Accompanying Note for Python Code

Kelvin Onggadinata

April 2018

1 Computing $\vec{E}$ field and Lorentz force

We already know that the potential due to the quadrupole follows

$$V(\vec{r}) = \frac{V_0}{2}\frac{z^2 - s^2/2}{d^2}, \tag{1}$$

where z and s is cylindrical coordinates and d is the characteristic trap dimension. Now, we calculate the $\vec{E}$ field due to this potential.

$$\vec{E} = -\nabla V = \frac{\partial V}{\partial s}\hat{s} + \frac{1}{s}\frac{\partial V}{\partial \phi}\hat{\phi} + \frac{\partial V}{\partial z}\hat{z}$$

$$\vec{E} = \frac{V_0}{2d^2}s\hat{s} - \frac{V_0}{d^2}z\hat{z} \tag{2}$$

Using change of coordinate, we substitute $\hat{s} = \cos\phi\hat{x} + \sin\phi\hat{y}$, $s\cos\phi = x$ and $s\sin\phi = y$ into the equation above to obtain:

$$\vec{E} = \frac{V_0}{2d^2}x\hat{x} + \frac{V_0}{2d^2}y\hat{y} - \frac{V_0}{d^2}z\hat{z} \tag{3}$$

In the Penning trap, there is uniform $\vec{B}$ field only in z-direction. Knowing the $\vec{E}$ field and $\vec{B}$ field, we can compute the Lorentz force experience by the particle (choose proton as the particle) and its acceleration:

$$\vec{a} = \frac{q}{m}\left(\vec{E} + \vec{v} \times \vec{B}\right). \tag{4}$$

So, inserting all the necessary equation to the equation above, we can obtain the acceleration vector:

$$a_x = \frac{q}{m}\left(E_x + v_y B\right) \tag{5}$$

$$a_y = \frac{q}{m}\left(E_y - v_x B\right) \tag{6}$$

$$a_z = \frac{q}{m}E_z \tag{7}$$

Now, we can get the expressions for the displacement components at the n^{th} time step where $n \geq 2$

$$\frac{x[n+1]-2x[n]+x[n-1]}{h^2} = \frac{q}{m}\left(\frac{V_0}{2d^2}x[n]+v_y[n]B\right) \quad (8)$$

$$\frac{y[n+1]-2y[n]+y[n-1]}{h^2} = \frac{q}{m}\left(\frac{V_0}{2d^2}y[n]-v_x[n]B\right) \quad (9)$$

$$\frac{z[n+1]-2z[n]+z[n-1]}{h^2} = \frac{q}{m}\left(-\frac{V_0}{d^2}z[n]\right) \quad (10)$$

After solving some algebra, we get

$$x[n+1] = k_3\Big[(k_1+2)x[n]+(k_1k_2+2k_2)y[n]+(k_2^2-1)x[n-1]-2k_2y[n-1]\Big] \quad (11)$$

$$y[n+1] = k_3\Big[(k_1+2)y[n]-(k_1k_2+2k_2)x[n]+(k_2^2-1)y[n-1]+2k_2x[n-1]\Big] \quad (12)$$

$$z[n+1] = 2(1-k_1)z[n]-z[n-1] \quad (13)$$

,where

$$k_1 = \frac{qV_0h^2}{2md^2}, \qquad k_2 = \frac{qBh}{2m}, \qquad k_3 = \frac{1}{1+k_2^2}.$$

Hence, with Eq. (11)-(13), we are already able to determine the path that the particle will take. We still need to specify several input parameters: valus of the field constant B and V_0; the initial position and velocity; the time step h; and the number of time steps N. All the values can be found inside the code. For longer time, we only need to set the time steps N to be larger.

2 Notes for the Python code

This section will contains some explanation on parameters that could be adjusted.

- Number of time steps, **N** The higher the N, the more time it will travel in the trajectories.
- The trapped particle
  The code is using proton as the trapped particle. To change the type of particle, simply change the value of its charge **q** and particle's mass **m**. It is important that the particle has to be charged.
- Field value
  The parameter **B** is for the magnitude of the $\vec{B}$ field and $\mathbf{V_0}$ for the coefficient of the quadrupole potential.
- Position of the endcap, $\mathbf{z_0}$
  The position of the endcap is suggested to be in the range from 0 to 1. The ring electrode position, s_0 is always $\sqrt{2}$ of z_0 as suggested by [5], but it is also adjustable if desired.
- Initial position and velocities The initial position is not very important to be adjusted, just keep it close to origin. The initial velocities could be adjusted to see slightly different trajectories.

- Time step **h**
  The time step h follows the equality

$$h < \frac{m}{qB} \tag{14}$$

  and so h must be have a factor that makes it satisfy the equality above. Use smaller factor for even better approximation of the trajectory and increase the number of time steps.

All the position is stored in array and the index runs from 0 to N. Only index from 1 to N has meaning, 0 is meaningless. This is so that it is consistent with the array defined by the formula and the Ian Cooper. The calculation for the velocity and acceleration is not very crucial, but it is included for completeness. The animation plots the trajectory line for all the position (from the array) and has red dot that represents the particle and it will follows the trajectory over time.

The code is written in Spyder (Python). If the code doesn't open a new interactive window, which shows the animation, do the following: Change the backend to automatic:Tools > preferences > IPython console > Graphics > Graphics backend > Backend: Automatic. Then close and open Spyder.

References

1. http://gabrielse.physics.harvard.edu/gabrielse/papers/1990/1990_tjoelker/chapter_2.pdf
2. http://www.physics.usyd.edu.au/teach_res/mp/doc/em_vBE.pdf
3. http://www.physics.usyd.edu.au/teach_res/mp/mscripts/em_vBE_01.m
4. https://jakevdp.github.io/blog/2012/08/18/matplotlib-animation-tutorial/
5. https://www.physi.uni-heidelberg.de/Einrichtungen/FP/anleitungen/F47.pdf
6. https://stackoverflow.com/questions/31303601/animate-a-python-pyplot-by-moving-a-point-plotted-via-scatter