CERN-ACC-NOTE-2019-0046

24 October 2019
foteini.asvesta@cern.ch

# Resonance Driving Terms From Space Charge Potential

F. Asvesta[1,2], H. Bartosik[1]
[1]CERN, Switzerland, [2]NTUA, Greece

Keywords: Resonance Driving Terms, Hamiltonian formalism, space charge potential, Gaussian distributions, space charge tune spread, analytical tools

Summary

This note presents the calculation of the Resonance Driving Terms (RDTs) and the tune spread coming from the space charge potential of a Gaussian beam. These can be obtained by including this potential in the linear Hamiltonian of the accelerator. Through the RDTs, incoherent resonances excited by the space charge potential can be identified. The analytical method has been implemented in a self contained Python [1] module, PySCRDT [2], in order to generalize the derivation and apply it to various rings and beams. The analytical approach as well as numerical examples and applications of PySCRDT are presented.

# Contents

# 1 Introduction

The Coulomb forces between the particles of a moving bunch are referred to as the space charge effect. Space charge is dominant in low energy and high-brightness machines [3], such as the CERN injectors. The main concern for a ring dominated by space charge is the large tune spread it introduces, since the particles of the bunch may overlap resonances far from the set tunes. In addition, the variation of the tunes through the synchrotron motion due to the longitudinal dependency of space charge and chromatic effects leads to periodic resonance crossing. If a resonance is driven by lattice errors the above can lead to losses and/or emittance blow-up [4]. However, under certain conditions space charge can directly excite resonances [5, 6]. In leading order, the resonances excited from space charge are of even order and their harmonic coincides with a multiple of the machine periodicity [7, 8].

The space charge effect can be studied using numerical self-consistent models (Poisson solvers [9]) or analytical models in which it is regarded as a constant (frozen) field, using for example the Bassetti-Erskine formula [10]. The self-consistent approach is essential for the study of the coherent effects while the latter is sufficient when studying the incoherent effects [11]. In this respect, throughout this note the space charge is viewed as a frozen potential of a Gaussian bunch [12].

To define the resonances driven by the space charge potential and obtain the space charge tune spread, perturbation theory is applied on the Hamiltonian describing the motion in a ring [13]. To generalize the calculation of the RDT and detuning terms the PySCRDT module was developed and can be applied to any ring. The estimation is done analytically using the SymPy [14] library in Python [1]. The optics needed for the computation are taken from MAD-X [15] (i.e. TWISS table). The mathematical derivation is presented and examples and applications for the PySCRDT are given.

# 2 Resonance Driving Terms

## 2.1 Linear Hamiltonian

To study the space charge effect on the particles of a Gaussian beam the starting point is the linear Hamiltonian describing the particle motion [16]:

$$H = \frac{1}{2}(K_x x^2 + K_y y^2 + p_x^2 + p_y^2) \tag{1}$$

where, $K_{x,y}$ are the magnetic strengths.

Using the Floquet transformation in which:

$$u = \sqrt{2\beta_u J_u}\cos\psi_u \tag{2}$$

where $u$ the $x$ or $y$ coordinate, $\beta_u$ the corresponding Twiss beta function, $J_u$ the action and $\psi_u$ the angle (betatron phase), the Hamiltonian can be expressed in action angle variables as:

$$\tilde{H} = \frac{1}{R}(Q_x J_x + Q_y J_y) \tag{3}$$

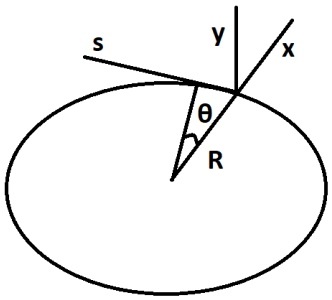


Figure 1: Frenet-Serret Coordinate system.

where R is the ring radius and $Q_{x,y}$ the corresponding tune.
In the above Hamiltonian there is a dependency on the $s$ Frenet-Serret coordinate, shown in Fig. 1. Rescaling to get the dependency on the orbital angle $\theta = 2\pi s/C$, where C is the ring circumference, the linear Hamiltonian is written as:

$$\bar{H} = R\tilde{H} = Q_x J_x + Q_y J_y \tag{4}$$

## 2.2 Space Charge Potential

The space charge potential of a transverse bi-Gaussian charge distribution is given as [12]:

$$V_{sc}(x,y) = -\frac{K_{sc}}{2}\int_0^\infty \frac{1 - e^{-\frac{x^2}{2\sigma_x^2+t}\frac{y^2}{2\sigma_y^2+t}}}{\sqrt{(2\sigma_x^2+t)(2\sigma_y^2+t)}}dt \tag{5}$$

where $\sigma_{x,y}$, are the respective beam sizes and $K_{sc}$ the space charge perveance:

$$K_{sc} = \frac{2r_0 N_B}{\beta^2\gamma^3\sqrt{2\pi}\sigma_s} \tag{6}$$

where: $r_0$ is the classical particle radius, $N_B$ the bunch intensity, $\beta$ and $\gamma$ the relativistic factors and $\sigma_s$ is the longitudinal beam size. Changing the longitudinal line density, i.e. the factor $\lambda = \frac{N_B}{\sqrt{2\pi}\sigma_s}$, non-Gaussian longitudinal distributions can be considered.

The integrant of Eq. (5) can be expanded, in both $x$ and $y$, using Taylor series under the paraxial approximation and the integral is evaluated analytically. The potential yields infinite even orders in $x$ and $y$. Up to $4^{\text{th}}$ order it reads:

$$\begin{aligned} V_{sc} = -\frac{K_{sc}}{2}\Bigg[&\left(\frac{x^2}{\sigma_x(\sigma_x+\sigma_y)} + \frac{y^2}{\sigma_y(\sigma_x+\sigma_y)}\right) - \\ &\left(\frac{(2\sigma_x+\sigma_y)x^4}{12\sigma_x^3(\sigma_x+\sigma_y)^2} + \frac{(2\sigma_x+\sigma_y)y^4}{12\sigma_y^3(\sigma_x+\sigma_y)^2} + \frac{x^2y^2}{2\sigma_x\sigma_y(\sigma_x+\sigma_y)^2}\right)\Bigg] \end{aligned} \tag{7}$$

## 2.3 Perturbed Hamiltonian

The space charge potential can be expressed in action angle variables with an extra dependency on the orbital angle. In this form, noted as $\bar{V}_{sc}$, it can be included in the Hamiltonian of Eq. (4) to obtain the perturbed Hamiltonian as:

$$\bar{H} = Q_x J_x + Q_y J_y + \bar{V}_{sc} \tag{8}$$

Since the space charge potential is a summation of infinite terms, the term corresponding to the resonance under study can be considered individually.

In terms of notation, resonances can be described using 5 indices as $\{h, i, j, k, l\}$ [13]. In this description the dependency on the transverse position and thus the potential is of order $(h + i)$ in $x$ and $(j + k)$ in $y$. The dependency on the tunes is $(h - i)$ in $\mathrm{Q_x}$ and $(j - k)$ in $\mathrm{Q_y}$, while '$l$' indicates the harmonic of the resonance. Therefore the resonance condition is: $(h - i)Q_x + (j - k)Q_y = l$. In the simplest case, where $i = k = 0$, the resonance can be described using only 3 indices $\{m, n, l\}$ where the dependency on both position and tune is of order $\{m\}$ in $x$ and $\{n\}$ in $y$ and the resonance condition becomes: $mQ_x + nQ_y = l$.

Applying the Floquet transformation of Eq. (2) on the potential corresponding to the resonance chosen yields:

$$\begin{aligned} &V_{sc}^{h,i,j,k}(x, y) = \mathcal{O}(x^{h+i} \cdot y^{j+k}) \xrightarrow{Floquet\ Transformation} \\ &\bar{V}_{sc}^{h,i,j,k}(J_x, J_y, \psi_x, \psi_y; \theta) = \mathcal{O}\Bigg( (2\beta_x J_x)^{\frac{h+i}{2}} (2\beta_y J_y)^{\frac{j+k}{2}} \cdot \cos\Big((h - i)\psi_x + (j - k)\psi_y\Big)\Bigg) = \\ &\tilde{V}_{sc}^{h,i,j,k} \cdot \Big[ J_x^{\frac{h+i}{2}} J_y^{\frac{j+k}{2}} \cdot \cos\big((h - i)\psi_x + (j - k)\psi_y\big)\Big] \end{aligned} \tag{9}$$

where the angles $\psi_{x,y}$ depend on the orbital angle and the respective tune and phase advance. To better understand the above expression, the $4^{\mathrm{th}}$ order resonance 4,0,0,0 can be used as an example. The order of the potential driving this resonance can be selected from Eq. (7) as:

$$V_{sc}^{4,0,0,0}(x, y) = \mathcal{O}(x^{4+0} \cdot y^{0+0}) = \mathcal{O}(x^4) = \frac{K_{sc}}{2} \frac{(2\sigma_x + \sigma_y)}{12\sigma_x^3(\sigma_x + \sigma_y)^2} \cdot x^4 \tag{10}$$

The selected part of the potential is then Floquet transformed as shown in Eq. (2) and yields:

$$V_{sc}^{4,0,0,0}(J_x, J_y, \psi_x, \psi_y; \theta) = \frac{K_{sc}}{2} \frac{(2\sigma_x + \sigma_y)}{12\sigma_x^3(\sigma_x + \sigma_y)^2} \cdot (2\beta_x J_x)^2 \cdot \frac{1}{8}(4\cos 2\psi_x + \cos 4\psi_x + 3) \tag{11}$$

However, the part relevant to the resonance 4,0,0,0 will be:

$$\begin{aligned} \bar{V}_{sc}^{4,0,0,0}(J_x, J_y, \psi_x, \psi_y; \theta) &= \mathcal{O}\Bigg( (2\beta_x J_x)^{\frac{4+0}{2}} (2\beta_y J_y)^{\frac{0+0}{2}} \cdot \cos\Big((4 - 0)\psi_x + (0 - 0)\psi_y\Big)\Bigg) = \\ &= \frac{K_{sc}}{16} \frac{(2\sigma_x + \sigma_y)}{12\sigma_x^3(\sigma_x + \sigma_y)^2} \cdot (2\beta_x J_x)^2 \cdot (\cos 4\psi_x) \end{aligned} \tag{12}$$

Finally the needed part for the RDT calculation is given as:

$$\tilde{V}_{sc}^{4,0,0,0} = \frac{K_{sc}}{48} \frac{(2\sigma_x + \sigma_y)}{\sigma_x^3(\sigma_x + \sigma_y)^2} \beta_x^2 \tag{13}$$

Since the potential is periodic, each order can be expanded in Fourier harmonics giving the RDTs of the resonances it excites conjugate to the action [13]. The Hamiltonian including only the term corresponding to the particular resonance of interest (Eq. 9) is then given by:

$$\bar{H} = Q_x J_x + Q_y J_y + G_{(h,i,j,k,l)} J_x^{\frac{h+i}{2}} J_y^{\frac{j+k}{2}} \cos\left((h-i)\phi_x + (j-k)\phi_y - l\theta + \xi_{(h,i,j,k,l)}\right) \tag{14}$$

where the resonance driving term, in amplitude and phase is given as:

$$G_{(h,i,j,k,l)} e^{j\xi_{(h,i,j,k,l)}} = \frac{1}{2\pi} \int_0^{2\pi} \tilde{V}_{sc}^{h,i,j,k} e^{j\left((h-i)\phi_x + (j-k)\phi_y + [l-(h-i)Q_x - (j-k)Q_y]\theta\right)} d\theta \tag{15}$$

In the case where $h = i$ and $j = k$ the terms of Eq. (9) are independent of the respective phases and the non linear detuning terms are obtained. In addition, off-momentum particles would have an extra dependency on the dispersion through their respective momentum deviation $\Delta p/p$. This additional dependence on $x$ leads to feed down terms, substituting $x$ as $x_{betatron} + \Delta p/p \cdot D_x$ in Eq. (9). Therefore, odd resonances in $x$ can be excited ($\{m\}$ order potential drives the $\{m-1\}$ order resonance for the off-momentum particles).

# 3 Python Implementation

The above described implementation is contained in the PySCRDT Python module [2]. The potential of Eq. (5) is expanded up to the user defined order and then evaluated analytically. Thereafter it is Floquet transformed to obtain the factor $\tilde{V}_{sc}^{h,i,j,k}$ of Eq. (9). Finally the integral of Eq. (15) is evaluated using the optics functions from MAD-X. If the RDT is requested, the result is returned in the compact form of a Python dictionary ('dict') that contains the complex RDT, its amplitude and phase. On the other hand, if the detuning is requested the result is a signed float value that yields the detuning coefficient.

## 3.1 Requirements

PySCRDT is compatible with both major versions of the Python programming language (v2.7+ and v3.6+). The dependencies include:

- SymPy [14] version>1.0 : for symbolic mathematics,
- numpy [17] version>1.11 : for vector numerical calculations,
- future [18] version>0.14 : for compatibility with Python 2

The optics functions can be taken directly from MAD-X twiss files, however, the needed functions, `S` (position in the lattice), `BETX` and `BETY` (twiss beta functions), `DX` and `DY` (dispersion functions), `MUX` and `MUY` (phase advance in $[2\pi]$) and `L` (length of each element) could be given as numpy vectors if calculated using different software.

## 3.2 Use of the code

PySCRDT is wrapped in a single class that contains all the necessary functions for the calculations as well as some accessory functions. The docstring of the initialiazation is:

---

**Signature**: PySCRDT.PySCRDT(parameters='input', order=[2,2,10], twissFile='twiss')

**Docstring**:
class for the calculation of the space charge RDT
Input : parameters : [bool|str] Parameters needed for the calculations (default=False)
if True the default values of [setParameters] are used
if str parameters are read in a file
mode : [3|5] Resonance description mode (default=None)
order : [list] Resonance order and harmonic (default=None)
twissFile : [str] MAD−X Twiss file (default=None)
**Returns**: void

---

Detailed documentation of the most useful member functions is given in Appendix A. The use of the code is illustrated through some real life applications in the following Section while more detailed examples are found in Appendix B.

# 4 Applications

To demonstrate the functionality of PySCRDT simple examples of RDT and detuning calculations are described in this Section.

## 4.1 Evaluating the RDT

The RDTs corresponding to the $10^{\text{th}}$ order structural resonances in the Super Proton Synchrotron (SPS) have been evaluated using PySCRDT. Using the files found in [2] the initialization of the class can be done as

```
import PySCRDT as rdt

# Create an instance of the class
p = rdt.PySCRDT(parameters='input.dat',order=[10,0,0,0,204],
                twissFile='Q204.tfs')
```

The RDT describing the excitation of the resonance $10Q_x = 204$ by the $10^{\text{th}}$ order potential in the SPS is obtained using:

```
# Calculate the RDT
p.getResonanceDrivingTerms()
{u'Amplitude': 1.023481761728524e+22,
u'Phase': 1.708148338141807,
u'RDT': (-1.401356834376697e+21+1.013842644012561e+22j)}
```

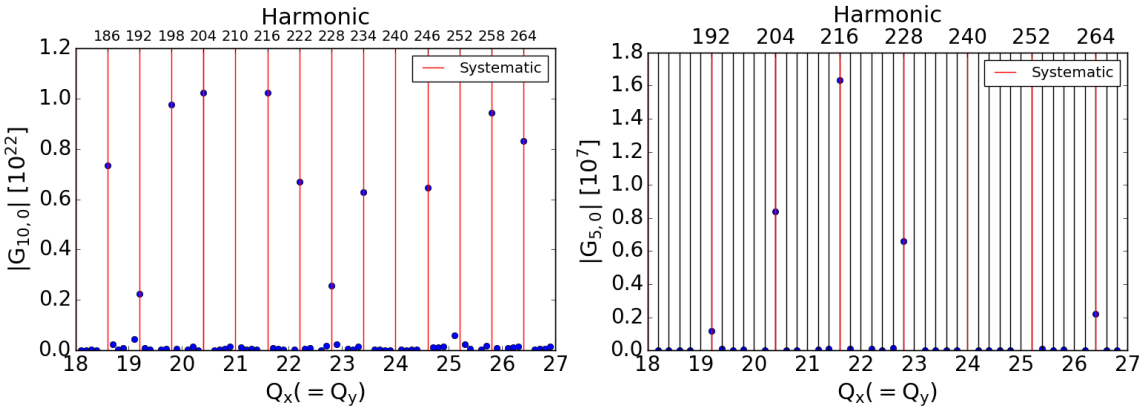


Figure 2: $10^{\text{th}}$(left) and $5^{\text{th}}$(right) order RDT calculated with PySCRDT in the SPS. The vertical lines show the harmonic of the corresponding resonances versus the tunes $Q_x = Q_y$.

Following this procedure for more harmonics the $10^{\text{th}}$ order RDT is shown in Fig. 2 (left). In addition the $5^{\text{th}}$ order excitation of the systematic resonances is shown in Fig. 2 (right). The $5^{\text{th}}$ order is the outcome of the feed down terms from the $6^{\text{th}}$ order space charge potential for off-momentum particles and can be calculated, as shown in the Example B.1. The results are in good agreement with the calculations presented in [19].

In a similar manner, the excitation of the resonance at $Q_y = 6.25$ in the Proton Synchrotron (PS) has been studied and is shown in Fig. 3. The validity of PySCRDT is demonstrated since the excitation of the harmonic 50 is seen in tracking simulations, frequency map analysis and measurements alike [8, 20].

## 4.2 Evaluating the Detuning

To calculate the maximum vertical tune shift, i.e. $dQ_y$, in the Proton Synchrotron (PS) the initialization of the PySCRDT would be

```
# Create an instance of the class
r = rdt.PySCRDT(parameters=True, order=[0,2,'any'], twissFile='twiss')
```

The detuning coefficient is then obtained:

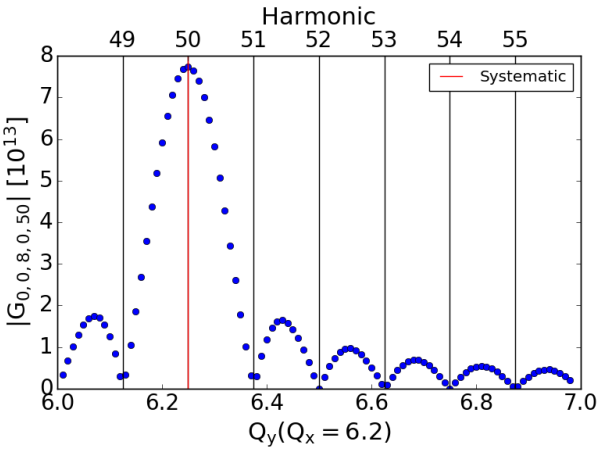


Figure 3: Variation of the amplitude of the RDT for the resonance {0,0,8,0,50} versus the vertical tune, for a fixed horizontal tune of $Q_x = 6.2$. Vertical lines show the corresponding harmonics.

---

```
# Calculate the Detuning
r.getDetuning()
-0.27642711212044718
```

---

Detuning terms up to $20^{\text{th}}$ order coming from the space charge potential have been evaluated using PySCRDT. Including these in the Hamiltonian the tunes of test particles can be calculated [16]. The resulting space charge tune spread using on-momentum particles up to 3 $\sigma$ is shown in Fig. 4. On-momentum particles up to 3 $\sigma$ are being tracked for one synchrotron period under space charge in PyORBIT [21]. The space charge is included in the simulation by the frozen solver, i.e. Bassetti Erskine formula. The tunes of the particles are calculated from the turn by turn data using PyNAFF [22] and are also plotted in Fig. 4. The comparison of the analytical evaluation using PySCRDT and the simulation data is in good agreement with the PySCRDT. Note that in the analytical evaluation only the space charge potential is taken into account while in the simulations extra detuning from the lattice non-linear elements can be seen. This is the reason for the minor differences between the analytical estimations and the simulations. Including higher potential orders the detuning of even higher amplitude particles could be evaluated.

# 5 Conclusions

The module PySCRDT is able to calculate the RDTs coming from the space charge potential of Gaussian distributions. Agreement between the code predictions for resonance excitation with simulations and measurements has been demonstrated. The non-linear detuning coming from the space charge potential is also evaluated in the presented code and can provide the expected space charge tune spread depending on the order of the potential used.

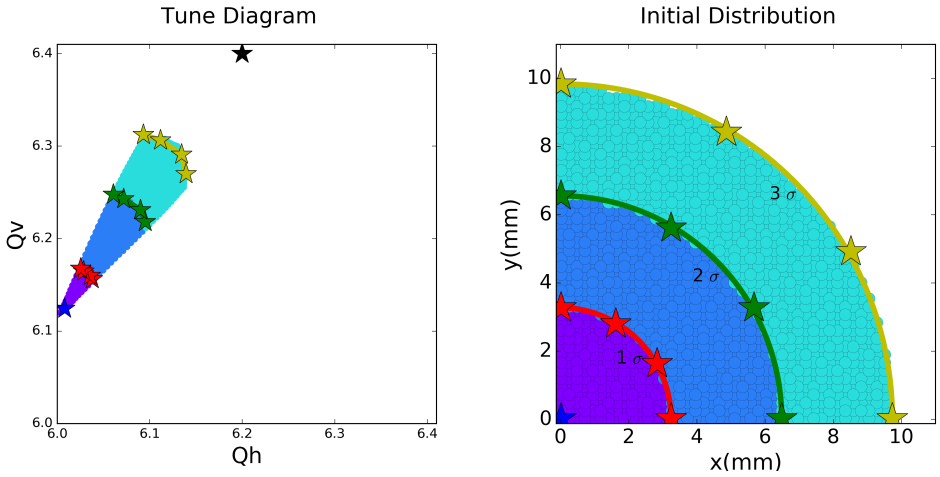


Figure 4: Space charge tune spread for particles up to $3\sigma$ (left). Initial transverse position of test particles (right). Analytical estimations, using the detuning terms up to $20^{\text{th}}$ order, are shown in stars. The different colors show the difference in beam size $\sigma$.

# 6 Acknowledgements

The authors would like to thank N. Karastathis and Y. Papaphilippou for their help and input.

# References


[1] Python Software Foundation. https://www.python.org/.

[2] F. Asvesta. https://github.com/fasvesta/PySCRDT.

[3] K. Schindl. Space charge. (CERN-PS-99-012-DI):26 p, Mar 1999.

[4] G. Franchetti, I. Hofmann, W. Fischer, and F. Zimmermann. Incoherent effect of space charge and electron cloud. *Phys. Rev. ST Accel. Beams*, 12:124401, Dec 2009.

[5] S. Machida. Space-charge-induced resonances in a synchrotron. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 384(2):316 – 321, 1997.

[6] S. Y. Lee, G. Franchetti, I. Hofmann, F. Wang, and L. Yang. Emittance growth mechanisms for space-charge dominated beams in fixed field alternating gradient and proton driver rings. *New Journal of Physics*, 8(11):291–291, nov 2006.

[7] R. Wasef, G. Arduini, H. Damerau, S. Gilardoni, S. Hancock, C. Hernalsteens, A. Huschauer, F. Schmidt, and G. Franchetti. Space Charge Effects and Limitations in the CERN Proton Synchrotron. (CERN-ACC-2013-0176):3 p, May 2013.

[8] F. Asvesta, H. Bartosik, A. Huschauer, Y. Papaphilippou, and G. Sterbini. Resonance identification studies at the CERN PS. *Journal of Physics: Conference Series*, 1067:062021, sep 2018.

[9] J. Qiang. Efficient three-dimensional poisson solvers in open rectangular conducting pipe. *Computer Physics Communications*, 203:122 – 127, 2016.

[10] M. Bassetti and G. A. Erskine. Closed Expression for the Electrical Field of a Two-dimensional Gaussian Charge. 1980. CERN-ISR-TH/80-06.

[11] F.Asvesta, H. Bartosik, A. Huschauer, Y. Papaphilippou, M. Serluca, and G. Sterbini. PS studies. Space Charge Workshop 2017, GSI, Germany, 4-6 October, 2017 https://indico.gsi.de/event/5600/.

[12] S. Kheifets. Potential of a Three-Dimensional Gaussian Bunch. Technical Report PETRA Note 119, 1976. Hand written document.

[13] G. Guignard. The general theory of all sum and difference resonances in a three-dimensional magnetic field in a synchrotron, pt 1. Technical Report CERN-ISR-MA-75-23. ISR-MA-75-23, CERN, Geneva, Apr 1975.

[14] SymPy. Python library for symbolic mathematics.

[15] L. Deniau, E. Forest, H. Grote, and F Schmidt. MAD-X, http://madx.web.cern.ch/madx/, 2016.

[16] S Y Lee. *Accelerator physics; 3rd ed.* World Scientific, Singapore, 2012.

[17] https://numpy.org/index.html. Numpy the fundamental package for scientific computing with python, 2005.

[18] https://docs.python.org/2/library/__future__.html.

[19] H. Bartosik and F. Schmidt. SPS space charge studies(from high order resonances to power converter ripple). Machine Studies Working Group meeting 2017 14, 2017. https://indico.cern.ch/event/667864.

[20] F Asvesta, H Bartosik, H Damerau, A Huschauer, Y Papaphilippou, M Serluca, G Sterbini, and P Zisopoulos. Space charge studies in the PS. *CERN Proceedings: Injector MD Days 2017*, pages 37–42. 6 p, 2017. http://dx.doi.org/10.23727/CERN-Proceedings-2017-002.37.

[21] A. Shishlo, S. Cousineau, J. Holmes, and T. Gorlov. The particle accelerator simulation code pyorbit. *Procedia Computer Science*, 51:1272 – 1281, 2015. International Conference On Computational Science, ICCS 2015.

[22] F. Asvesta, N. Karastathis, and P. Zisopoulos. https://github.com/nkarast/PyNAFF.

# A Member Functions

The class instance can be created with no inputs and the required parameters can be included in a latter state with the corresponding functions:
**Parameters**

- `setParameters` Sets the parameters needed for the calculations. They need to be defined before the MAD-X twiss file.

---

**Signature**:
PySCRDT.PySCRDT.setParameters(intensity=41e10,
bunchLength=5.96, ro=1.5347e−18, emittance_x=2e−06,
emittance_y=1.1e−06, dpp_rms=0.5e−3, dpp=0.0)

**Docstring**:
Sets the parameters for the calculation:
Input : intensity : [float] bunch intensity in ppb (Default=41e10)
bunchLength: [float] RMS bunch length in m (Default=5.96)
ro : [float] classical particle radius in m (Default=1.5347e−18 {proton})
emittance_x: [float] normalized horizontal emittance in m∗rad (Default=2e−6)
emittance_y: [float] normalized vertical emittance in m∗rad (Default=1.1e−6)
dpp_rms : [float] RMS Dp/p (Default=0.5e−3)
dpp : [float] Single particle Dp/p (Default=0)
**Returns**: void

---

- `readParameters` Reads the parameters needed for the calculations from a given text file. If not given at the creation they need to be read before the MAD-X twiss file.

---

**Signature**: PySCRDT.scrdt.readParameters(inputFile)

**Docstring**:
Reads the parameters from an input file:
Input Format: intensity = [float] # bunch intensity in ppb (Default=41e10)
bunchLength = [float] # RMS bunch length in m (Default=5.96)
o = [float] # classical particle radious in m (Default=1.5347e−18 {proton})
emittance_x = [float] # normalized horizontal emittance in m∗rad (Default=2e−6)
emittance_y = [float] # normalized vertical emittance in m∗rad (Default=1.1e−6)
dpp_rms = [float] # RMS Dp/p (Default=0.5e−3)
dpp = [float] # Single particle Dp/p (Default=0)
**Returns**: void

---

- `updateParameters` Updates any of the parameters already set or read from the input file. If the twiss file is already read the energy can also be updated through the relativistic factors 'b' or 'g'.

---

**Signature**: PySCRDT.PySCRDT.updateParameters(∗∗kwargs)
**Docstring**:
Updates the parameter dictionary
Input : any of : intensity
bunchLength
ro
emittance_x
emittance_y
dpp_rms
dpp
b
g
**Returns**: void

---

### Resonance Order

- `setOrder` Defines the Order of the resonance to be studied. The resonance mode is defined by the number of arguments given, i.e. 3 or 5 numbers. The harmonic 'l' can be set to 'any' to study an effect regardless of the harmonic for instance the 3 numbers description using 'any' could be used for the calculation of the detuning terms.

---

**Signature**: PySCRDT.PySCRDT.setOrder(args)
**Docstring**:
Sets the Resonance Orders
In "3 mode"
Input : [list] : [m,n,l]
m : [int] order in H
n : [int] order in V
l : [int|'any'] harmonic of the resonance
In "5 mode"
Input : [list] : [h,i,h,k,l]
h : [int] characteristic of H
i : [int] characteristic of H
j : [int] characteristic of V
k : [int] characteristic of V
l : [int|'any'] harmonic of the resonance
**Returns**: void

---

### Optics Functions

- `prepareData` Reads the optics functions, the tunes and the energy from the MAD-X Twiss file, and calculates the beam sizes and the space charge perveance.

---

**Signature**: PySCRDT.PySCRDT.prepareData(twissFile)
**Docstring**:
Prepares the data from a MADX Twiss file including at least {s, betx, bety, dx, dy, mux, muy, l}
Input : twissFile : [str] twiss file (default=None)
**Returns**: Void

---

The **driving terms** and the **detuning** can be calculated once the above are set using:

- `getResonanceDrivingTerms` Returns the driving terms calculated for the given parameters, optics and order, in a dictionary including 'RDT', 'Amplitude', 'Phase'

---

**Signature**: PySCRDT.PySCRDT.getResonanceDrivingTerms(feedDown)
**Docstring**:
**Returns** the RDTs
**Inputs**: feedDown: [bool] If True feed−down effects due to dispersion are included
**Returns**: [dict] the resonance driving terms

---

- `getDetuning` Returns the detuning terms calculated for the given parameters, optics and order.

---

**Signature**: PySCRDT.PySCRDT.getDetuning()
**Docstring**:
**Returns** the detuning for the order requested
**Returns**: [float] the detuning

---

After **updating parameters** the RDTs/Detuning need to be recalculated before the get functions described above using:

- `resonanceDrivingTerms`

---

**Signature**: PySCRDT.PySCRDT.resonanceDrivingTerms(feedDown)
**Docstring**:
Calculates the resonance driving terms for the resonance requested
**Inputs**: feedDown: [bool] If True feed−down effects due to dispersion are studied
**Returns**: Void

---

- `detuning`

---

**Signature**: PySCRDT.PySCRDT.detuning()
**Docstring**:
Calculates the non linear detuning terms
**Returns**: [float] the detuning

---

After **updating the potential order** the potential needs to be recalculated before recalculating the RDT/Detuning (as above) using:

- `potential`

---

**Signature**: PySCRDT.PySCRDT.potential(feedDown=False)
**Docstring**:
Calculates the space charge potential for the given resonance order
**Inputs** : feedDown : [bool] needed when single particle Dp/p non 0 (default=False)
**Returns**: Void

---

# B Examples

## B.1 Evaluating the RDT — feedDown — parameters from txt file

PySCRDT can be initialized using parameters read from txt files. Using the file in [2] it yields:

```
# Create an instance of the class
s = rdt.PySCRDT(parameters='input', order=[5,0,0,0,10],twissFile='twiss')
```

In this case, the RDT corresponds to the resonance $5Q_x = 10$. The odd order in $x$ is coming from the $6^{th}$ order potential when a non-zero $\Delta p/p$ is taken into account. It is obtained as:

```
# Calculate the feed down RDT
s.getResonanceDrivingTerms(feedDown=True)
{u'Amplitude': 5956005.673851572,
 u'Phase': -1.6912767402729372,
 u'RDT': (-715847.27187717578-5912830.6478621662j)}
```

## B.2 Re-evaluating the RDT — updating parameters, resonance order, twiss file

PySCRDT can update the created objects with new values for the parameters, optics or even the resonance under study. Extra functions are called to re-calculate the RDT for the updated values. Using the object created in Example 3.3.1

- update parameters

```
# Updating parameters
p.updateParameters(intensity=5e10)

# Re-evaluate RDT
p.resonanceDrivingTerms()

# Get the new RDT
p.getResonanceDrivingTerms()
{u'Amplitude': 17245457510237.131,
 u'Phase': 0.097424999499474182,
 u'RDT': (17163678470167.432+1677482074451.2266j)}
```

- load new twiss file

```
# Load Twiss file
p.prepareData('twiss_620_623')

# Re-evaluate RDT
p.resonanceDrivingTerms()

# Get the new RDT
p.getResonanceDrivingTerms()
{u'Amplitude': 6468621612582.9766,
u'Phase': 0.93229926415444897,
u'RDT': (3855229057634.7114+5194253986853.6045j)}
```

- change resonance order

```
# Set new Order
p.setOrder([0,0,4,0,25])

# Re-evaluate the potential
p.potential()

# Re-evaluate RDT
p.resonanceDrivingTerms()

# Get the new RDT
p.getResonanceDrivingTerms()
{u'Amplitude': 778.6278967569026,
 u'Phase': 0.096355696036904109,
 u'RDT': (775.01613820785781+74.909192529732366j)}
```