checks

March 8, 2022

[1]:
```
import import_ipynb
from run import Run
from constants import Constants
import numpy as np
```

```
importing Jupyter notebook from run.ipynb
importing Jupyter notebook from batch.ipynb
importing Jupyter notebook from particle.ipynb
importing Jupyter notebook from field.ipynb
importing Jupyter notebook from constants.ipynb
```

[2]:
```
constants = Constants()
```

0.0.1 Section 1: Checks

Before we can do any studies confidently, we need to know if our program works as we expect it to; or wheather there might be bugs causing some erros. For this we perform a few checks, comparing calculations done by hand against the results of the program.

As of now, we can check if the particle sampling and particle update is performed as expected.

[ ]:
```

```

1.1 Sampling So far we have used the Maxwellian distribution for sampling.

1.1.1 Maxwellian sampling For a Maxwellian distribution, we can calculate the average speeds of the particles based on the input parameters like the Plasma temperature and the mass of the species (assuming the same types of species for now). Here we get the average speeds of the sampled particles; which we will compare with a manual calculation.

[3]:
```
# c1Maxwell means checking the Maxwellian sampling
c1Maxwell = Run()
```

[4]:
```
# Create 100 particles based on the data available in the files
c1Maxwell.create_batch_with_file_initialization('H+', constants.
 ↪constants['e'][0],\
                                                 constants.constants['m_H'][0] *␣
 ↪constants.constants['amu'][0], \
```

```
                                         100, 100, 'H ions', r_index=0,␣
↪v_index=1)
```

[5]:
```
# Take the 0th batch of particles
c1Maxwell_batch = c1Maxwell.batches[0]['H ions']
```

[6]:
```
# Take the initial positions and velocities of the particles
c1Maxwell_positions = []
c1Maxwell_velocities = []
for particle in c1Maxwell_batch.particles:
    c1Maxwell_positions.append(particle.r)
    c1Maxwell_velocities.append(particle.v)
```

[7]:
```
# Let's look at the positions
c1Maxwell_positions
# They look fine, all particles are supposed to be initialized at [-0.5, 0, 0]
# where a cube of sides 1m is assumed as the chamber and is described between␣
↪coordinates (-0.5, 0.5) for x, y and z coordinates
```

[7]:
```
[array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
```

```
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
array([-0.5,  0. ,  0. ]),
```

```
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ]),
 array([-0.5,  0. ,  0. ])]
```

[8]:
```
# Let's now look at the velocities
c1Maxwell_velocities
# We need to check if they are really Maxwellian distributed
```

[8]:
```
[array([ 1579.629935, -5723.201089, -2863.641674]),
 array([  897.689953, -5029.491038, -3292.544752]),
 array([ 386.944504, 1027.791639,  756.037653]),
 array([-9204.054924,  9615.130475, -1667.435373]),
 array([ -3641.215516, -10947.299598,  -3360.247147]),
 array([-5491.759943, -7127.660213,  2834.606515]),
 array([16625.53181 ,  7261.248008, 13182.501854]),
 array([ 3531.057922, -8670.854777,  5576.898277]),
 array([-6468.761439, -1081.00466 , 17040.23406 ]),
 array([-6782.695867, -1185.527177,  5481.859317]),
 array([ 2831.836461,  9814.523398, -4603.758126]),
 array([ 6601.297764, -5010.938691,  3131.004259]),
 array([   310.863591,  -1253.75245 , -11537.549212]),
 array([17607.190221, 10702.738406, 10743.593831]),
 array([  1841.536055,  -2739.779189, -12083.999033]),
 array([ 4245.610812, 11143.159359, -4244.35843 ]),
 array([ 6175.85488 , -5538.932063, -2785.536498]),
```

```
array([-18786.399666,  -6481.17661 ,   8700.805047]),
array([ -2630.875393, -12784.228172,   5763.276336]),
array([14103.980493, -5259.190381, 11294.072721]),
array([3218.123565, 2083.965365, 3701.724523]),
array([  8110.7652  ,   1611.055445, -10797.61461 ]),
array([ -5795.493516,  -6048.466929, -16270.769749]),
array([ -5497.184733, -12480.106514,   1557.429411]),
array([-13982.702604,   4446.07858 ,   8748.309843]),
array([ 6562.348371,  2376.507555, 16756.176696]),
array([ 1564.595443,  4923.296004, -1502.965034]),
array([ 3999.538647,  5265.782508, 20981.06687 ]),
array([2167.907498, 4715.441563, -383.229814]),
array([-8141.437181,  -707.79037 , 18484.123017]),
array([  4328.135274,  -2726.982622, -17110.480553]),
array([14064.293936,  1939.197086,  8497.465692]),
array([ 15755.484194,   1021.183689, -10923.123019]),
array([ -4505.746111, -10715.356198,  -4957.862115]),
array([  311.037312, 10372.211689, -7924.10112 ]),
array([   194.746288, -16705.966313, -25158.665103]),
array([3917.707815, 2788.029633, 3845.298211]),
array([-6805.073646,  -114.745473, -2991.933822]),
array([-20778.676767,   1204.886391,   1224.16076 ]),
array([-1049.374644, 25399.499396,  3260.046569]),
array([-1613.853665, 11450.244641,  6833.552532]),
array([-4450.714515, 15520.095122, -4093.634965]),
array([ -302.040006, -2774.294494,  8039.097495]),
array([  837.443914, 19004.422265, -3366.240414]),
array([-18089.011037,  -3702.184851,    973.063181]),
array([ 1290.989261, -6265.270624, -6101.462157]),
array([  627.872857,  7529.794087, -9586.239895]),
array([-15877.458159,  -1685.890055,   2469.728849]),
array([   401.980725,   4648.362411, -22877.706831]),
array([ 13391.765702, -13467.286656,  -5302.578527]),
array([  4424.274293,   1718.322218, -10218.543702]),
array([-16491.662572,  -1376.577214,   1212.84242 ]),
array([ 7398.623459, 12996.316882, -9925.234194]),
array([-5142.450136, -4269.804713,  9158.1664  ]),
array([ 2574.281443, 12535.307273,  1430.257433]),
array([  9149.705078, -16458.827279,   3392.746971]),
array([-7809.646458, -9975.384339,  6515.163693]),
array([ 3364.849961,  9514.632099, 10146.688711]),
array([11034.368466,  2228.723503, 10336.824708]),
array([  3592.570787,   9359.729427, -15987.66503 ]),
array([-9948.531042,  8687.897191, -1985.537473]),
array([-21346.36499 ,    315.575762,  -4095.710895]),
array([8745.421099, 1817.124211, -202.137507]),
array([-4503.432576,  5444.488763, -4660.01537 ]),
```

```
array([22320.899741,  -285.109853, 10270.464622]),
array([11379.305971,  5524.826899,  3963.333405]),
array([-4991.704923, -3821.720667,  2668.679899]),
array([-1864.217875, -1091.525555, -5304.013859]),
array([-11280.765472,  14952.883947,   6873.951826]),
array([  7157.964588,  10566.45093 , -10696.666068]),
array([-5803.460516,   174.639384,  7935.079774]),
array([-7054.045335,  4371.138014, 14677.551923]),
array([6675.742123, 2086.149166, 2812.089298]),
array([ 1629.343137, 12814.296519, -1575.882811]),
array([  1372.398103,   6908.124601, -11080.46534 ]),
array([ 4077.427095,  9806.823708, -2952.864354]),
array([20687.74124 , 10234.694055,  4265.535278]),
array([-1161.868381, -9148.174165, -5257.7664  ]),
array([ -9429.693013,   4349.497355, -17731.864223]),
array([ 4360.301531, -2415.241844, -9765.025732]),
array([12484.9937  , -5046.17275 ,  2094.336482]),
array([  920.001379, -2826.506275, -6130.965348]),
array([  7977.635576, -14613.882278,  -7575.019106]),
array([ 2208.036015, -9531.481613,  5584.708069]),
array([  -92.825592,  4472.404515, -2098.645854]),
array([ -2929.734057, -12483.318971,   -383.726147]),
array([ 7781.299771,  7546.658659, -7793.261556]),
array([ 4995.961777, -9268.285396,  8484.176491]),
array([11248.219314,  8449.902291, 10343.354335]),
array([ -9901.37255 ,  -1143.945737, -11700.414233]),
array([  1499.538688, -10332.224092,   1408.641439]),
array([17961.404303, 15944.610445,  3105.683781]),
array([17084.296062, -2164.739467, -2088.521291]),
array([-11764.890976,  -4225.774814,   7203.046946]),
array([ -4983.672944,   2659.037113, -14658.970414]),
array([-2639.873802, -7685.606637,  5666.133006]),
array([ -3893.19326 , -10739.510159,  12748.028668]),
array([3349.164108,  527.511892, 1366.528714]),
array([-11336.739018,  13658.143203,  13176.412033]),
array([  -765.772707,  -3802.623207, -10641.798323])]
```

[9]:
```
# Get the speeds
c1Maxwell_speeds = np.sqrt( \
                          [ (c1Maxwell_velocities[i][0] ** 2) +␣
↪(c1Maxwell_velocities[i][1] ** 2) + \
                           (c1Maxwell_velocities[i][2] ** 2) for i in␣
↪range(len(c1Maxwell_velocities)) ] )
```

[10]:
```
c1Maxwell_speeds
```

[10]:
```
array([ 6591.7148811 ,  6078.03243632,  1333.29465428, 13414.38413862,
       12016.36716363,  9433.87309001, 22425.70814333, 10897.42570538,
       18258.77932528,  8801.20556986, 11204.40788951,  8859.44842548,
       11609.63277617, 23237.61100611, 12526.79838091, 12657.40060792,
        8751.00939772, 21694.20362969, 14267.91328164, 18818.53947143,
        5329.35224641, 13600.31224062, 18300.5367547 , 13725.80361799,
       17082.63773428, 18151.62972763,  5380.12140539, 21998.40771087,
        5204.04430653, 20210.06606342, 17858.82789898, 16546.05005563,
       19198.76857515, 12637.28622839, 13056.45043995, 30200.75605565,
        6156.93609596,  7434.63932187, 20849.54985051, 25629.35153965,
       13431.68143601, 16656.53203751,  8509.70191245, 19318.40967731,
       18489.60099412,  8840.1419512 , 12206.06483031, 16156.59195053,
       23348.62590073, 19718.63427135, 11267.00801696, 16593.39886673,
       17948.653303  , 11337.90269399, 12876.58688088, 19134.28416282,
       14245.91971255, 14311.0352888 , 15283.1423778 , 18871.05012758,
       13356.46609518, 21738.0250732 ,  8934.49439917,  8463.98878223,
       24572.06333273, 13255.9544479 ,  6829.67929688,  5727.06725464,
       19952.33374001, 16652.47751145,  9832.40275289, 16861.10712789,
        7538.26223048, 13013.23791256, 13129.44683553, 11023.54801541,
       23471.82121652, 10615.23133073, 20548.87457705, 10963.63763323,
       13628.10230413,  6813.53625941, 18291.77790908, 11265.59048601,
        4941.18742377, 12828.24383664, 13350.49095434, 13521.908112  ,
       17461.62264403, 15370.27922475, 10535.07198209, 24217.51264119,
       17347.08015999, 14427.53332216, 15709.64315252,  9906.69199637,
       17117.42583262,  3655.48488503, 22106.20639955, 11326.7039619 ])
```

[11]:
```
c1Maxwell_meanspeed = np.sum(c1Maxwell_speeds) / c1Maxwell_speeds.size
c1Maxwell_meanspeed
```

[11]:
```
14202.764572898674
```

[12]:
```
c1Maxwell_meanspeed_expected = (2 / np.sqrt(np.pi) ) * np.sqrt( (2 * constants.
↪constants['K'][0] * constants.constants['N_A'][0] * 10000)/ ( constants.
↪constants['m_H'][0] * 10**(-3)) )
c1Maxwell_meanspeed_expected
```

[12]:
```
14492.952993825973
```

[13]:
```
(c1Maxwell_meanspeed_expected - c1Maxwell_meanspeed) * 100/ c1Maxwell_meanspeed
# We see a maximum of 2.04 % discrepency
```

[13]:
```
2.0431826454479785
```

We see that the numbers are close We were using Hydrogen atom samples. from the second (1th) available velocity sampled file. Now let's use Hydrogen gas samples, from the third (2nd) available velocity sampled file; to see if the numbers are still close.

[14]:
```
# Create 100 particles based on the sampled velocities of H2 gas
# We consider just for this test case that a Hydrogen molecules to be sampled␣
 ↪with a Maxwellian distribution
# We create a new batch on the same Run instance
c1Maxwell.create_batch_with_file_initialization('H2', constants.
 ↪constants['e'][0],\
                                                 2 * constants.constants['m_H'][0] *␣
 ↪constants.constants['amu'][0], \
                                                 100, 100, 'H2 gas', r_index=0,␣
 ↪v_index=2)
```

[15]:
```
# Do the same as before for the second batch
c1Maxwell_batch2 = c1Maxwell.batches[1]['H2 gas']
c1Maxwell_positions_batch2 = []
c1Maxwell_velocities_batch2 = []
for particle in c1Maxwell_batch2.particles:
    c1Maxwell_positions_batch2.append(particle.r)
    c1Maxwell_velocities_batch2.append(particle.v)
c1Maxwell_speeds_batch2 = np.sqrt( \
                           [ (c1Maxwell_velocities_batch2[i][0] ** 2) +␣
 ↪(c1Maxwell_velocities_batch2[i][1] ** 2) + \
                            (c1Maxwell_velocities_batch2[i][2] ** 2) for i in␣
 ↪range(len(c1Maxwell_velocities_batch2)) ] )
c1Maxwell_meanspeed_batch2 = np.sum(c1Maxwell_speeds_batch2) /␣
 ↪c1Maxwell_speeds_batch2.size
c1Maxwell_meanspeed_batch2
```

[15]:
```
10149.754879907316
```

[16]:
```
# We multiply the mass of Hydrogen by 2 to get the numbers for hydrogen molecule
c1Maxwell_meanspeed_expected_batch2 = (2 / np.sqrt(np.pi) ) * np.sqrt( (2 *␣
 ↪constants.constants['K'][0] * constants.constants['N_A'][0] * 10000)/ ( 2 *␣
 ↪constants.constants['m_H'][0] * 10**(-3)) )
c1Maxwell_meanspeed_expected_batch2
```

[16]:
```
10248.06534135222
```

[17]:
```
(c1Maxwell_meanspeed_expected_batch2 - c1Maxwell_meanspeed_batch2) * 100/␣
 ↪c1Maxwell_meanspeed_batch2
# We see a maximum of 0.97% discrepency
```

[17]:
```
0.9685993662716086
```

Again we see that the numbers are close enough For a sample of large enough number of particles, we would expect the match to be better. However, for now we take this as an indication that our program is working as we expect it to.

[ ]:

1.2 Update Here we check if our particle update works as expected.

1.2.1 Boris Update So far we use the Boris Algorithm to update individual particles in the Electric and Magnetic fields, here we set up a simple configuration, and observe the result of update of a single particle. We compare it to calculations done by hand.

[18]:
```
# c2Boris means checking the Boris update
c2Boris = Run()
```

[19]:
```
#Create 10 particles
c2Boris.create_batch_with_file_initialization('H+', constants.
 ↪constants['e'][0],\
                                               constants.constants['m_H'][0] *␣
 ↪constants.constants['amu'][0], \
                                               100, 10, 'H ions', r_index=0,␣
 ↪v_index=1)
```

[20]:
```
c2Boris_index_update = 0 # Update the first batch in this Run instance␣
 ↪run_Boris_check
c2Boris_particle_track_indices = [i for i in range(10)] # Track all 10 particles
c2Boris_dT = 10**(-6) # 1 microseconds
c2Boris_stepT = 10**(-7) # 0.1 microseconds time step
c2Boris_E0 = 1000 # say 1000 Volts (voltage) per meter (size of chamber)
c2Boris_Edirn = [1,0,0] #in the x-direction [1,0,0]
c2Boris_B0 = 10 * (10**(-3)) # Meant to say 10 mT
c2Boris_Bdirn = [0,1,0] #in the y-direction [0,1,0]
c2Boris_argsE = [element * c2Boris_E0 for element in c2Boris_Edirn] # currently␣
 ↪the uniform_E_field configuration is used
c2Boris_argsB = [element * c2Boris_B0 for element in c2Boris_Bdirn]# currently␣
 ↪the uniform_B_field configuration is used
```

[21]:
```
c2Boris_positions_and_velocities = c2Boris.
 ↪update_batch_with_unchanging_fields(c2Boris_index_update, \
                                                                                 ␣
 ↪c2Boris_dT, c2Boris_stepT, \
                                                                                 ␣
 ↪c2Boris_argsE, c2Boris_argsB, \
                                                                                 ␣
 ↪c2Boris_particle_track_indices)
```

[22]:
```
#Let's inspect the positions and velocities of the particles at index 1 and 7
c2Boris_p1 = c2Boris_positions_and_velocities[1]
c2Boris_p7 = c2Boris_positions_and_velocities[7]
```

[23]:
```
#Let's look at particle 1's positions and velocities
c2Boris_p1
```

[23]:
```
[(0,
  array([-4.98921519e-01, -5.02949104e-04, -2.74850639e-04]),
  array([10784.80849437, -5029.491038  , -2748.50639353])),
 (1,
  array([-4.96859534e-01, -1.00589821e-03, -4.00658364e-04]),
  array([20619.85191499, -5029.491038  , -1258.07724484])),
 (2,
  array([-4.93828311e-01, -1.50884731e-03, -2.83282555e-04]),
  array([30312.2320901 , -5029.491038  ,  1173.75808561])),
 (3,
  array([-4.89851127e-01, -2.01179642e-03,  1.70051838e-04]),
  array([39771.83801957, -5029.491038  ,  4533.34393251])),
 (4,
  array([-0.48496014, -0.00251475,  0.00104989]),
  array([48909.86581774, -5029.491038  ,  8798.39924387])),
 (5,
  array([-0.47919618, -0.00301769,  0.00244371]),
  array([57639.6443318 , -5029.491038  , 13938.14269746])),
 (6,
  array([-0.47260843, -0.00352064,  0.00443506]),
  array([65877.44878403, -5029.491038  , 19913.49684507])),
 (7,
  array([-0.4652541 , -0.00402359,  0.00710279]),
  array([73543.29487349, -5029.491038  , 26677.37013834])),
 (8,
  array([-0.45719793, -0.00452654,  0.01052029]),
  array([80561.70588222, -5029.491038  , 34175.01496554])),
 (9,
  array([-0.44851169, -0.00502949,  0.01475474]),
  array([86862.44551045, -5029.491038  , 42344.45911554]))]
```

[24]:
```
#Let's take the positions and velocities of the particle 1 after the 3rd and␣
 ↪the 4th time steps.
c2Boris_p134_p = [c2Boris_p1[3][1], c2Boris_p1[4][1]]
c2Boris_p134_v = [c2Boris_p1[3][2], c2Boris_p1[4][2]]
```

[25]:
```
#Likewise for particle 7
c2Boris_p734_p = [c2Boris_p1[3][1], c2Boris_p1[4][1]]
c2Boris_p734_v = [c2Boris_p1[3][2], c2Boris_p1[4][2]]
```

[26]:
```
#Let's look at the positions to check if they match up against the positions␣
 ↪and velocities from above
c2Boris_p134_p
```

[26]:
```
[array([-4.89851127e-01, -2.01179642e-03,  1.70051838e-04]),
 array([-0.48496014, -0.00251475,  0.00104989])]
```

[27]:
```
#Let's look at the velocities to check if they match up against the positions␣
 ↪and velocities from above
c2Boris_p134_v
```

[27]:
```
[array([39771.83801957, -5029.491038  ,  4533.34393251]),
 array([48909.86581774, -5029.491038  ,  8798.39924387])]
```

[28]:
```
print(f'The velocity changed from \n {[c2Boris_p134_v[0][0],␣
 ↪c2Boris_p134_v[0][1], c2Boris_p134_v[0][2]]} \n to \n␣
 ↪{[c2Boris_p134_v[1][0], c2Boris_p134_v[1][1], c2Boris_p134_v[1][2]]} \n')
print(f'The position changed from \n {[c2Boris_p134_p[0][0],␣
 ↪c2Boris_p134_p[0][1], c2Boris_p134_p[0][2]]} \n to \n␣
 ↪{[c2Boris_p134_p[1][0], c2Boris_p134_p[1][1], c2Boris_p134_p[1][2]]} \n')
```

```
The velocity changed from
 [39771.83801956555, -5029.491038, 4533.343932509505]
 to
 [48909.86581773698, -5029.491038, 8798.399243869582]

The position changed from
 [-0.48985112694809785, -0.0020117964152, 0.00017005183797547688]
 to
 [-0.4849601403663242, -0.0025147455189999997, 0.001049891762362435]
```

[29]:
```
#Similary for particle 7
c2Boris_p734_p = [c2Boris_p7[3][1], c2Boris_p7[4][1]]
c2Boris_p734_v = [c2Boris_p7[3][2], c2Boris_p7[4][2]]
print(f'The velocity changed from \n {[c2Boris_p734_v[0][0],␣
 ↪c2Boris_p734_v[0][1], c2Boris_p734_v[0][2]]} \n to \n␣
 ↪{[c2Boris_p734_v[1][0], c2Boris_p734_v[1][1], c2Boris_p734_v[1][2]]} \n')
print(f'The position changed from \n {[c2Boris_p734_p[0][0],␣
 ↪c2Boris_p734_p[0][1], c2Boris_p734_p[0][2]]} \n to \n␣
 ↪{[c2Boris_p734_p[1][0], c2Boris_p734_p[1][1], c2Boris_p734_p[1][2]]} \n')
```

```
The velocity changed from
 [38895.85871094631, -8670.854777, 13914.969423620274]
 to
 [47135.881299118584, -8670.854777, 18096.176367366777]

The position changed from
 [-0.4896669752432719, -0.0034683419108, 0.0038875496903932644]
 to
 [-0.48495338711336006, -0.0043354273885, 0.0056971673271299424]
```

[30]:
```
c2Boris_qp = (constants.constants['e'][0] * 10**(-7)) / (2 * constants.
↪constants['m_H'][0] * constants.constants['amu'][0])
```

[31]:
```
c2Boris_qp
```

[31]:
```
4.78597877761177
```

[32]:
```
# The velocity after update 4, doing a manual update.
c2Boris_vminus_p1 = np.add(c2Boris_p134_v[0], c2Boris_qp * np.
↪array(c2Boris_argsE))
c2Boris_vplus_p1 = np.add(c2Boris_vminus_p1, 2 * c2Boris_qp * np.
↪cross(c2Boris_vminus_p1, c2Boris_argsB))
c2Boris_vnew_p1 = np.add(c2Boris_vplus_p1, c2Boris_qp * np.array(c2Boris_argsE))
```

[33]:
```
c2Boris_vnew_p1
#This is indeed the same as the velocity of particle 1 after update 4.
np.isclose(c2Boris_vnew_p1, c2Boris_p134_v[1]) #Indeed the values are close (i.
↪e. the same)
```

[33]:
```
array([ True,  True,  True])
```

[34]:
```
#Similarly for position of the particle 1 after update 4
c2Boris_pnew_p1 = np.add(c2Boris_p134_p[0], c2Boris_stepT * c2Boris_vnew_p1)
```

[35]:
```
c2Boris_pnew_p1
#This is indeed the same as the position of particle 1 after update 4.
np.isclose(c2Boris_pnew_p1, c2Boris_p134_p[1]) #Indeed th values are close (i.e.
↪ the same)
```

[35]:
```
array([ True,  True,  True])
```

[36]:
```
c2Boris_p134_p[0]
```

[36]:
```
array([-4.89851127e-01, -2.01179642e-03,  1.70051838e-04])
```

[37]:
```
c2Boris_p134_v[0]
```

[37]:
```
array([39771.83801957, -5029.491038  ,  4533.34393251])
```

[ ]:
```

```

[38]:
```
#Similarly for particle 7
c2Boris_vminus_p7 = np.add(c2Boris_p734_v[0], c2Boris_qp * np.
↪array(c2Boris_argsE))
```

```
c2Boris_vplus_p7 = np.add(c2Boris_vminus_p7, 2 * c2Boris_qp * np.
↪cross(c2Boris_vminus_p7, c2Boris_argsB))
c2Boris_vnew_p7 = np.add(c2Boris_vplus_p7, c2Boris_qp * np.array(c2Boris_argsE))
```

[39]:
```
np.isclose(c2Boris_vnew_p7, c2Boris_p734_v[1])
```

[39]:
```
array([ True,  True,  True])
```

[40]:
```
c2Boris_pnew_p7 = np.add(c2Boris_p734_p[0], c2Boris_stepT * c2Boris_vnew_p7)
np.isclose(c2Boris_pnew_p7, c2Boris_p734_p[1])
```

[40]:
```
array([ True,  True,  True])
```

[41]:
```
c2Boris_pnew_p7
```

[41]:
```
array([-0.48495339, -0.00433543,  0.00569717])
```

[ ]:
```

```