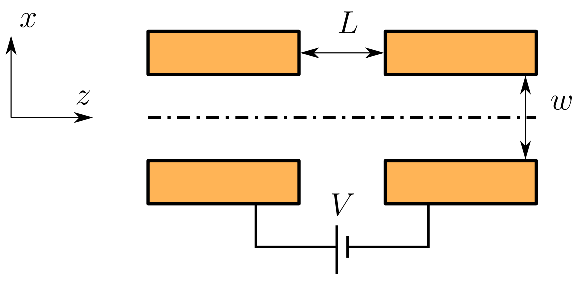
Exercise for lecture on IBSimu

LISA Specialized Training 4: Advanced Computational Techniques
7.–10. Nov 2022
Taneli Kalvas (taneli.kalvas@jyu.fi)

Let us look at a two-dimesional acceleration gap with a gap width $w = 10$ mm in the x-direction and length $L = 10$ mm in the z-direction. The system is two-dimensional, so the gap size in y-direction is infinite. The beam (source voltage 10 kV) propagates towards the positive z-direction. Acceleration $V = 10$ kV.



Using numerical modelling, find the first order transfer matrix coefficients describing the gap focusing properties. Assume that the gap element length is L and that the beam is of low intensity.

Options for solving the exercise

1. If you have a Linux computer at your disposal, install IBSimu and calculate the field and particle trajectories within the field using IBSimu. Do not start installing IBSimu and all the requirements on a Windows computer.

2. Write a Python (or whatever programming language) code for solving the Poisson equation (potential) in the geometry. There are two options for this: A) Build the FDM matrix and use a general purpose solver from scipy/numpy or B) Use the Gauss-Seidel method (see wikipedia) to solve the FDM. Fly particles through the electric field computed from the potential

3. Use some other code with the necessary capabilities: SimIon, Comsol Multiphysics and Opera VectorFields could do it all or use FEMM to calculate the field and write the particle tracker yourself.

Exercise details (tips)

1. Notice that the model used for calculating the field has to be long enough (in z) to prevent the simulation boundaries from affecting the field too much. My guess is $z_{\mathrm{min}} = -5w$, $z_{\mathrm{max}} = 5w$.

2. Run 1000 or so particles with initial coordinates covering the region of interest. Particles should start at the boundary $z = z_{\mathrm{min}}$.

3. Particles should be limited to being somewhat close to the axis to avoid nonlinear focusing effect of the gap field.

4. Almost any ODE integrator is sufficient for calculating the trajectories. Runge-Kutta 45 would be a standard choice and is available in scipy. If you are writing this in python, you also need a field interpolator, for example the RegularGridInterpolator, also from scipy.

5. With initial coordinates $(x, x')_i$ and final coordinates $(x, x')_f$ from the trajectory calculation, make a linear fit to find matrix coefficients in

$$\begin{pmatrix} x \\ x' \end{pmatrix}_f = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} x \\ x' \end{pmatrix}_i$$

6. At this stage you have calculated the matrix $M_{\mathrm{drift}} M_{\mathrm{gap}} M_{\mathrm{drift}}$. Multiply with the inverse of the drift matrix from right and left to acquire the pure M_{gap}.