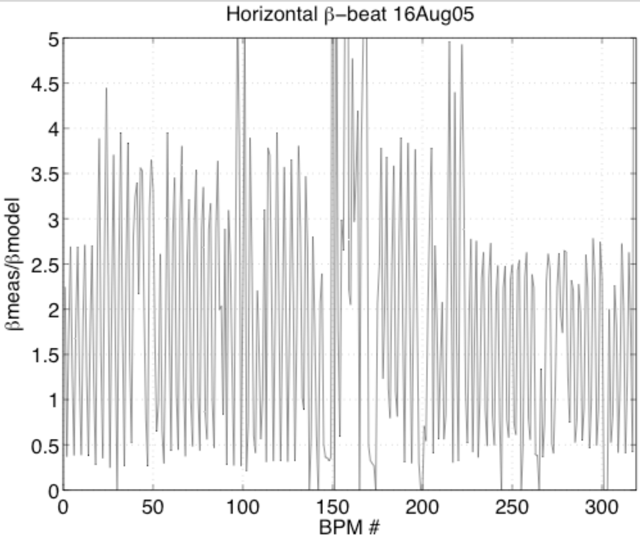
120% in LHC, commissioning 2016



The LHC Interaction Region (IR)



≈400% in PEP-II, commissioning 2005

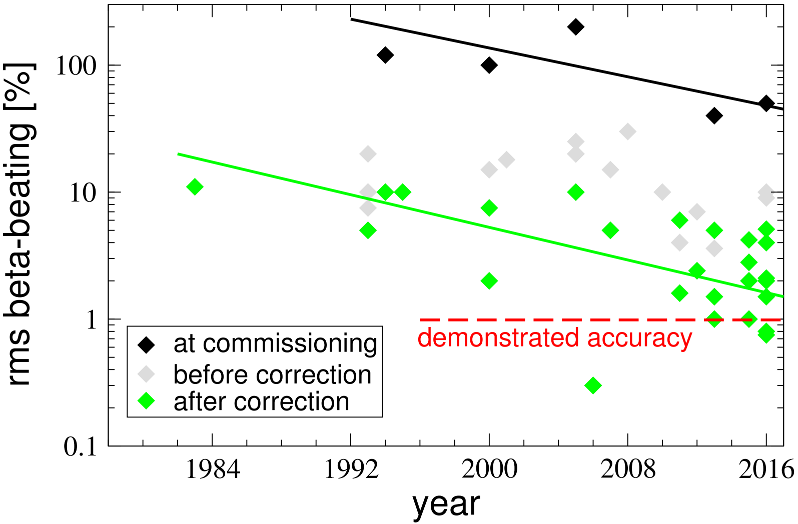


G. Yocky,
SLAC-PUB-12523

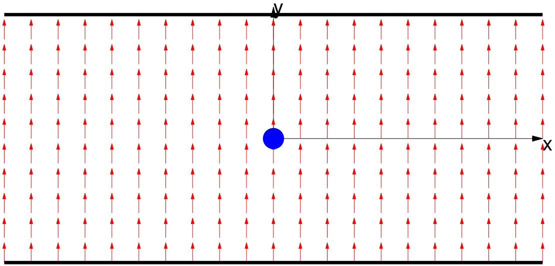
Even $\Delta\beta/\beta \approx 700\%$ was reached when LER tune was pushed closer to the half integer

β-beating versus time

see [2]



Dipole magnetic field

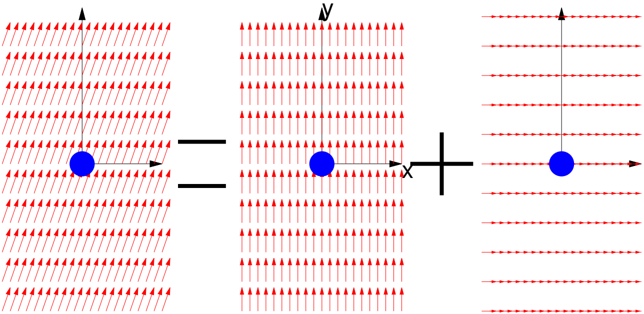


Lorentz force:

$$\vec{F} = q\vec{v} \times \vec{B}$$

Dipole errors

- An error in the strength of a main dipole causes a perturbation on the horizontal closed orbit.
- A tilt error in a main dipole causes a perturbation on the vertical closed orbit.



Orbit perturbation formula

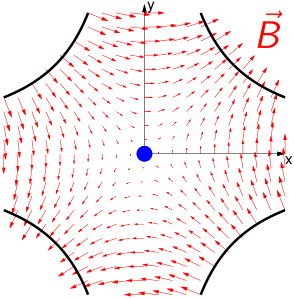
From distributed angular kicks θ_i the closed orbit results in:

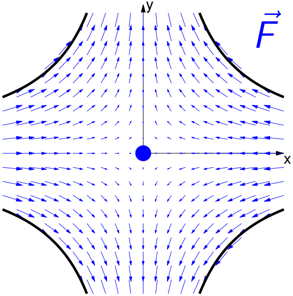
$$CO(s) = \frac{\sqrt{\beta(s)}}{2\sin\pi Q}\sum_i \sqrt{\beta_i}\theta_i \cos(\pi Q - |\phi(s) - \phi_i|)$$

Attention to the denominator $\sin(\pi Q)$ that makes closed orbit to diverge at the integer resonance $Q \in \mathbb{N}$.

Another source of orbit errors is offset quadrupoles.

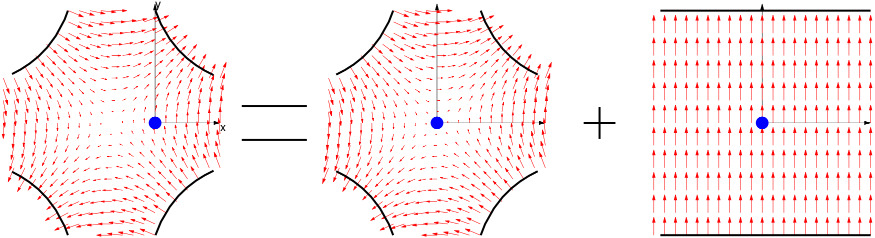
Quadrupole field and force on the beam





Note that $F_x = -kx$ and $F_y = ky$ making horizontal dynamics totally decoupled from vertical.

Offset quadrupole - Feed-down



An offset quadrupole is seen as a centered quadrupole plus a dipole. This is called feed-down.

Quadrupole strength error - Formulae

Tune change (single source):

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

β-beating from many sources:

$$\frac{\Delta\beta}{\beta}(s) \approx \pm\sum_i \frac{\Delta k_i L_i \overline{\beta_i}}{2\sin(2\pi Q)}\cos(2\pi Q - 2|\phi(s) - \phi_i|)$$

Attention to the denominator $\sin(2\pi Q)$ that makes β-beating diverge at the integer and half integer resonances, $2Q \in \mathbb{N}$.

Phase beating and higher orders

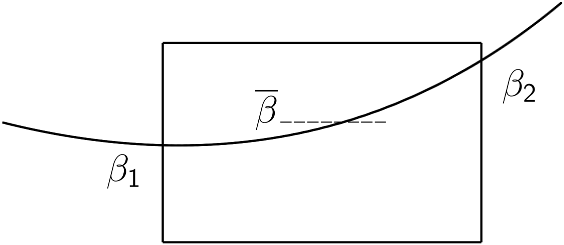
$$\Delta\phi(s_0, s) = \int_{s_0}^{s} \frac{\mathrm{d}s'}{\beta(s')} \left(\frac{1}{1 + \frac{\Delta\beta}{\beta}(s')} - 1 \right)$$

For first and higher order expansions see [3, 4, 5].
β-beating from RDT:

$$f_{2000} = \frac{\sum \Delta k L \overline{\beta_x} e^{2i\phi_x}}{1 - e^{4i\pi Q_x}} + \mathcal{O}(\Delta k^2)$$

$$\frac{\Delta\beta}{\beta}(s) = 2 \sinh |f_{2000}| \Big(\sinh |f_{2000}| + \cosh |f_{2000}| \sin \phi_{2000} \Big)$$

Average beta function in a quad ($\overline{\beta}$)

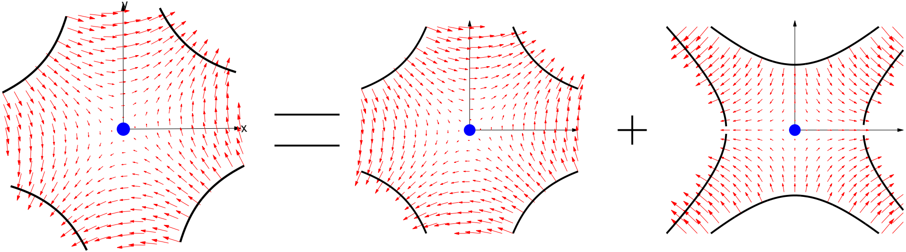


quadrupole (k, L)

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

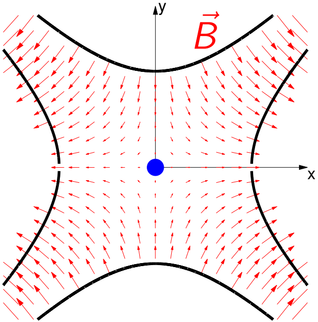
A. Hofmann and B. Zotter [6]. Exact $\overline{\beta}$ depends also on the quadrupole strength k [7].

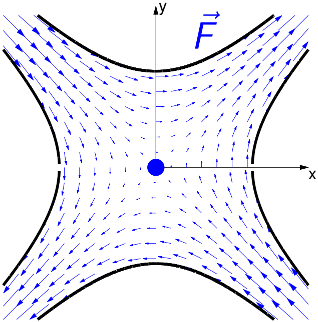
Tilted quadrupole



A tilted quadrupole is seen as a normal quadrupole plus another quadrupole tilted by 45° (this is called a skew quadrupole).

Skew quadrupole → x-y Coupling





Note that $F_x = k_s y$ and $F_y = k_s x$ making horizontal and vertical dynamics to couple.

Transverse coupling in the 1-turn map

In the ideal uncoupled case:

$$\begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix}_f = \begin{pmatrix} M_{11} & M_{12} & 0 & 0 \\ M_{21} & M_{22} & 0 & 0 \\ 0 & 0 & M_{33} & M_{34} \\ 0 & 0 & M_{43} & M_{44} \end{pmatrix} \begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix}_i$$

In presence of coupling:

$$\begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix}_f = \begin{pmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{pmatrix} \begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix}_i$$

Motion with coupling

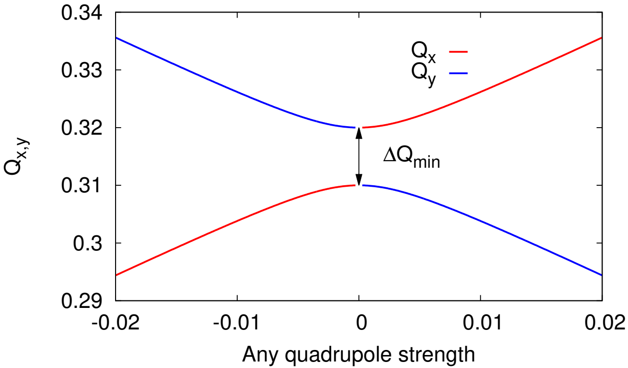
To first order in the coupling the transverse motion can be approximated as [8, 9]

$$
\begin{aligned}
x(N,s) &\approx \sqrt{\beta_x(s)}\,\Re\Big\{\sqrt{\epsilon_x}e^{i(2\pi Q_x N+\phi_x(s)+\phi_{x0})} \\
&\quad -2if_{1010}\sqrt{\epsilon_y}e^{-i(2\pi Q_y N+\phi_y(s)+\phi_{y0})} \\
&\quad -2if_{1001}\sqrt{\epsilon_y}e^{i(2\pi Q_y N+\phi_y(s)+\phi_{y0})}\Big\} \\
f_{\substack{1010\\1001}} &= \frac{\oint_s^{s+C} ds' k_s\sqrt{\beta_x\beta_y}e^{i(\phi_x\pm\phi_y)}}{4(1-e^{2\pi i(Q_x\pm Q_y)})}
\end{aligned}
$$

f_{1001} drives the difference resonance $Q_x - Q_y = N$ and f_{1010} the sum resonance $Q_x + Q_y = N$

Bothering effects of coupling

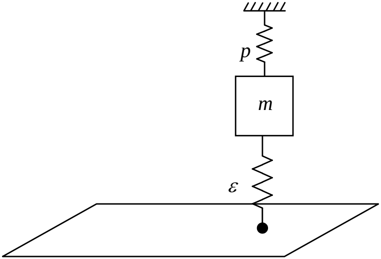
Lepton machines: increases the vertical equilibrium emittance.
Hadron machines: makes it impossible to approach tunes below ΔQ_{min}

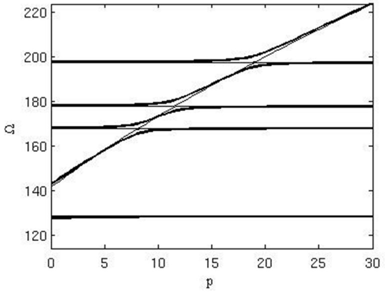


In solid mechanics this is called **mode veering**.

Example of mode veering

Simply supported plate with an attached oscillator:





http://past.isma-isaac.be/downloads/isma2012/papers/isma2012_0137.pdf

ΔQ_{min} formula

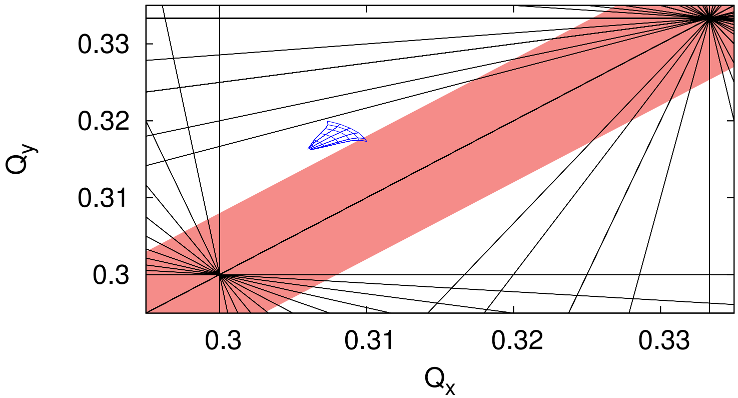
$$\Delta Q_{\mathrm{min}} = \left| \frac{1}{2\pi} \sum_j k_{s,j} L_j \sqrt{\beta_x \beta_y} e^{-i(\phi_x - \phi_y) + is(\hat{Q}_x - \hat{Q}_y)/R} \right|$$

$$\begin{aligned} \Delta Q_{\mathrm{min}} &= \left| \frac{4(\hat{Q}_x - \hat{Q}_y)}{2\pi R} \oint \mathrm{d}s f_{1001} \mathrm{e}^{-i(\phi_x - \phi_y) + is(\hat{Q}_x - \hat{Q}_y)/R} \right| \\ &\lesssim 4 |\hat{Q}_x - \hat{Q}_y| \overline{|f_{1001}|} \end{aligned}$$

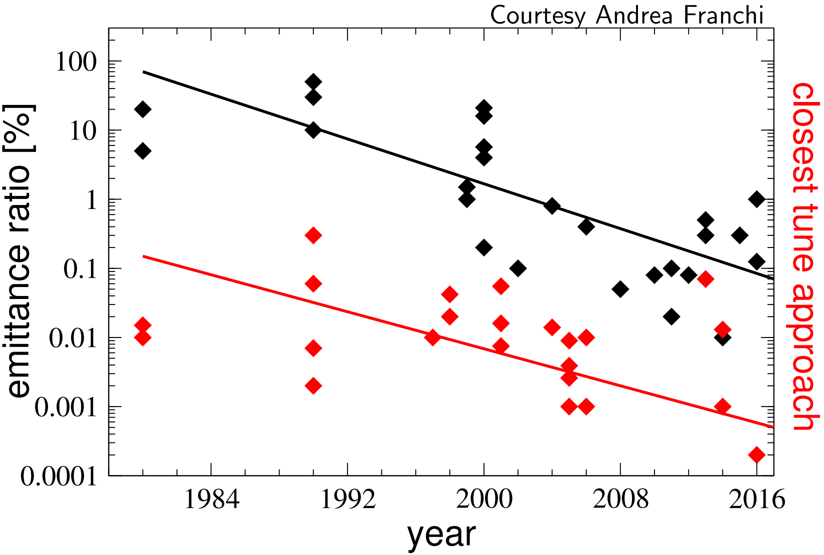
f_{1001} defines phase space and the stopband.
See [10, 11, 12] for further details.

$\Delta Q_{\min}$ limits the resonance-free space

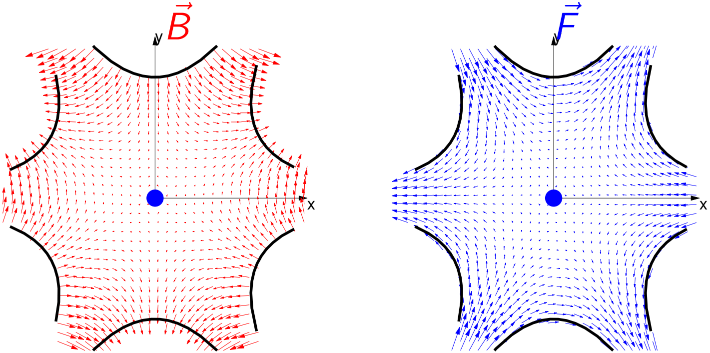
LHC beam-beam tune footprint and a hypothetical large coupling:



Coupling control versus time



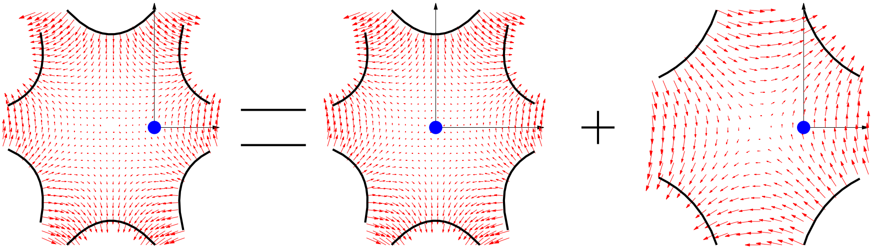
Sextupole field and force



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

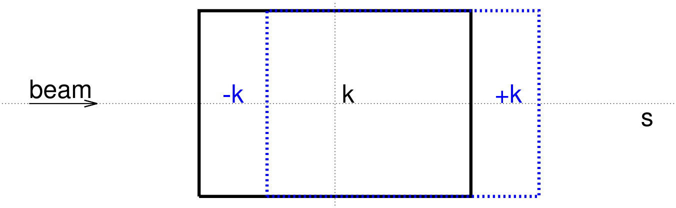
Sextupoles are needed to compensate chromaticity:
Q'=dQ/dδ, with $\delta = (p - p_0)/p_0$

Offset sextupole



A sextupole horizontally (vertically) displaced is seen as a centered sextupole plus an offset quadrupole (skew quadrupole). Offset sextupoles are also sources of quadrupole and skew quadrupole errors.

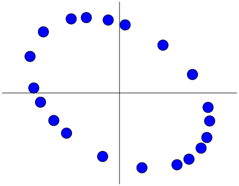
Longitudinal misalignments

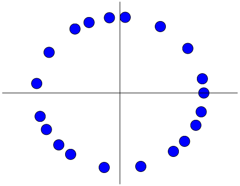


Longitudinal misalignments can be seen as perturbations at both ends of the magnet with opposite signs. Tolerances are generally larger for longitudinal misalignments.

Phase-space turn-by-turn motion

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





Floquet Normal Form

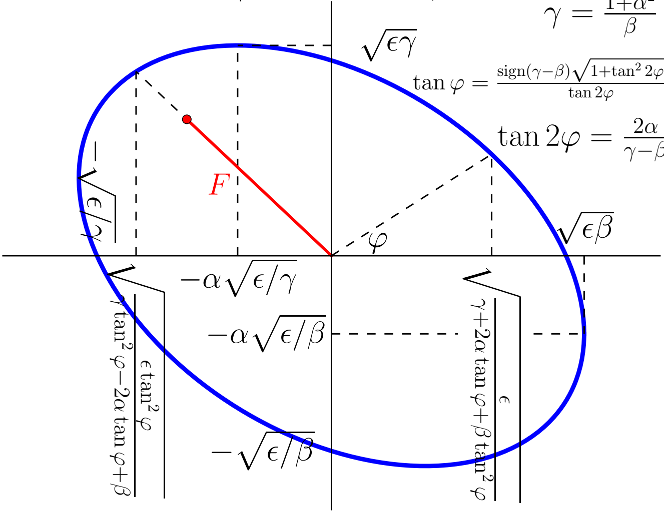
Phase-space ellipse

$$\epsilon = \gamma x^2 + 2\alpha x x' + \beta x'^2$$

$$\gamma = \frac{1+\alpha^2}{\beta}$$

$$\tan\varphi = \frac{\mathrm{sign}(\gamma-\beta)\sqrt{1+\tan^2 2\varphi}-1}{\tan 2\varphi}$$

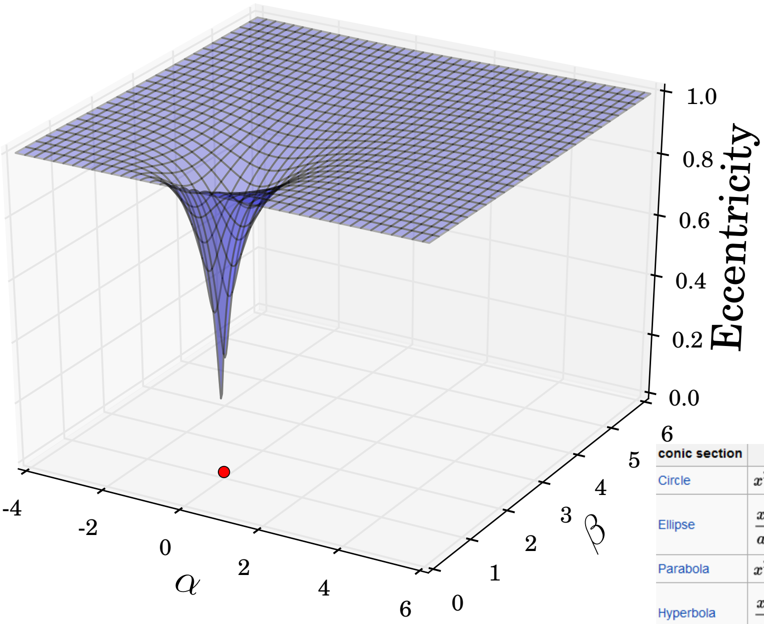
$$\tan 2\varphi = \frac{2\alpha}{\gamma-\beta}$$



$$\text{eccentricity}^2 = \frac{\alpha}{\tan\varphi}\,\frac{(1+\tan^2\varphi)^2}{\gamma+2\alpha\tan\varphi+\beta\tan^2\varphi}$$

$$F^2 = \epsilon\frac{(\gamma^2+\beta^2-2+2\alpha^2)(1-\tan^4\varphi)}{(\gamma-\beta)[\tan^2\varphi(\gamma^2+\beta^2)+(1-\alpha^2)(\tan^4\varphi+1)]}$$

Ellipse eccentricity vs. α and β



conic section	equation	eccentricity
Circle	$x^2 + y^2 = r^2$	0
Ellipse	$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$	$\sqrt{1 - \frac{b^2}{a^2}}$
Parabola	$x^2 = 4ay$	1
Hyperbola	$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$	$\sqrt{1 + \frac{b^2}{a^2}}$

α, β, ϵ from turn-by-turn data & SVD

SVD of x, x' turn-by-turn data:

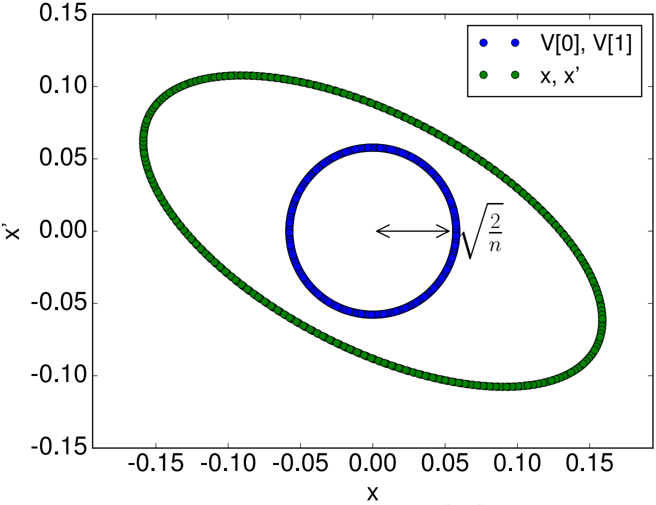
$$\begin{pmatrix} x(1) & x(2) & x(3) & \dots & x(n) \\ x'(1) & x'(2) & x'(3) & \dots & x'(n) \end{pmatrix}_{2\times n} = U_{2\times 2} S_{2\times 2} V^T_{2\times n}$$

U and V are unitary, S is diagonal. $V_{2\times n}$ is turn-by-turn motion in a circle (like normal form) in an arbitrary phase origin. There must be a rotation $R(\theta)$, $USV^T = USRR^{-1}V^T$, that brings $R^{-1}V^T$ to the Floquet Normal Form:

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

θ is determined to make zero the element (1,2).

Looking at x, x', V and ϵ



n is the number of turns, $\epsilon = \dfrac{\det(S)}{n/2}$

α, β, ϵ from turn-by-turn data – code

```python
import numpy as np

def getbeta(x,px):
    U, s, V = np.linalg.svd([x,px])     # SVD
    N = np.dot(U,np.diag(s))
    theta = np.arctan(-N[0,1]/N[0,0])   # Angle of rotation of matrix
    c = np.cos(theta); s = np.sin(theta)
    R = [ [c, s] , [-s, c] ]
    X = np.dot(N,R)                     # Floquet up to 1/det(USR)
    betx = np.abs(X[0,0]/X[1,1])
    alfx = X[1,0]/X[1,1]
    ex=s[0]*s[1]/(len(x)/2.)           # emit = det(S)/(n/2)
    return betx, alfx, ex

alpha = 0.2
beta = 1.
ex = 2e-3
Q = 0.31
Nturns = 600
x = np.sqrt(beta*ex)*np.cos(2*np.pi*Q*np.arange(0,Nturns)) #easy tracking
px = -alpha*x/beta + np.sqrt(ex/beta)*np.sin(2*np.pi*Q*np.arange(0,Nturns))
betx, alfx, exc = getbeta(x,px)
```

1st version of the code: P. Gonçalves Jorge and X. Buffat, CERN-THESIS-2016-317.

How to compute β from beam data?

- Beam Position Monitors (BPMs) measure transverse beam centroid position turn-by-turn
- Excited betatron motion is required either via a single kick or via a resonant excitation

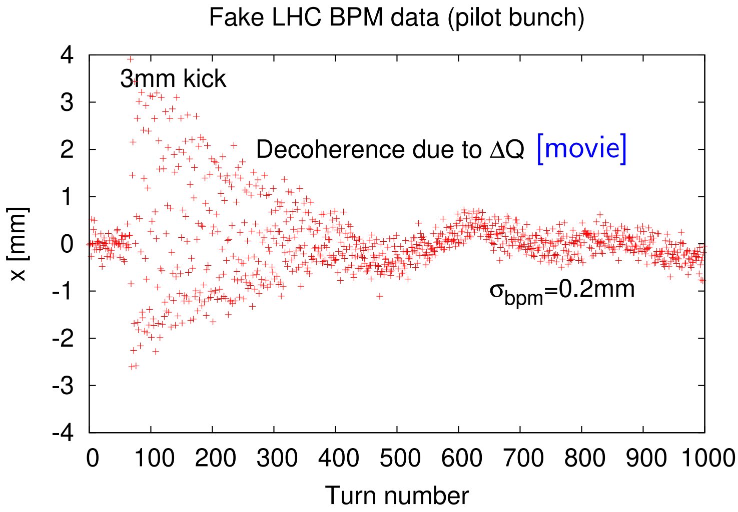
$$x(N,s) = \sqrt{\beta_x(s)\epsilon_x}\cos(2\pi Q_x N + \phi_x(s)) + CO(s)$$

β and ϕ are related by:

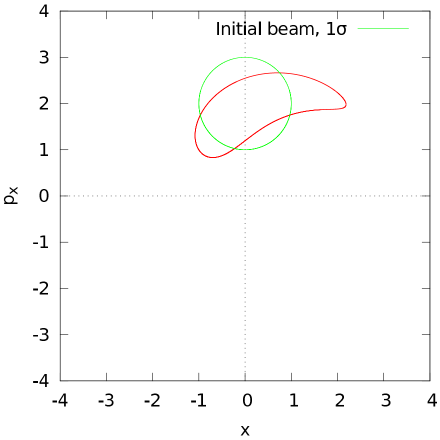
$$\phi_{0\to 1} = \phi(s_1) - \phi(s_0) = \int_{s_0}^{s_1} \frac{\mathrm{d}s}{\beta(s)}$$

so β and ϕ carry the same information, ϕ being a BPM calibration independent observable.

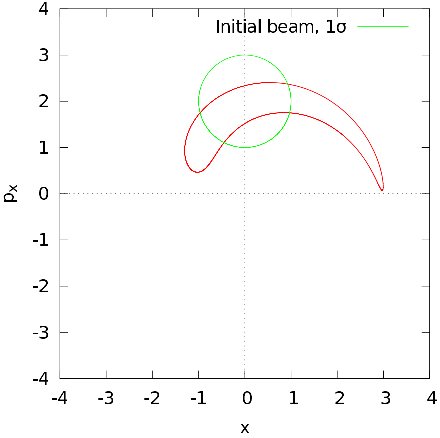
Turn-by-turn data from single kick



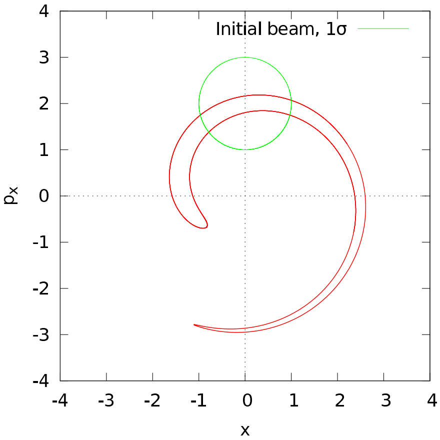
Decoherence



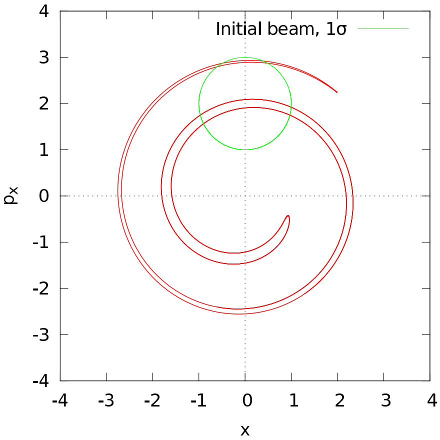
Decoherence



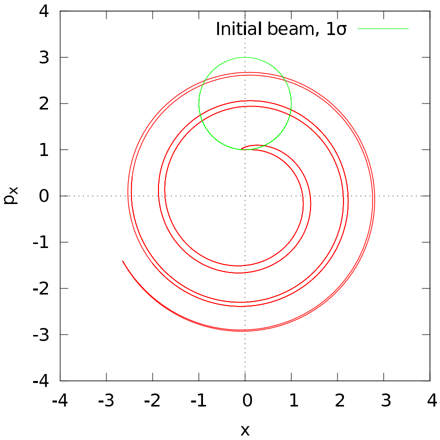
Decoherence



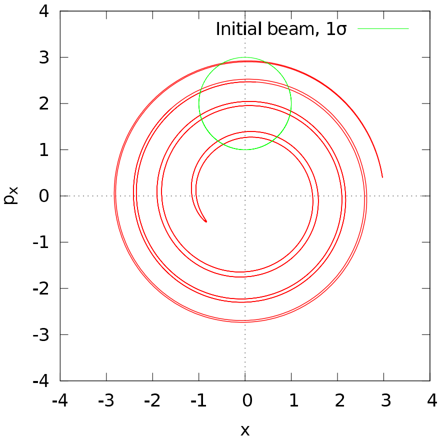
Decoherence



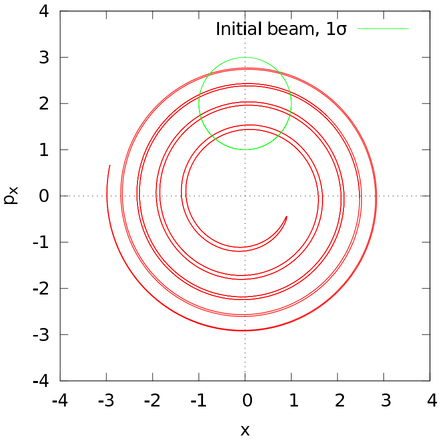
Decoherence



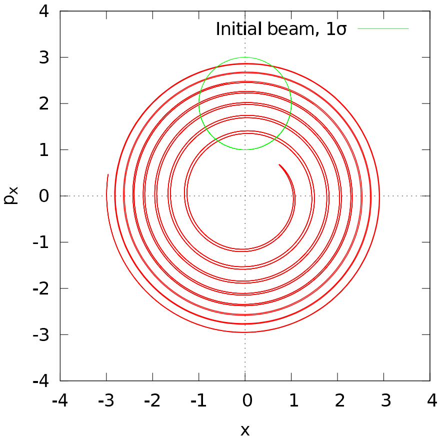
Decoherence



Decoherence

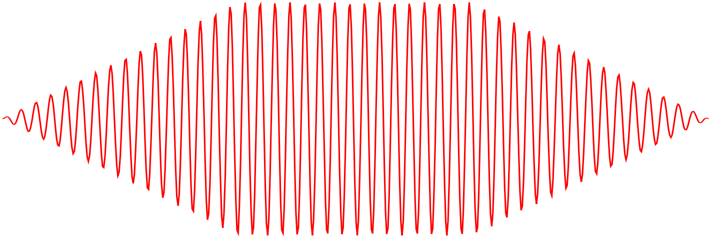


Decoherence



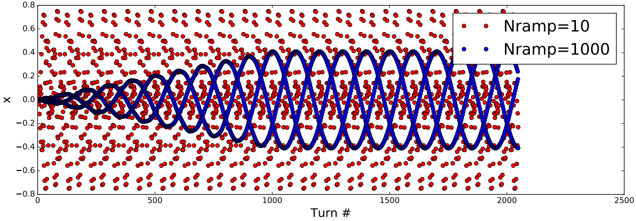
Forced oscillations with AC dipole

- An AC dipole forces betatron oscillations
- If addiabatically ramped up & down causes no emittance blow up (contrary to kick)
- Can be used as many times as needed with the same beam

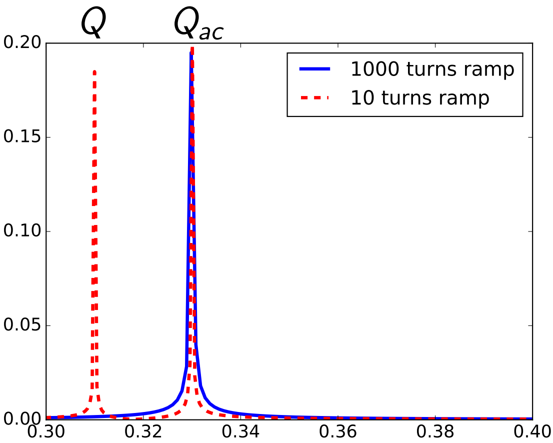


Simulate your own AC dipole

```python
from numpy import *
import matplotlib.pyplot as plt

Q = 0.31          # Machine tune (fractional part)
Qac = Q + 0.02    # AC dipole tune
q = 2*pi*Q
R = array([[cos(q), -sin(q)],[sin(q), cos(q)]]) #1 turn map
x=[[0.,0.]]   # initial x, px
Nramp = 1000    # Number of turns to ramp up AC dipole strength
Nturn = 2048    # Number of turns to track

def ramp(j):  # define the linear ramp
    return min(1, j*1.0/Nramp)

for i in range(Nturn):    # tracking loop   R x + cos(QacN)
        x.append( dot(R,x[-1]) + ramp(i)*array([0, 0.1*cos(Qac*i*2*pi)]))

F = np.fft.fft(array(x)[Nramp:].T[0])  # FFT data after AC ramp
```

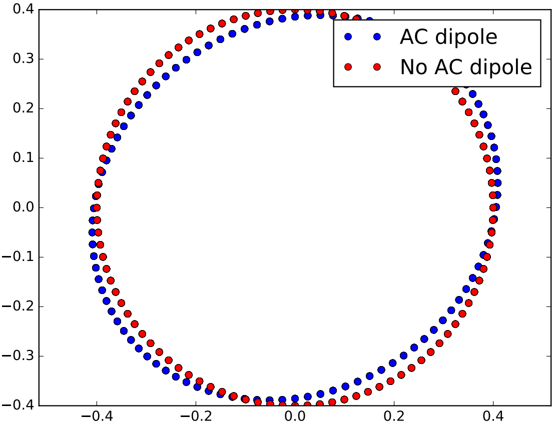


FFT after AC ramp (adiabaticity check)



Seeing Q in the particle motion means that the AC dipole transferred energy to the particle, so AC dipole ramp was not adiabatic.

...and check phase space (free Vs AC dip)




```
xnoac=[[0.4,0.]]
for i in range(2048):
        xnoac.append( dot(R,xnoac[-1]))
```

What is happening?

Forced oscillation $\neq$ Free oscillation

- β functions differ as if there was a quadrupole at the location of the AC dipole [3]
- Non-linear dynamics also deviate from free motion [18, 19],
- including Dynamic Aperture [20].
- Free optics have to be reconstructed from measurements with forced oscillations.

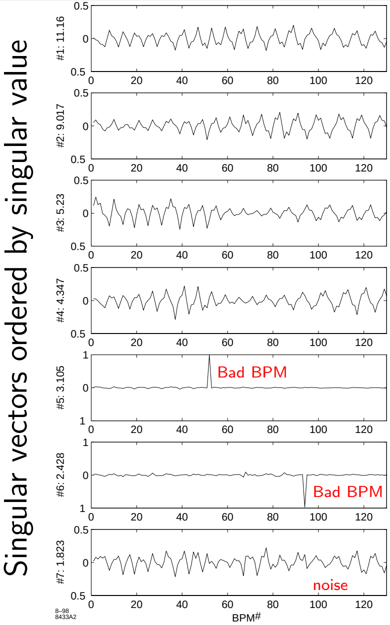
Denoising via SVD

$$R = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \\ 0 & 0 & 0 \end{pmatrix} V^T$$

Imagine $\sigma_3 \ll \sigma_2 \leq \sigma_1$, then just neglect σ_3 and reconstruct R:

$$R_{denoised} = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} V^T$$

SLC: Cleaning with SVD, 1999



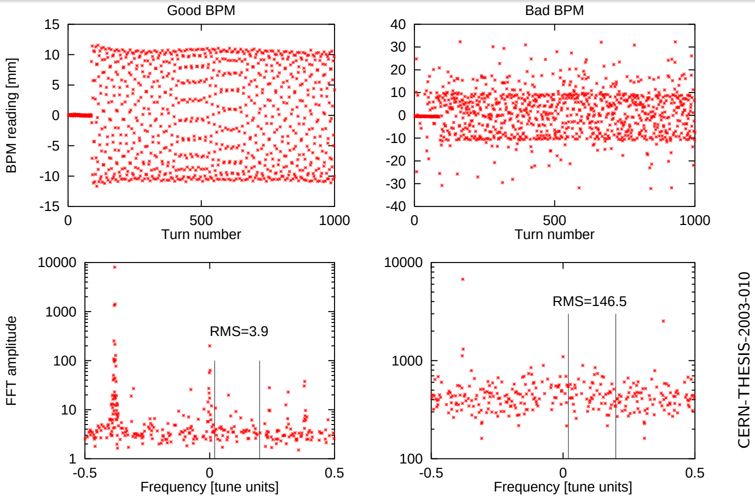
$$\underset{\substack{bpm\\ matrix}}{B_{t-b-t}} = USV^T$$

Bad BPMs easily identified as uncorrelated signals.

Noise removed by cutting low singular values

J. Irwin et al., Phys. Rev. Letters **82**, 8

Good and bad BPM signals (SPS)

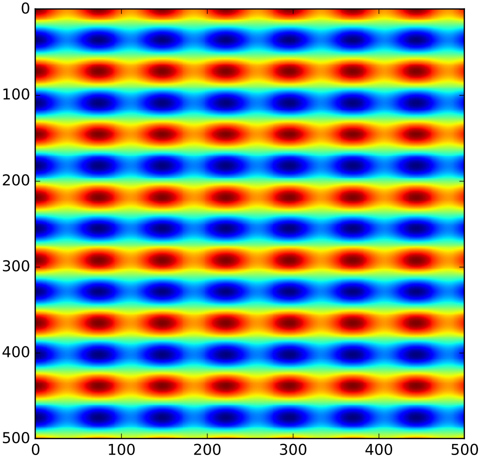


BPM issues required bad BPM detection techniques.
The **RMS** in a FFT window is a good indicator.

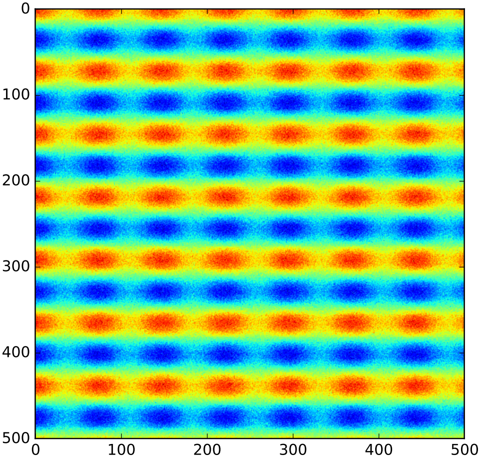
Denoising data with SVD

```
import matplotlib.pyplot as plt
from scipy import misc,ndimage
import numpy as np
from numpy.linalg import svd

#Generating ideal (fake) Beam Position data
im = np.zeros((500, 500))
for i in range(500):
    for j in range(500):
        im[i,j]=np.cos(i*0.0137*2*np.pi) *(np.cos(0.00678*j*2*np.pi)**2+1)

#Adding noise like measurement error
im = im + 0.2 * np.random.randn(*im.shape)

#Denoising  with Singular Value Decomposition
k=2
U,s,V=svd(im, full_matrices=False)
rim = np.dot(U[:,:k], np.dot(np.diag(s[:k]),V[:k,:]))
```

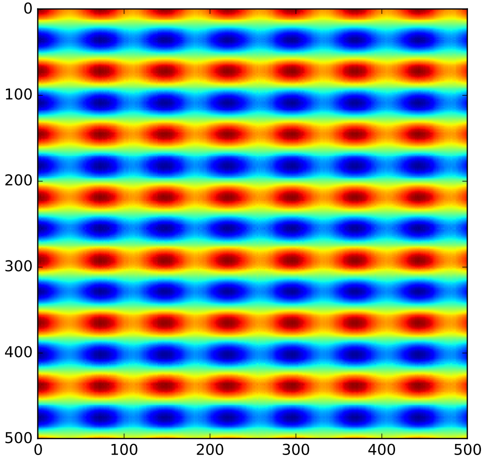
Ideal (fake) BPM Data



Adding noise to BPM data

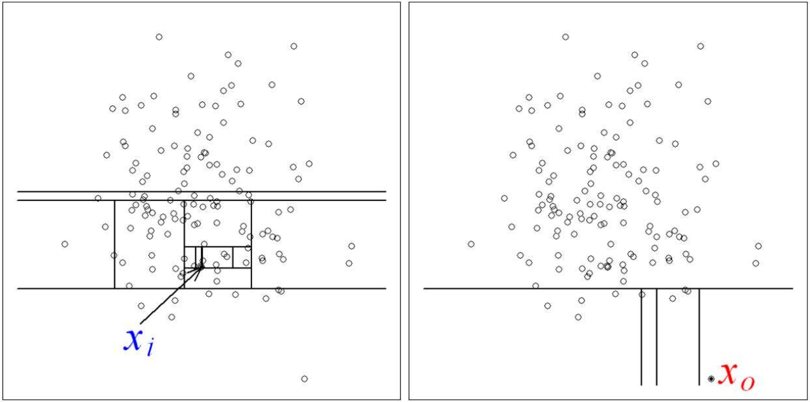


Denoised data with SVD – Magic!



Outlayer detection: Isolation forest (IF)

Principle: By applying random divisions to a data set, outlayers are identified
by needing fewer divisions for isolation than core data.

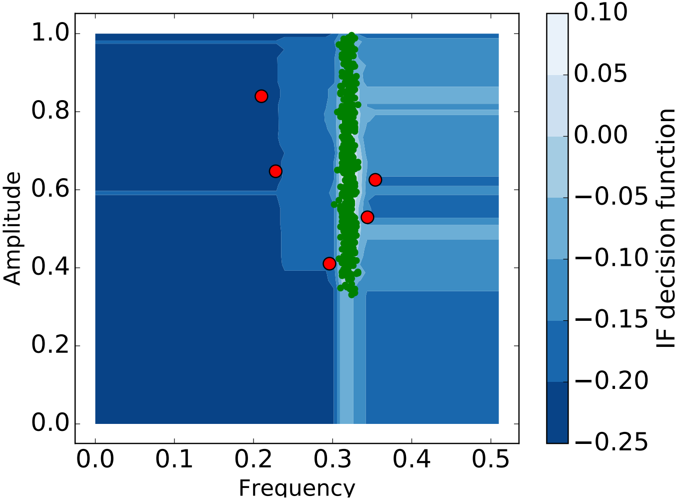


(a) Isolating x_i (b) Isolating x_o

Outlayer detection: Isolation forest

```python
import numpy as np
from sklearn.ensemble import IsolationForest

N_TURNS = 500
N_BPMS = 500
# generate bpm data with some bad signal - different tune, additional noise
bad_bpms_idx = [1, 10, 20, 30, 40]
im = np.zeros((N_TURNS, N_BPMS))
for bpm in range(N_BPMS):
    err= 0.05 * np.random.randn()
    amp=(np.cos(0.00678 * bpm * 2 *np.pi) ** 2 + 1)  # sqrt(beta e)
    for turn in range(N_TURNS):
        if bpm in bad_bpms_idx:   # A bad BPM has different tune and more noise
          im[turn,bpm]=amp*np.cos(turn*(0.32+err)*2*np.pi)+0.3*np.random.randn()
        else:                 # Good BPM
          im[turn,bpm]=amp*np.cos(turn*(0.32+err/10)*2*np.pi) + 0.1*np.random.randn()

# extract frequency and amplitude - features - from bpm signal
amplitudes = [np.max(x) for x in np.abs(np.fft.rfft(im.T))/N_TURNS]
frequencies= np.array([np.argmax(x) for x in np.abs(np.fft.rfft(im.T))])*1.0/N_TURNS
features = np.vstack((frequencies, amplitudes)).T

# fit Isolation Forest model to the data and detect anomalies (contamination is the fraction of anomalies)
iforest = IsolationForest(n_estimators=10, contamination=0.01)
outlier_detection = iforest.fit(features).predict(features) # Bad BPMs ==-1
```

Isolation forest illustration



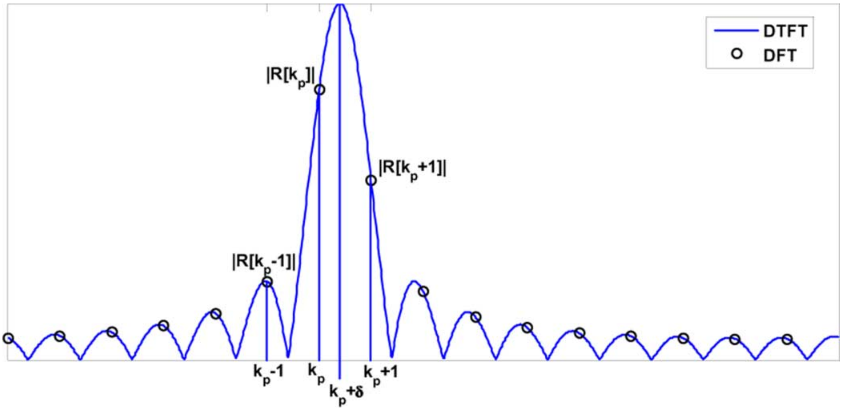
Almost ready to measure β

Bad BPMs are discarded and noise is reduced.

$$x(N,s) = \sqrt{\beta_x(s)\epsilon}\cos(2\pi Q_x N + \phi_x(s)) + CO(s)$$

In absence of decoherence FFT on x(N,s) should give us a frequency Q_x with amplitude $\sqrt{\beta_x(s)\epsilon}$ and phase $\phi_x(s)$, but how accurate?

3-point frequency interpolation



IEEE Signal processing letters **18**, No. 6, JUNE 2011 351

3-point frequency interpolation

Parabolic Interpolation [1]	$\widehat{\delta} = (\lvert R_{k+1}\rvert - \lvert R_{k-1}\rvert)/(4\lvert R_k\rvert - 2\lvert R_{k-1}\rvert - 2\lvert R_{k+1}\rvert)$
Quinn, [4] 1995	$\alpha_1 = \text{Real}(R_{k-1}/R_k), \quad \alpha_2 = \text{Real}(R_{k+1}/R_k)$ $\delta_1 = \alpha_1/(1-\alpha_1), \quad \delta_2 = \alpha_2/(1-\alpha_2)$ if $\delta_1 > 0$ and $\delta_2 > 0$, $\widehat{\delta} = \delta_2$ else $\widehat{\delta} = \delta_1$
MacLeod, [3]	$d = \text{Real}(R_{k-1}R_k^* - R_{k+1}R_k^*)/\text{Real}(2\lvert R_k\rvert^2 + R_{k-1}R_k^* + R_{k+1}R_k^*)$ $\widehat{\delta} = (\sqrt{1+8d^2} - 1)/(4d)$
Jacobsen, [7]	$\widehat{\delta} = \text{Real}\{(R_{k-1} - R_{k+1})/(2R_k - R_{k-1} - R_{k+1})\}$
Jacobsen with Bias Correction	$\widehat{\delta} = \frac{\tan(\pi/N)}{\pi/N}\text{Real}\{(R_{k-1} - R_{k+1})/(2R_k - R_{k-1} - R_{k+1})\}$

IEEE Signal processing letters **18**, No. 6, JUNE 2011 351

E. Asseo, CERN PS/85–3(1985): $\lvert R_{k+1}\rvert/(\lvert R_k\rvert + \lvert R_{k+1}\rvert)$

CERN-SL-96-048: $\frac{1}{\pi}\text{atan}\{\lvert R_{k+1}\rvert \sin(\pi/N)/(\lvert R_k\rvert + \lvert R_{k+1}\rvert \cos(\pi/N)\}$

NAFF

- ★ Instead of interpolating, find Q that maximizes $|\sum x(N)e^{i2\pi QN}|$
- ★ then **iterate** by subtracing the found frequency component from $x(n)$

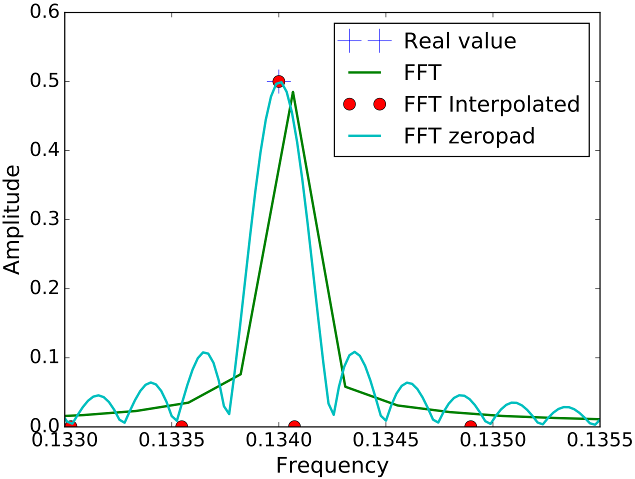
Some codes to do this:

- ★ NAFF: Icarus **88**, Issue 2, 1990. https://pypi.org/project/PyNAFF/
- ★ Sussix: CERN SL/Note 98-017, 1998
- ★ Harpy: IPAC2018-THPAF045 (with Jacobsen interpolation instead of maximization to speed-up), see next slide

Harpy

```python
import numpy as np
PI2I = 2 * np.pi * complex(0, 1)

def harpy(samples, num_harmonics):
    n = len(samples)
    int_range = np.arange(n)
    coefficients = []
    frequencies = []
    for _ in range(num_harmonics):
        frequency = _jacobsen(np.fft.fft(samples), n) # Find dominant freq.
        exponents = np.exp(-PI2I * frequency * np.arange(n))
        coef =  np.sum(exponents*samples)/n     # compute amplitude and phase
        coefficients.append( coef )
        frequencies.append(frequency)
        new_signal = coef * np.exp(PI2I * frequency * int_range)
        samples = samples - new_signal  # Remove dominant freq.
    coefficients, frequencies = zip(*sorted(zip(coefficients, frequencies),
        key=lambda tuple: np.abs(tuple[0]), reverse=True))
    return frequencies, coefficients

def _jacobsen(dft, n): # Interpolate to find dominant freq.
    k = np.argmax(np.abs(dft))
    delta = np.tan(np.pi / n) / (np.pi / n)
    kp = (k + 1) % n
    km = (k - 1) % n
    delta = delta * np.real((dft[km]-dft[kp])/(2*dft[k] - dft[km] - dft[kp]))
    return (k + delta) / n

N=4096;  i = 2 * np.pi * np.arange(N)
data = np.cos(0.134 * i) + np.cos(0.244 * i) + 0.01 * np.random.randn(4096)
freqs, coeffs = harpy(data, 300)
```

Jacobsen interpolation & zero padding



Zero padding: just add zeros to original signal

```
data_zeropad=np.pad(data, (0, 9*N), 'constant')
f_zeropad=np.abs(np.fft.fft(data_zeropad)/(N))

```

FFT Vs NAFF Vs Sussix with noise

N. Biancacci et al, PRAB **19**, 054001 (2016):

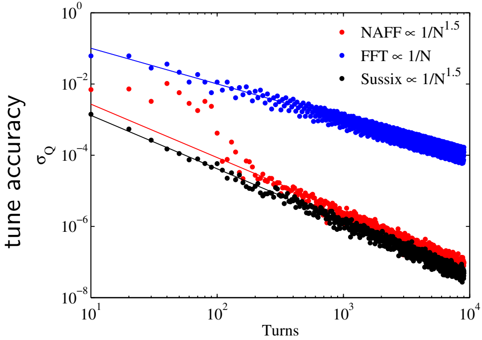


FIG. 5. Uncertainty in the tune determination with NAFF, FFT and SUSSIX versus number of turns calculated up to $N \simeq 10^4$ turns considering amplitude Gaussian noise of NSR $= 5\%$. Dots are the simulated data, lines are the fits. SUSSIX and NAFF both implement a hanning window.

Phase advance measurement

The phase advance between 2 BPMs $\phi_{ij} = \phi_j - \phi_i$ is a fundamental optics observable, it is model and BPM calibrationi independent.
Care with averaging many measurements is needed, a safe approach is the **circular mean**:

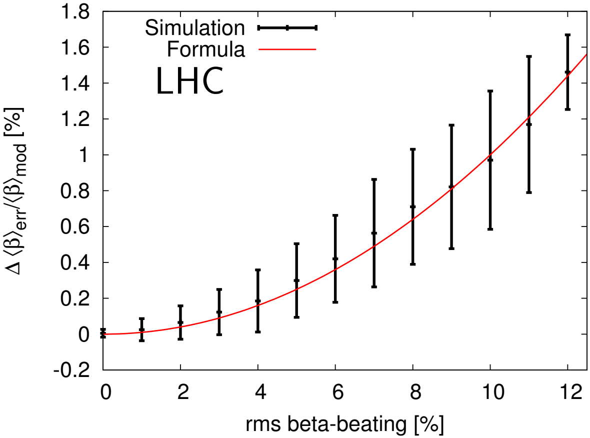
$$\overline{\alpha} = \mathrm{atan2}\left(\frac{1}{n}\sum_i^n \sin\alpha_i,\ \frac{1}{n}\sum_i^n \cos\alpha_i\right)$$

```
from scipy.stats import circmean
import numpy as np
circmean([0., 2*np.pi])
```

output: 2π

Ring average β with random errors

The average β in presence of random errors is well behaved [14]: $\left\langle \frac{\Delta\beta}{\beta} \right\rangle = \mathrm{rms}^2\left(\frac{\Delta\beta}{\beta}\right)$



random errors are defocusing!

β from amplitude

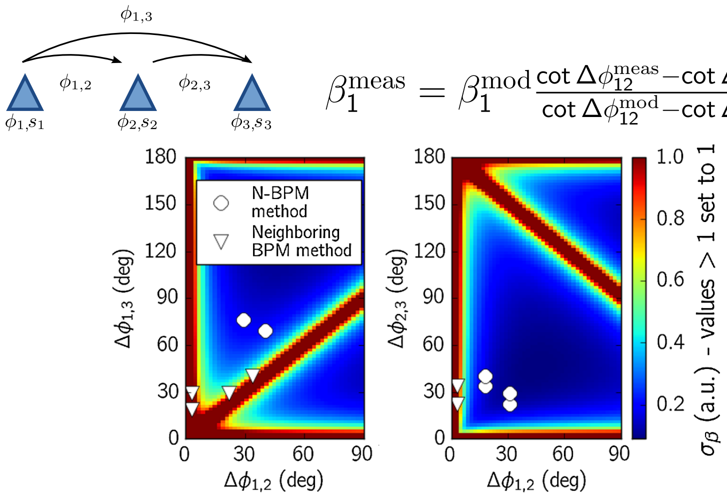
$$x(N,s) = \sqrt{\beta_x(s)\epsilon}\cos(2\pi Q_x N + \phi_x(s)) + CO(s)$$

- 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.
- ϵ can be computed with

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

- Biggest limitation of this technique is BPM calibration errors.

β from phase



$$\beta_1^{\text{meas}} = \beta_1^{\text{mod}} \frac{\cot\Delta\phi_{12}^{\text{meas}} - \cot\Delta\phi_{13}^{\text{meas}}}{\cot\Delta\phi_{12}^{\text{mod}} - \cot\Delta\phi_{13}^{\text{mod}}}$$

3-BPM method [15], N-BPM method [16] & analytical N-BPM method [17].

Random and systematic errors

Error analysis is fundamental. β from phase is model dependent. In [16] impact from model uncertainties are assessed with Montecarlo. In [17] an analytical approach is taken:

$$\beta_l(s_i) \approx \frac{\cot\phi_{ij_l} - \cot\phi_{ik_l}}{\cot\phi^{\mathrm{m}}_{ij_l} - \cot\phi^{\mathrm{m}}_{ik_l} + \bar{g}_{ij_l} - \bar{g}_{ik_l}} [\beta^{\mathrm{m}}(s_i) - 2\alpha^{\mathrm{m}}(s_i)\delta s_i]$$

$\overline{g}_{ik}$ and δs_i contain possible model perturbations. Random errors on ϕ_{ik} and their correlations are also considered.

Momentum reconstruction

BPMs only measure x. With 2 nearby we can reconstruct p_x in the Floquet Normal Form:

$$\begin{aligned}
\hat{x}_1(N) &= \cos(2\pi Q_x N + \phi_1) \\
\hat{x}_2(N) &= \cos(2\pi Q_x N + \phi_2) \\
\hat{p}_{x1}(N) &= \sin(2\pi Q_x N + \phi_1) \\
&= \frac{\hat{x}_2(N)}{\cos\delta} + \hat{x}_1(N)\tan\delta
\end{aligned}$$

with $\delta = \phi_2 - \phi_1 - \pi/2$.

Measurement of Resonance Driving Terms

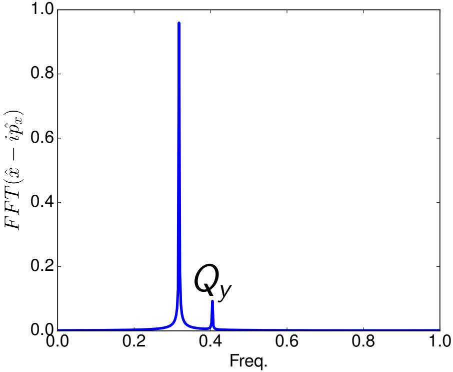
From [8]:

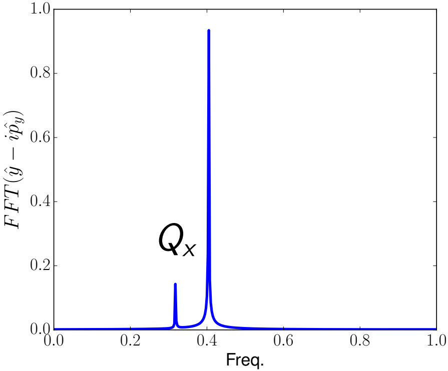
$$\hat{x}_1 - i\hat{p}_{x1} = e^{i2\pi Q_x N} - 2i\sum j f_{jklm} \epsilon_x^{\frac{j+k-2}{2}} \epsilon_y^{\frac{l+m}{2}} e^{i2\pi N[(1-j+k)Q_x+(m-l)Q_y]+i\varphi}$$

So f_{jklm} can be measured from the complex FFT of $\hat{x}_1 - i\hat{p}_{x1}$. Let's illustrate this with coupling.

Coupling RDT

Using 2 nearby double-plane BPMs to reconstruct $\hat{x}_1 - i\hat{p}_{x1}$ and $\hat{y}_1 - i\hat{p}_{y1}$:



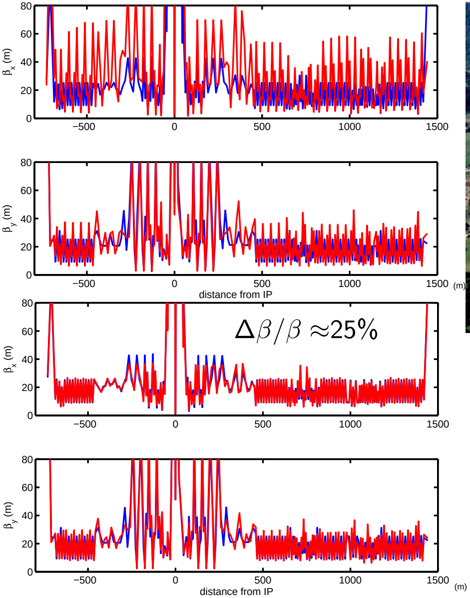


$$2|f_{1001}| = \sqrt{\mathrm{amp}_x(Q_y)\mathrm{amp}_y(Q_x)}$$

PEP-II, from ϕ to virtual model to β

before corrections

after corrections



Using SVD modes

Y. Yan et al, SLAC-PUB-11925 2006

Farey sequences (1802)

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 $N \rightarrow$ **Resonances of order** N **or lower** (in one plane)

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

Some properties of Farey sequences

- 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:

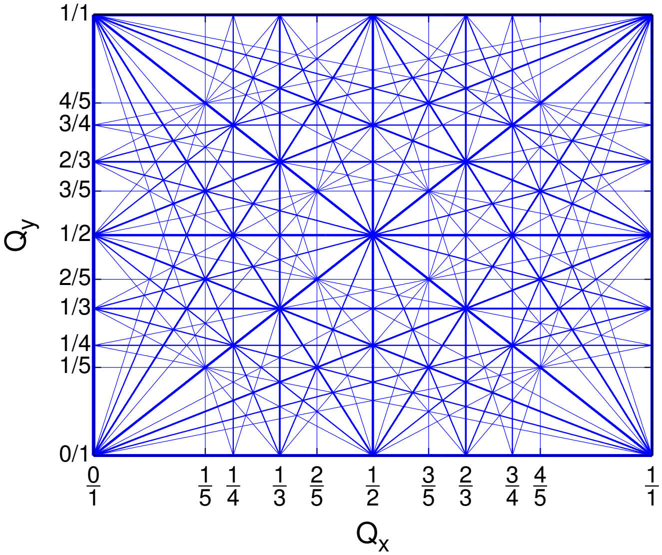
$$\frac{a+c}{b+d}$$

- The number of 1D resonances of order N or lower tends asymptotically to $3N^2/\pi^2$

Farey sequence code

```
def Farey(n):
    """Return the nth Farey sequence, ascending."""
    seq=[[0,1]]
    a, b, c, d = 0, 1,  1  , n
    while c <= n :
        k = int((n + b)/d)
        a, b, c, d = c, d, k*c - a, k*d - b
        seq.append([a,b])
    return seq
```

Resonance diagram



Resonance diagram & Farey

From [21]:

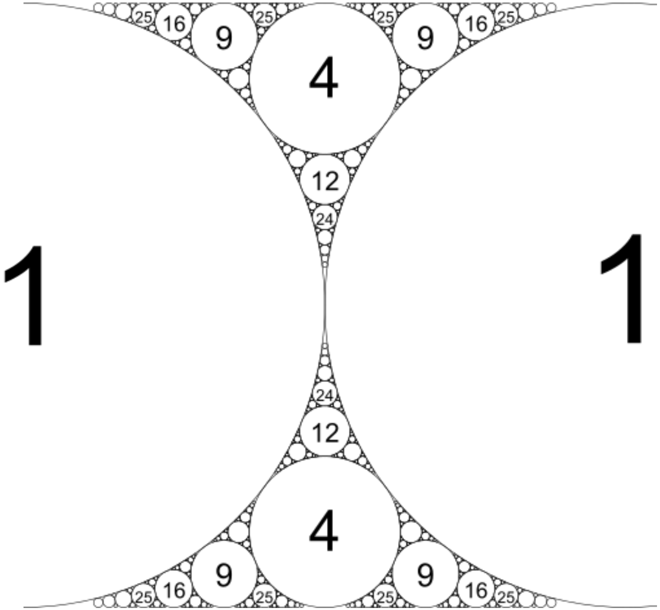
- The lines going trough $Q_x = \frac{h}{k}$, $Q_y = 0$ relate to the elements in F_N below $\frac{1}{k}$.
- The number of resonance lines in the 2D diagram is

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

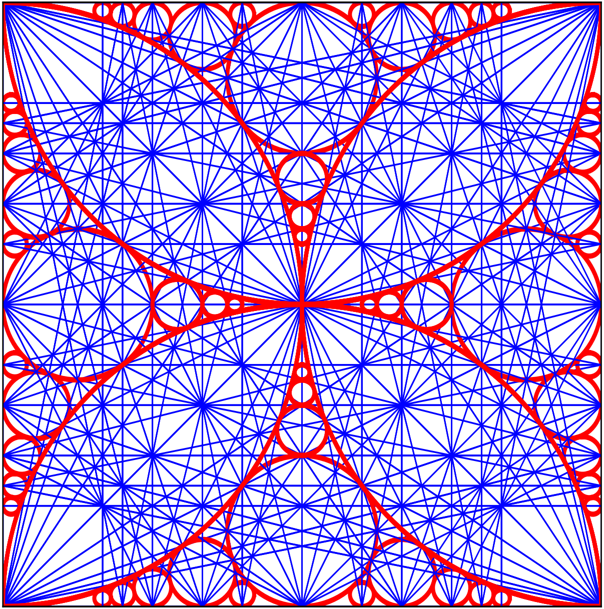
Plotting the resonance diagram

```python
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure()
ax = plt.axes()
plt.ylim((0,1))
plt.xlim((0,1))
x = np.linspace(0, 1, 1000)
FN = Farey(5)   # Farey function defined 3 slides ago
for f in FN:
    h , k = f      # Node h/k on the axes
    for sf in  FN:
        p , q = sf
        c=float(p*h)
        a=float(k*p)    # Resonance line  a Qx + b Qy = c linked to p/q
        b=float(q-k*p)
        if a>0:
            plt.plot(x, c/a - x*b/a, color='blue')
            plt.plot(x, c/a + x*b/a, color='blue')
            plt.plot(c/a - x*b/a, x, color='blue')
            plt.plot(c/a + x*b/a, x, color='blue')
            plt.plot(c/a - x*b/a, 1-x, color='blue')
            plt.plot(c/a + x*b/a, 1-x, color='blue')
        if q==k and p==1:   # FN elements below 1/k
            break
plt.show()
```

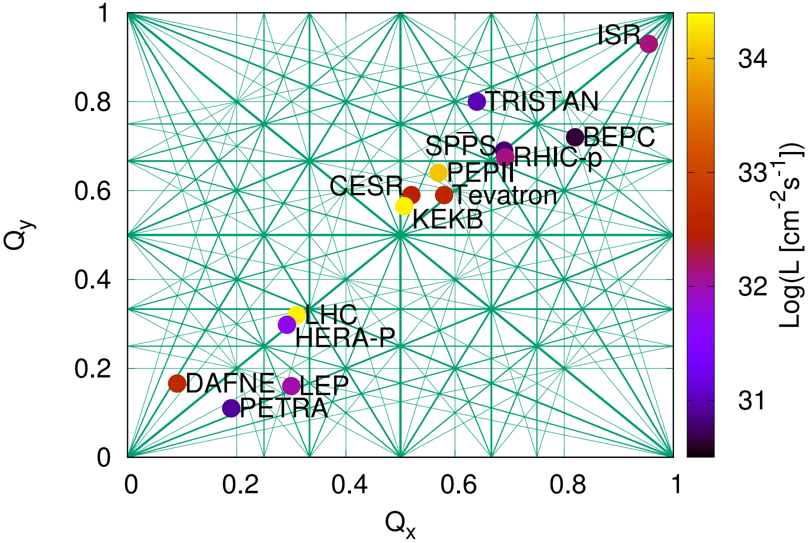
Apollonian gasket 0,0,1,1



Resonance diagram & Apollonian gasket



Colliders in the resonance diagram



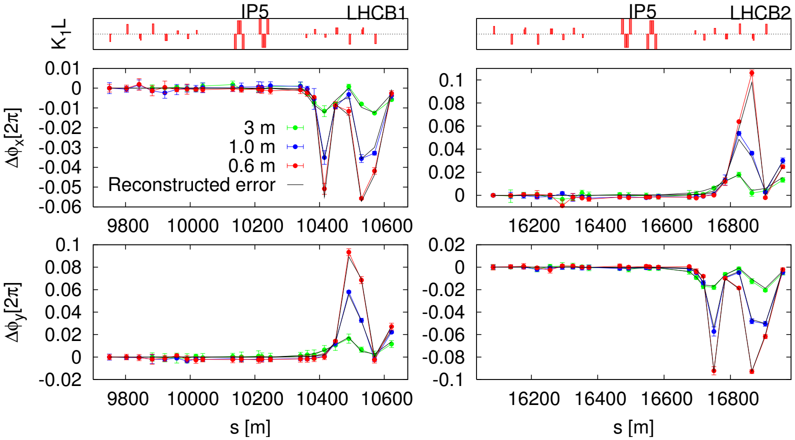
Correction

★ Local corrections
- Ideal correction: Error source identification and repair.
- Effective local error correction.
- Best few correctors (no guarantee of locality).

★ Global corrections
- Pre-designed knobs for varying particular observables in the least invasive way (like tunes, coupling, β^*, etc.)
- Best N correctors
- Response matrix approach

★ Blind corrections (optimizing, scanning, etc.)

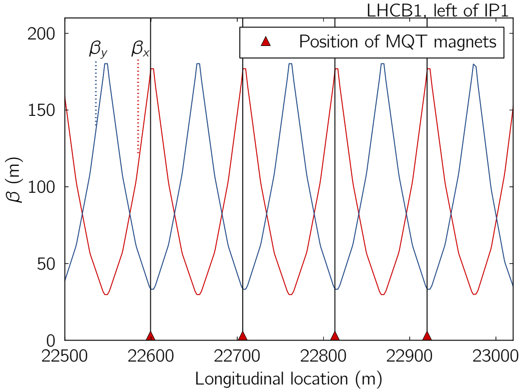
Local correction: segment-by-segment



Key point: Isolate a segment of the machine by imposing boundary conditions from measurements and find corrections [13].

Pre-designed knobs - Tunes

- In most machines it is OK to use all focusing quads to change Q_x and all defocusing quads for Q_y: PSB, PS, SPS
- In the LHC dedicated tune correctors (MQT) are properly placed to minimize impact on β-beating:



Pre-designed knobs - Coupling

- The full control of the difference resonance (f_{1001}) needs two independent families of skew quadrupoles.
- PSB, PS and SPS can survive only with one family since $\mathrm{int}(Q_x) = \mathrm{int}(Q_y)$, making errors in phase with correctors.
- In LHC there are two families to vary the real and imaginary parts of f_{1001} independently.

Response matrix approach

- Available correctors: $\vec{c}$
- Available observables: $\vec{a}$
- Assume for small changes of correctors linear approximation is good:

$$R\Delta\vec{c} = \Delta\vec{a}$$

- Use, e.g., MADX to compute R
- Invert or pseudo-invert R to compute an effective global correction based on measured $\Delta\vec{a}$:

$$\Delta\vec{c} = R^{-1}\Delta\vec{a}$$

- This works for orbit, $\Delta\beta/\beta$, coupling, etc.

Pseudo-inverse via SVD

$$R = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \\ 0 & 0 & 0 \end{pmatrix} V^T$$

Imagine $\sigma_3 \ll \sigma_2 \leq \sigma_1$, then just neglect σ_3:

$$R^{-1} = V \begin{pmatrix} \frac{1}{\sigma_1} & 0 & 0 \\ 0 & \frac{1}{\sigma_2} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} U^T$$

Correcting optics and coupling

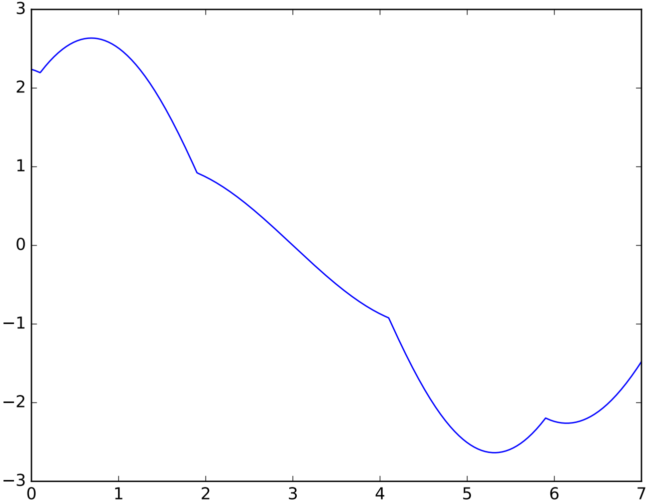
$$\begin{pmatrix} \Delta\vec{\phi}_x \\ \Delta\vec{\phi}_y \\ \frac{\Delta\vec{\beta}_x}{\beta_x} \\ \frac{\Delta\vec{\beta}_y}{\beta_y} \\ \Delta\vec{D}_x \\ \Delta\vec{Q} \end{pmatrix}_{\mathrm{meas}} = \mathbf{P}_{(\mathrm{theo})} \cdot \Delta\vec{k}$$

$$\begin{pmatrix} \vec{f}_{1001} \\ \vec{f}_{1010} \\ \vec{D}_y \end{pmatrix}_{\mathrm{meas}} = \mathbf{T}_{(\mathrm{theo})} \cdot \Delta\vec{k}_s$$

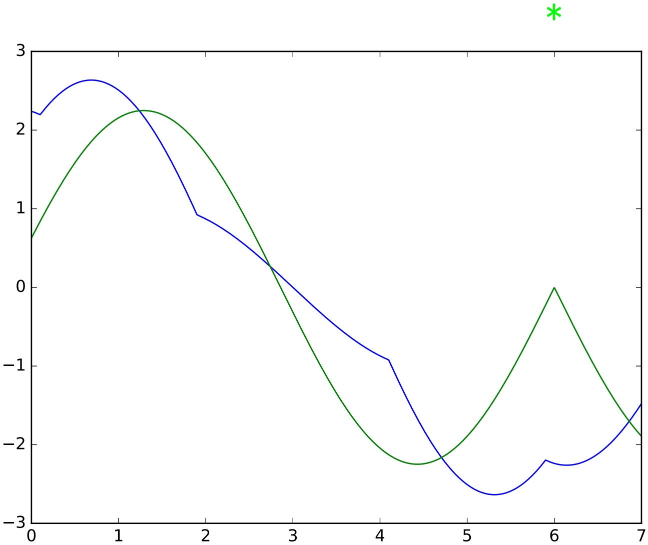
Best N corrector: exact solution

```python
import numpy as np
from scipy.optimize import least_squares
from itertools import product
import matplotlib.pyplot as plt

N_corrs=7
s=np.linspace(0, N_corrs, 1000)   # 1000 observation points

def corrs(x,i):   # Assume correctors at i=integer < N_corrs
    return np.sin(np.abs(x-i))

def model(x, c):  # Orbit at x from corrector strengths as c
    if len(x)==1:
        return sum(c*corrs(x,np.arange(N_corrs)))
    return [model([y],c) for y in x ]

def measured_orbit(x):  # Target Orbit
    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))

def f(c):      #Figure of merit for given corrector choice encoded in mask
    return  model(s, c*mask) - measured_orbit(s)

best=1e16*np.ones(N_corrs+1) ; bestmask=np.zeros([N_corrs+1,N_corrs])
for mask in product([0,1],repeat=N_corrs):  # Try all corrector combinations
    res = least_squares(f, x0=np.ones(N_corrs))  #Orbit correction
    if res.cost < best[sum(mask)]:
        bestmask[sum(mask)]=mask*res.x ; best[sum(mask)]=res.cost

plt.plot(s, measured_orbit(s))
plt.plot(s, model(s, bestmask[1]))   #Best 1 corrector
plt.plot(s, model(s, bestmask[2]))   #Best 2 correctors
```

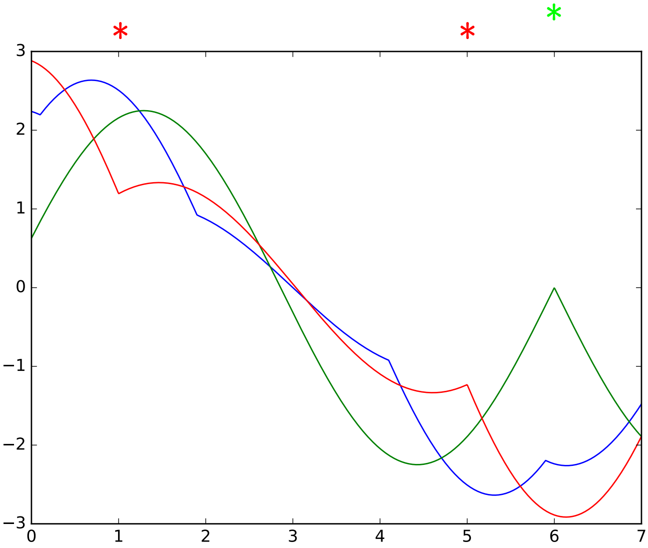
Measured orbit



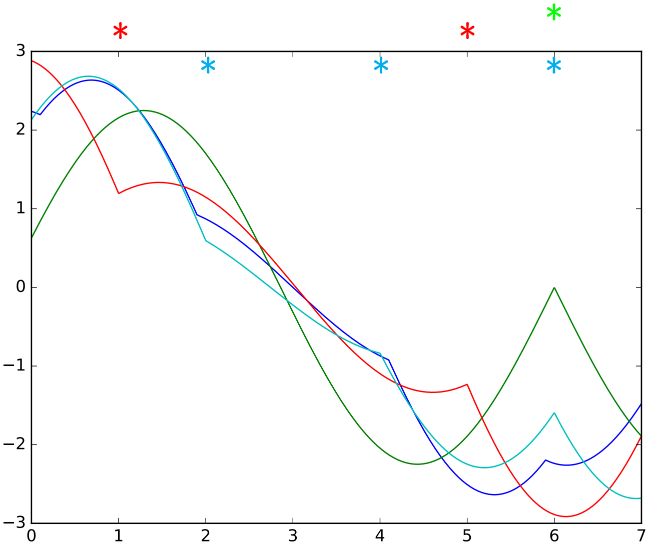
Best 1 corrector



Best 2 correctors



Best 3 correctors



Best N corrector: approximation

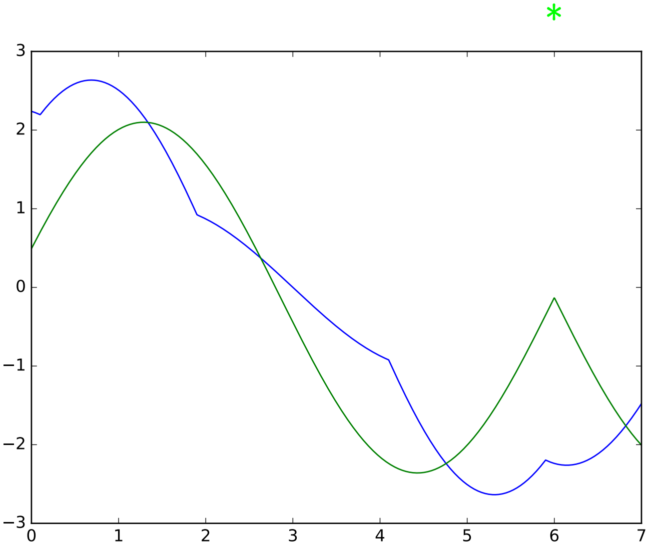
How many combinations of 500 correctors taking 20 at a time exist?
Various algorithms to find approximations:

- ★ MICADO, CERN ISR-MA/73-17, 1973
- ★ Projection pursuit regression, *J. Amer. Statist. Asso.* **76** 1981
- ★ Matching pursuit, *Transactions on signal processing* **41**, No 12, 1993.
  Later improved and named Orthogonal Matching pursuit (OMP)

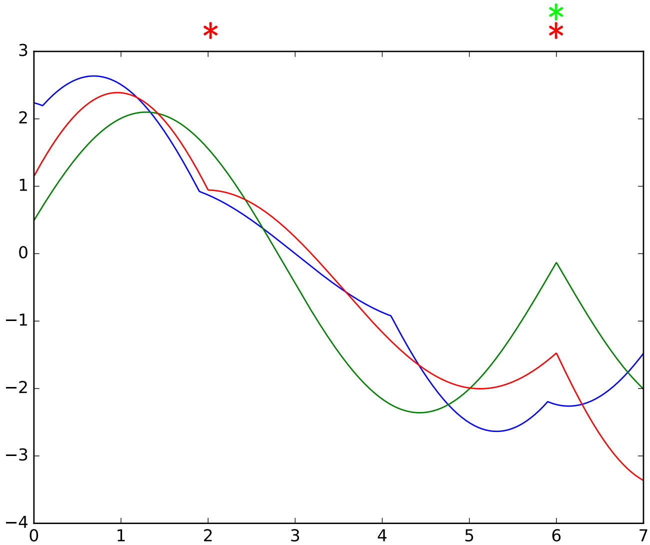
Best N corrector: OMP (= MICADO)

```python
from sklearn.linear_model import OrthogonalMatchingPursuit
import numpy as np
import matplotlib.pyplot as plt

N_corrs=7
N_BPMs=1000
s=np.linspace(0, N_corrs, N_BPMs)  # 1000 BPMs

def corrs(x,i):
    return np.sin(np.abs(x-i))

def measured_orbit(x):
    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))

def measured_orbit(x):
   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))
#################   New part for OMP ##################
X=[]
for  i in range(N_BPMs): # Prepare response matrix for OPM
    X.append(corrs(s[i],np.arange(N_corrs)))
y= measured_orbit(s)
reg = OrthogonalMatchingPursuit(n_nonzero_coefs=1).fit(X, y)  #Run OMP for best 1 corr.
print reg.coef_  # coefficient of best 1 corr
plt.plot(s, reg.predict(X))
```

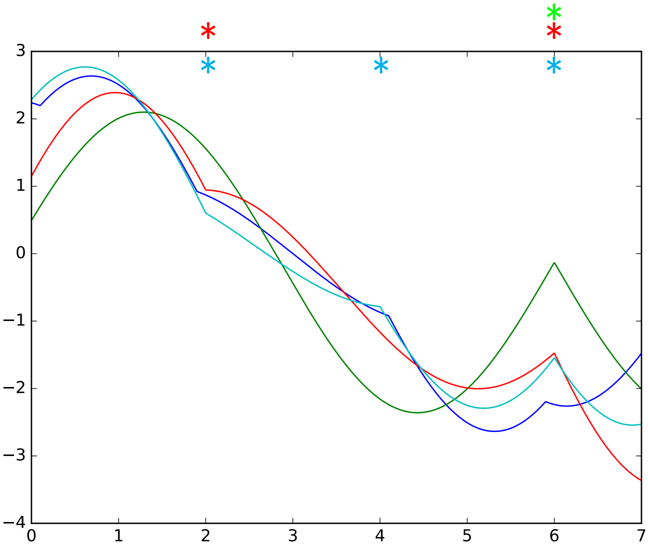
OMP: Best 1 corrector



OMP: Best 2 correctors

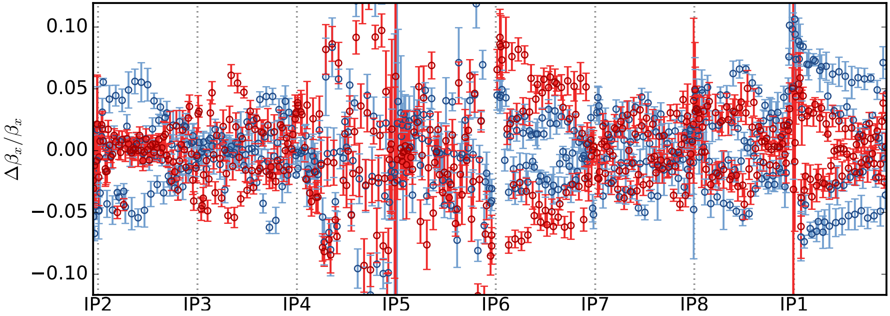


OMP: Best 3 correctors



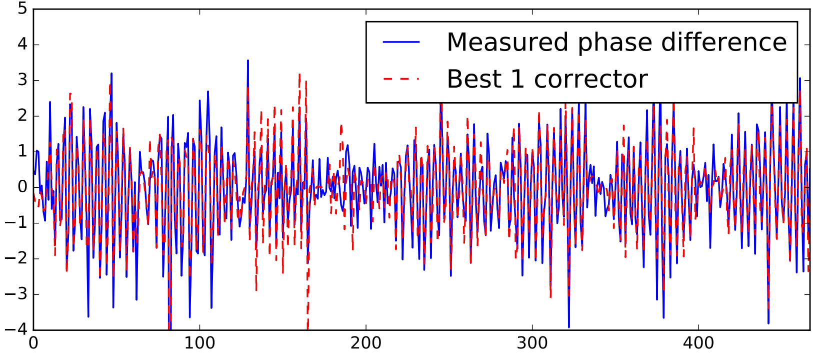
Real life example: Optics drift in 4 months

LHC beam 1, $\beta^* = 60/15$ cm, 6.5 TeV:



What changed? A single quadrupole?

Possible candidate found with OMP



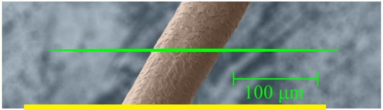
The identified quadrupole is currently under investigation.

Analytical equations

$$\delta \frac{D_{z,j}}{\sqrt{\beta_{z,j}}} = \sum_m \left[\left(\pm\delta K_{0,m} + \delta J_{1,m} D_{y,m} \mp \delta K_{1,m} D_{z,m} \right) \frac{\sqrt{\beta_{z,m}}}{2} \frac{\cos(\tau_{z,mj})}{\sin(\pi Q_z)} \right.$$
$$\left. + \frac{D_{z,j}}{\sqrt{\beta_{z,j}}} \delta K_{1,m} \frac{\beta_{z,m}}{4} \frac{\cos(2\tau_{z,mj})}{2\sin(\pi Q_z)} \right]$$

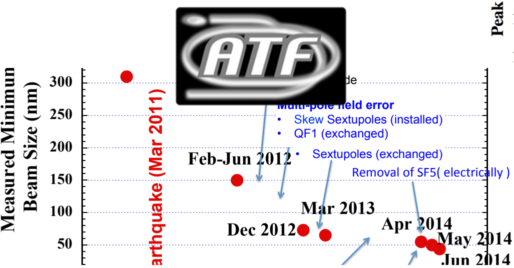
$$\delta \Phi_{z,wj} = \pm \sum_m \delta K_{1,m} \frac{\beta_{z,m}}{4} \left[2\left(\Pi_{mj} - \Pi_{mw} + \Pi_{jw} \right) + \frac{\sin(2\tau_{z,mj}) - \sin(2\tau_{z,mw})}{\sin(2\pi Q_z)} \right]$$

Three world records with optimizers



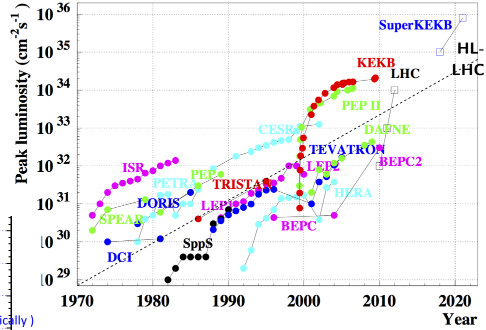
$\epsilon_y = 0.9 \pm 0.4$ pm

via random walk optimization



$\sigma_y = 44 \pm 3$ nm

via scanning orthogonal knobs



$L = 2.1 \times 10^{34}\ \mathrm{cm}^{-2}\mathrm{s}^{-1}$

Luminosity optimized via downhill Simplex

Dynamic linear imperfections

- Ground motion and vibrations in quadrupoles produce sinusoidal dipolar fields
- Electrical noise can cause currents in quadrupoles and dipoles to oscillate in time
- Electromagnetic pollution can act directly on the beam.
- Slow variations ($f << Q_{x,y} f_{rev}$) just cause a time varying orbit and optics
- Fast variations ($f \approx Q_{x,y} f_{rev}$) can cause resonances and emittance growth

An oscillating dipolar field

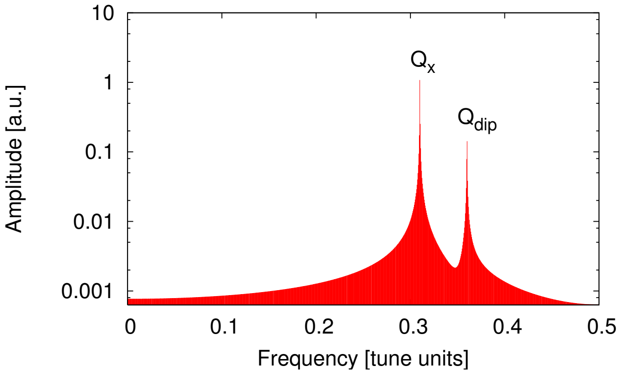
- Let $Q_{dip} = f_{dip}/f_{rev}$ be the tune of the dipolar field oscillation.
- This causes the appearance of new resonances
- Linear resonances: $Q_x \pm Q_{dip} = N$
- Non-linear resonances of sextupolar order:

$$Q_x \pm 2Q_{dip} = N$$

$$2Q_x \pm Q_{dip} = N$$

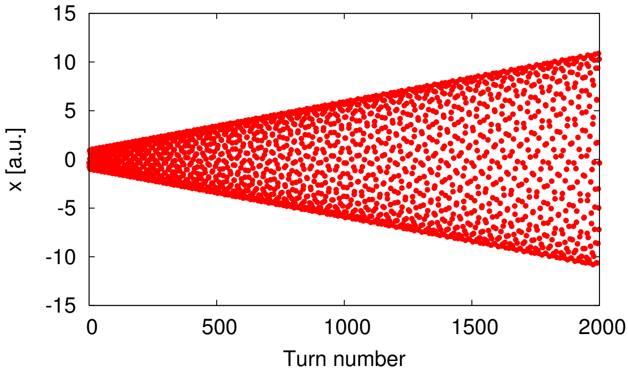
- Note that $mQ_{dip} = N$ is not a problem

Oscillating dipolar field, $Q_x \neq Q_{dip}$



Orbit oscillates with Q_{dip} but there is no emittance growth far from resonances.

Oscillating dipolar field, $Q_x = Q_{dip}$

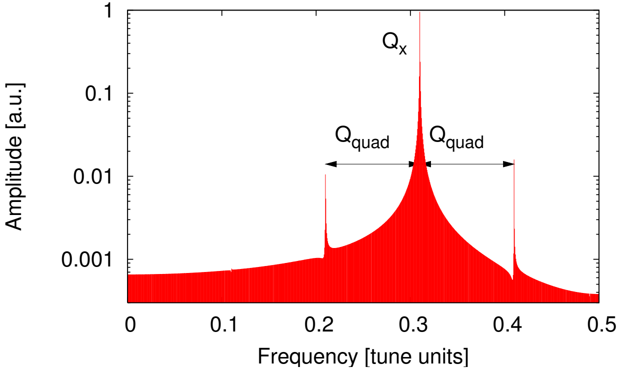


Linear growth in time $\rightarrow$ Emittance growth.

An oscillating quadrupolar field

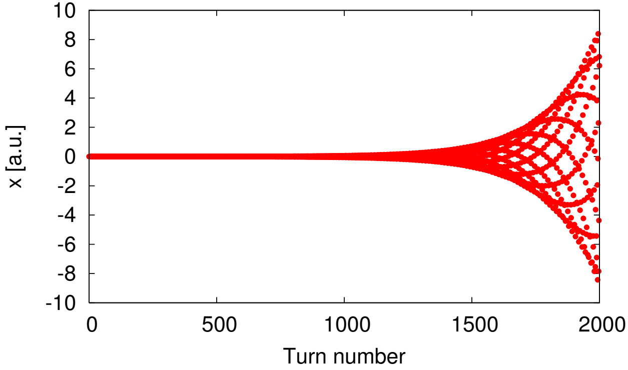
- Let $Q_{quad} = f_{quad}/f_{rev}$ be the tune of the quadrupolar field oscillation.
- This causes the appearance of new resonances
- Linear resonances: $2Q_x \pm Q_{quad} = N$

Oscillating quadrupolar field, $2Q_x \neq Q_{quad}$



Tune is modulated with Q_{quad}, displaying sidebands at $Q_x \pm Q_{quad}$ but there is no emittance growth far from resonances.

Oscillating quadrupolar field, $2Q_x = Q_{quad}$

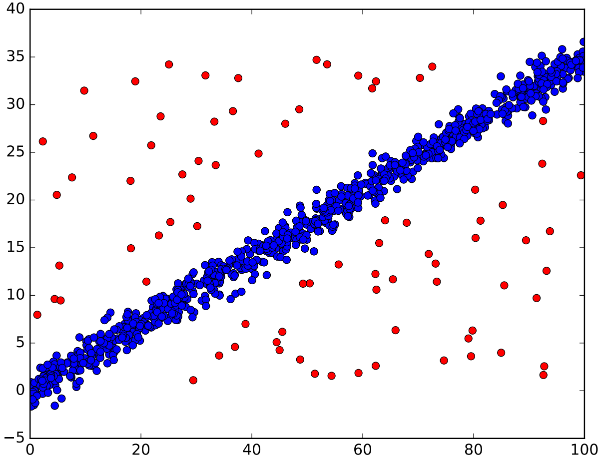


Exponential growth, clear signatures depending on the oscillating field type.

Outlayer detection: Linear correlation

```python
import numpy as np
from scipy.stats import t
import matplotlib.pyplot as plt

def get_filter_mask(data, x_data=None, limit=0.0, niter=20):
    """Assumes linear correlation, normal distr. Returns a filter mask for the original array"""
    mask = np.ones(len(data), dtype=bool)
    nsigmas = t.ppf([1 - 0.5 / len(data)], len(data))
    prevlen = np.sum(mask) + 1
    for _ in range(niter):
        if not ((np.sum(mask) < prevlen) and (np.sum(mask) > 2)):
            break
        prevlen = np.sum(mask)
        if x_data is not None:
            m, b = np.polyfit(x_data[mask], data[mask], 1)
            y, y_orig = data[mask] - b - m * x_data[mask], data - b - m * x_data
        else:
            y, y_orig = data[mask], data[:]
        mask = np.abs(y_orig - np.mean(y)) < np.max([limit, nsigmas * np.std(y)])
    return mask

if __name__ == '__main__':
    x_data = 100 * np.random.rand(1000);
    y_data = 0.35 * x_data + np.random.randn(1000); y_data[-100:] = y_data[99::-1]
    mask = get_filter_mask(y_data, x_data=x_data)
    f, ax = plt.subplots(1)
    ax.plot(x_data, y_data, 'ro')
    ax.plot(x_data[mask], y_data[mask], 'bo')
    plt.show()
```

Outlayer detection: Assuming linear correlation and normal distribution



Bibliography

[1] E.D. Courant and H.S. Snyder, Annals of Physics **3** (1958).
[2] R. Tomás et al, Phys. Rev. Accel. Beams **20** 054801 (2017)
[3] R. Miyamoto, PhD thesis, Uni. of Texas at Austin (2008).
[4] N. Biancacci et al., Phys. Rev. Accel. Beams **19**, 054001 (2016).
[5] A. Franchi, arXiv:1603.00281 (2016).
[6] A. Hofmann and B. Zotter, Issued by: ISR-TH-AH-BZ-amb, Run: 640-641-642 (1975).
[7] F. Carlier et al., Phys. Rev. Accel. and Beams, 2016.
[8] F. Schmidt and R. Bartolini, LHC Project Report 132 (1997).
[9] R. Tomás et al., Phys. Rev. ST Accel. Beams **8**, issue 2, 024001.
[10] M. Minty and F. Zimmermann, *Measurement and Control of Charged Particle Beams*, Springer, Berlin (2003).
[11] Y. Alexahin et al., Journal of Instrumentation **6**, P10006 (2011).
[12] T. H. B. Persson et al., Phys. Rev. ST Accel. Beams **17**, 051004.
[13] R. Tomás et al., Phys. Rev. ST Accel. Beams **15**, 091001 (2012).
[14] R. Tomás et al., CERN-ACC-NOTE-2018-0025.

Bibliography

[15] P. Castro, Thesis CERN-SL-96-070-BI
[16] A. Langner et al, Phys. Rev. ST Accel. Beams **18**, 031002 (2015)
[17] A. Wegscheider et al, Phys. Rev. Accel. Beams **20**, 111002 (2017)
[18] R. Tomás, Phys. Rev. ST Accel Beams **5** 54001 (2002)
[19] S. White et al., Phys. Rev. ST Accel. Beams **16** 071002 (2013)
[20] F. Carlier et al., doi:10.18429/JACoW-IPAC2018-MOPMF033
[21] R. Tomás, Phys. Rev. ST Accel. Beams **17** 014001 (2014) & arXiv:1406.6991v2