# Simulation of Beam and Plasma Systems
# Homework 8

D. Bruhwiler, S. M. Lund, J.-L. Vay, R. Lehe, and D. Winklehner

Wednesday, Jan 24$^{\text{th}}$, 2018

## Problem 1 - Projectional and Sectional Emittance

As discussed in class, the rms emittance

$$\epsilon_x = 4\sqrt{\langle x^2\rangle_\perp \langle x'^2\rangle_\perp - \langle xx'\rangle_\perp^2} \qquad [mm\text{-}mrad]$$

is calculated from a projection of the full 4-D trace space into the x-x' plane. Here, the second moments of $f(x, y, x', y')$ are defined as

$$\langle x^2\rangle = \frac{\iiiint x^2 f(x, y, x', y')dxdydx'dy'}{\iiiint f(x, y, x', y')dxdydx'dy'}$$

and similarly for $\langle x'^2\rangle$ and $\langle xx'\rangle$. An identical treatment holds for the y-y' emittance.
However, many systems we are interested in are cylindrically symmetric and thus it is convenient to run a simulation in a 2D mode with coordinates R and Z. This can speed up the calculation time significantly compared to 3D simulations. Unfortunately, if we just substitute r for x in the above equations for the emittance, we obtain the emittance of a slice and not the projection of the beam (both can be useful, though). It stands to reason that we would like a way to calculate both from the results of a RZ symmetric simulation.
The transformation from cartesian coordinates (x,y,x',y') into polar coordinates (r, $\theta$, r', $\alpha$') is given by

$$\begin{aligned} r^2 &= x^2 + y^2 \\ r'^2 + \alpha'^2 &= x'^2 + y'^2 \\ x' &= r'\cos\theta - \alpha'\sin\theta \\ y' &= r'\sin\theta + \alpha'\cos\theta \end{aligned}$$

where

$$x' = \frac{\mathrm{d}x}{\mathrm{d}z} = \frac{v_x}{v_z}, \quad r' = \frac{\mathrm{d}r}{\mathrm{d}z} = \frac{v_r}{v_z}, \quad \alpha' = r\frac{\mathrm{d}\theta}{\mathrm{d}z} = \frac{v_\theta}{v_z}.$$

a) In the simple case where the beam has azimuthal symmetry and no azimuthal velocity component ($\alpha' = 0$), transform the above moment equations ($\langle x^2\rangle$, $\langle x'^2\rangle$, and $\langle xx'\rangle$) into polar coordinates to obtain a formula for the projectional emittance $\epsilon_{rms}(x, x')$ that depends only on r and r'.

b) Implement both ways of calculating $\epsilon_{x,rms}$ (from x, x' and from r, r') as functions in a python script.

**Hint:** Recall $\langle \ldots \rangle = \int \ldots f dV$ and that the volume element $dV$ after the non-canonical coordinate transformation will be $rdrdr'$. So, e.g. $\iint f(r,r')rdrdr'$, which has to be the number of particles (and should always be conserved) can thus be written in python simply as `len(r)`.

c) Use the random number generator `numpy.random` to generate a round, azimuthally symmetric particle distribution with no azimuthal velocity component in polar coordinates. Transform the distribution into cartesian coordinates and confirm your calculations in a) by calculating $\epsilon_{x,rms}$ from either distribution using the appropriate function. Send your script to Daniel.

**Note:** A more rigorous treatment of the dynamics of skew beams in an R-Z simulation can be found in [Chan et al., NIM A, 1991].

## Problem 2 - Convergence

Download the file `https://people.nscl.msu.edu/~lund/uspas/sbp_2018/lec_intro/07.conv_examples/xy-quad-mag-mg-conv.py` from the course website.

The script is setup with some basic parameters and diagnostics:

- rms matched waterbag distribution with zero centroid.
- 100 steps per lattice period axial advance and a 20 lattice period advance, with transverse symmetry options on the field solver to reduce run time.
- Diagnostic setup for $x$-$y$, $x$-$x'$, $y$-$y'$ and $x'$-$y'$ snapshot distribution projection plots and beam centroid ($X$,$Y$), envelope ($r_x$,$r_y$), and emittance ($\varepsilon_x$, $\varepsilon_y$) history evolution plots.

First run the script as is and review the output cgm file.

a) Adjust nstep and repeat the run for double the initial $s$-steps per lattice period and compare results. Then repeat the run reducing nstep to reduce the $s$-steps per lattice period until simulation results start to change. Comment on the number of steps where emittance, rms envelope, and phase-space projections start to become significantly impacted. Note that ndiag and nstep must be set consistently for history diagnostic moments to be plotted at the same phase of the lattice period.

b) Change back to the original advance nstep value. Use formulas in class to estimate the Debye screening length of electrostatic interactions in the beam for the parameters given. How do the default mesh parameters compare to this length? Change the grid to poorly resolve the beam edge and comment on changes with the beam envelope radius, emittance, and phase-space projections.

c) Change back to the original parameters in mesh resolution. Drop the particle number set by `top.npamx` strongly until you see changes in beam envelope radius, emittance, and phase-space projections. Comment on which are impacted first.

d) Change back to the original parameters in resolution and and particle number and then change to a KV distribution and repeat the run. Next, make a choice of grid to have $\approx 6$ cells across the radial extent of the initially matched beam. Repeat the run and compare results. Then drop particle number strongly in addition to having poor grid resolution and repeat the run. Compare results and comment on whether one might expect more or less instability depending on numerical parameter choices.

## Problem 3 - Software Testing

Your assignment is to create one or more Python tests, which can be executed via `pytest`. You may choose one of the two tasks below.

The first task involves use of the `RsBeams` library, which you are encouraged to choose.

**Note:** Some students were unable to work with the `RsBeams` library and `pytest` from the classroom desktop computer. Therefore, you have the option to choose the second task.

When you have a working test file, email your file to bruhwiler@radiasoft.net

a) First task: create a test for the `RsBeams` library

- work in the subdirectory `tests/`
- create a Python test file, which does the following:
  - create an instance of the class `RsDistrib6D`
  - use the method `clean_phase_space_6d()` to "clean" the phase space
  - create a test that will determine whether any of the 6 rms values change in the $4^{\text{th}}$ significant figure
  - make sure it is working correctly with pytest

b) Second task: create 4 Python tests

- create a separate function for each test
- put all the functions into a single Python file
  - each test must make use of a different mathematical function
  - number your tests; the Nth test must enforce agreement to N+1 significant figures