tmps

thermo-magnetic particle simulator
mop_sim.py
By Logan Hillberry
lhillberry@gmail.com
Last updated 21 September 2017

DESCRIPTION

This script enables dynamical simulation of Natoms = 100000+ particles with dipole moment μ and mass m interacting with an applied magnetic field $\mathbf{B}$. The particular field considered in this file is that of Helmholtz and anti-Helmoltz coil pairs. This field is of interest because it provides an approximatly constant gradient superposed with a uniform, directional bias. The field is used to provide a one-dimensional kick to the dipolar particles. The magneto-static equations for the field vectors come from the exact solution to a current carrying loop. The unique components (radial and z) are proportional to linear combinations of the complete elliptic integrals K and E. Supported 3D geometries include single current loops, MxN current coils, Helmoltz (HH)/ anti-Helmholtz (AH) coil pairs, and the HH + AH configuration used for magnetic-optical (MOP) cooling, called the MOP configuration. Additionally, the script supplies useful coil vizualization functions which orient a closed tube in three dimensional space with a thickness given in the same units as the axes. Several functions drew heavily from existing online resources, urls for which are given contextually in the source code. Dynamics are computed with an rk4 algorithm available as a method to the Cloud class. The acceleration used in rk4 is calculated at any point in spacetime by multiplying the gradient of the norm of the normalized magnetic field evaluated at point r, by a time dependent current signal evaluated at time t,

$$a(r,t) = \pm\frac{\mu}{m} I(t)\ \nabla|B(r, I=1))| \tag{1}$$

The sign is given by the internal magnetic state of the atom (+1 or -1) which is tracked as a seventh dimension of phase space. We assume a quantization axis of z. This means the mop coils are extpected to have a normal of [0,0,1] even though the feild calculator can handle arbitrary orientation. Factoring the time dependent current past the spatial derivative is only possible in the mop configuration if all coils have current signals proportional to one another. Furthermore, since evaluating the complete elliptic integrals can become expensive, we first evalute the field everywhere on a spatial grid, take its magnitude, then take the gradient of the result. Then, during time evolution, the acceleration vector of each particle is given component wise by three independent linear interpolations of the grad-norm-B grid.

At the end of the simulation, the full state of the cloud (including trajectories, velocities, temperatures, densities, and phase space density, as well as the B field grid used define the acceleration function) is saved to disk as a binary file. The default behavior also generates several plots saved to a single file called mop_sim_default.pdf inside a user created directory c alled `plots`. If `animate_traj` is uncommented inside the default behavior the script will also generate an animated trajectory of the simulation, **but this is very slow and shold not be used in its current state with more than a couple hundered particles**, possibly running over night. Once a simulation is saved, it can be re loaded. this can be particularly useful to avaoid recalculating the B-field grid while changing cloud and pulse parameters. See the function `run_sim` for more loading flags.

USAGE

(Assuming a bash-like shell) Create a folder called `plots` in the directory containing this script mkdir plots To execute the default behavior, run python3 mop_sim.py in the directory containing this script. It will generate a file of several plot called plots called mop_sim.pdf in the same directory.

REQUIREMENTS

python3, numpy, scipy, and matplotlib

COIL GEOMETRY

n, loop normal, current flow given by right hand rule
r0, center-of-loop or center-of-coil-pair position vector [x0, y0, z0], cm
R, loop radius, cm
d, diameter of wire, cm
M, number of loops per coil layer, int
N, number of layers per coil, int
A, coil pair half-separation, cm
r, field points [x, y, z] or [[x1, y1, z1], [x2, y2, z2], . . .]
B* functions return [Bx, By, Bz] or [[Bx1, By1, Bz1], [Bx2, By2, Bz2], . . .]
Vacuum permiability sets units for the simulation, defined here globally