```
#--------------------
# PYTHON VARIABLES
#--------------------

from math import pi, sqrt, log, exp
import numpy as np

L_0 = 2.*pi          # laser wavelength (lambda)
t_0 = L_0            # optical cycle

dx = 0.05*L_0         # cells per wavelength
dr = 0.2*L_0
Lx = 160.0 * L_0      # grid size
Lr = 160.0 * L_0
nx = Lx/dx
nr = Lr/dr

t_sim = 25000.0 * t_0

t_fwhm = 10.6*t_0      # c*tau/lambda

nppc = 16             # number of particles-per-cell
lambda0 = 0.85e-6

#ne = 0.02 #(3e-2/1.1)*L_0**2 # n_e = 3e19
diag_every = 10*t_0 / dt

def t_to_timestep(time):
    return time / dt

#get number density of gas target
scale = 33

a1 = 8.7e17
b1 = 66527 + 76050
c1 = 7392
a2 = 8.7e17
b2 = -66527 + 76050
c2 = 7392

p1 = 1.183e-12
p2 = -5.387e-07
p3 = 0.09808
p4 = -9090
p5 = 4.495e+08
p6 = -1.118e+13
p7 = 9.282e+17
```

```
Radius_plasma =128*L_0

def n_He(x,r):
    if x > 60*L_0:
        return (r<Radius_plasma)*0.5*(0.85**2)/(1.1*10e21)*( (scale/9
                *(a1*exp(-((x-b1)/c1)**4) + a2*exp(-((x-b2)/c2)**4))
                *(x < -64819  + 77450)) + (scale/9*(p1*(x)**6 + p2*(x)**5
                + p3*(x)**4 + p4*(x)**3 + p5*(x)**2 + p6*(x) + p7)
                *(x > -64819 +77450)*(x < 64819 +75150)) + (scale/9*
                (a1*exp(-((x-b1)/c1)**4) + a2*exp(-((x-b2)/c2)**4))
                *(x > 64819 + 75150)*(x < 76273 + 75150)) )
    else:
        return 0

def ne(x,r):
    if x > 60*L_0:
        return (r<Radius_plasma)*(0.85**2)/(1.1*10e21)*( (scale/9
                *(a1*exp(-((x-b1)/c1)**4) + a2*exp(-((x-b2)/c2)**4))
                *(x < -64819  + 77450)) + (scale/9*(p1*(x)**6 + p2*(x)**5
                + p3*(x)**4 + p4*(x)**3 + p5*(x)**2 + p6*(x) + p7)
                *(x > -64819 +77450)*(x < 64819 +75150)) + (scale/9
                *(a1*exp(-((x-b1)/c1)**4) + a2*exp(-((x-b2)/c2)**4))
                *(x > 64819 + 75150)*(x < 76273 + 75150)) )
    else:
        return 0

#----------------
# SMILEY NAMELIST
#----------------

Main(
    geometry = "AMcylindrical",
    number_of_AM = 2,
    #patch_arrangement = "linearized_XY",
    timestep_over_CFL = 0.95,
    simulation_time = t_sim,
    cell_length  = [dx, dr],
    grid_length = [ Lx, Lr ],
    number_of_patches = [128, 32],
    EM_boundary_conditions = [
        ["PML","PML"],
        ["PML","PML"],
    ],
    solve_poisson = False,
    print_every = 100,
    reference_angular_frequency_SI = 2*pi*3e8/lambda0,
    random_seed = smilei_mpi_rank
)
```

```
MovingWindow(
    time_start = 130.*t_0,
    velocity_x = 0.9985
)

Species(
    name = 'He_ions',
    atomic_number = 2,
    position_initialization = 'regular',
    momentum_initialization = 'cold',
    particles_per_cell = nppc,
    mass = 4.*1836.0,
    charge = 2.0,
    time_frozen = t_sim,
    number_density = n_He,
    boundary_conditions = [
        ["remove", "remove"],
        ["reflective", "remove"],
    ],
)


Species(
    name = 'electron',
    position_initialization = 'regular',
    momentum_initialization = 'cold',
    particles_per_cell = nppc,
    mass = 1.0,
    charge = -1.0,
    number_density = ne,
    mean_velocity = [0.0, 0.0, 0.0],
    boundary_conditions = [
        ["remove", "remove"],
        ["reflective", "remove"],
    ],
)

LaserGaussianAM(
    box_side         = "xmin",
    a0               = 2.1,
    omega                    = 1,
    focus            = [40000.0, 0], # [295.0 * L_0, 0.5 * Ly]
    waist            = 53 * L_0,  # at 1/e electric field ... w_0
    polarization_phi = pi / 2.0,
    time_envelope    = tgaussian(fwhm=t_fwhm, center=2.0*t_fwhm)
)
```

```
Checkpoints(
    # restart_dir = "dump1",
    dump_step = 15000,
    #dump_minutes = 240.,
    exit_after_dump = False,
    keep_n_dumps = 2,
)

DiagFields(
    every = 30 * t_0 / dt,
    fields = ["Et_mode_1"]
)

def my_filter(particles):
    return np.sqrt(1.0 + particles.px**2 + particles.py**2 + particles.pz**2) > 2

DiagTrackParticles(
    species = "electron",
    every = 30 * t_0 / dt,
#     flush_every = 100,
    filter = my_filter,
    attributes = ["x", "y", "z", "px", "py", "pz", "w"]
)

DiagProbe(
    every = 30 * t_0 / dt,
    origin   = [0., -nr*dr,0.],
    corners  = [ [nx*dx,-nr*dr,0.], [0,nr*dr,0.] ],
    number   = [nx, 2*nr],
    fields = ['Ex','Ey','Rho_electron','Ez']
)

DiagParticleBinning(
    #name = "energy spectrum"
    deposited_quantity = "weight",
    every = diag_every,
    time_average = 1,
    species = ["electron"],
    axes = [ ["ekin", 0, 150, 500] ]
)
```