```
#--------------------
# 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 = 0.0125*L_0        # cells per wavelength
dr = 0.05*L_0
Lx = 64.0 * L_0     # grid size
Lr = 32.0 * L_0
nx = Lx/dx
nr = Lr/dr
dt = 0.95*dx  # Courant-Levy condition
t_sim = 800.0 * t_0
d_fwhm = 3.0 * L_0  # focal spot / lambda
t_fwhm = 5.5*t_0     # c*tau/lambda
nppc = 16            # number of particles-per-cell
lambda0 = 0.85e-6
diag_every = 10*t_0 / dt

#get number density of gas target
a1 =   1.325e+19
b1 =   2461
c1 =   717.7
a2 =   1.325e+19
b2 =   1309
c2 =   717.7

Radius_plasma = 30*L_0
def ne(x,r):
    if r < Radius_plasma:
        return (1.8*0.65*((0.85**2)*(a1*exp(-(((x)-b1)/c1)**2)
        + a2*exp(-(((x)-b2)/c2)**2))/(1.1*10e21)))*(x< 600*2*pi)
    else:
        return 0

r_n = 0.085*pi #d=85nm, r = 0.085/2*2*pi
def nanoparticle(x,r):
    if (x-250*2*pi)**2+(r)**2 < r_n**2:
       return 40
    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 = [256, 64],
    EM_boundary_conditions = [
        ["silver-muller","silver-muller"],
        ["buneman","buneman"],
    ],
    solve_poisson = False,
    print_every = 100,
    reference_angular_frequency_SI = 2*pi*3e8/lambda0,
    random_seed = smilei_mpi_rank
)

MovingWindow(
    time_start = 50.*t_0,
    velocity_x = 0.995
)

Species(
    name = 'nitrogen',
    ionization_model = 'tunnel',
    ionization_electrons = 'electronN',
    atomic_number = 7,
    position_initialization = 'regular',
    momentum_initialization = 'cold',
    particles_per_cell = nppc,
    mass = 14.*1836.0,
    charge = 0.0,
    time_frozen = t_sim,
    number_density = ne,
    boundary_conditions = [
        ["remove", "remove"],
        ["reflective", "remove"],
    ],
)
```

```
Species(
    name = 'aluminium',
    ionization_model = 'tunnel',
    ionization_electrons = 'electronAl',
    atomic_number = 13,
    position_initialization = 'regular',
    momentum_initialization = 'cold',
    particles_per_cell = 100*nppc,
    mass = 26.*1836.0,
    charge = 0.0,
        time_frozen = t_sim,
    number_density = nanoparticle,
    boundary_conditions = [
        ["remove", "remove"],
        ["reflective", "remove"],
    ],
)

Species(
    name = 'electronN',
    position_initialization = 'regular',
    momentum_initialization = 'cold',
    particles_per_cell = 0,
    mass = 1.0,
    charge = -1.0,
    number_density = 0,
    mean_velocity = [0.0, 0.0, 0.0],
    boundary_conditions = [
        ["remove", "remove"],
        ["reflective", "remove"],
    ],
)

Species(
    name = 'electronAl',
    position_initialization = 'regular',
    momentum_initialization = 'cold',
    particles_per_cell = 0,
    mass = 1.0,
    charge = -1.0,
    number_density = 0.0,
        mean_velocity = [0.0, 0.0, 0.0],
    boundary_conditions = [
        ["remove", "remove"],
        ["reflective", "remove"],
    ],
)
```

```
LaserGaussianAM(
    box_side         = "xmin",
    a0               = 2.2,
    omega            = 1,
    focus            = [295.0 * L_0, 0], # [295.0 * L_0, 0.5 * Ly]
    waist            = 3.5 * L_0,
    polarization_phi = pi / 2.0,
    time_envelope    = tgaussian(fwhm=t_fwhm, center=2.0*t_fwhm)
)

DiagProbe(
    every = 10 * 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_electronN', 'Rho_electronAl', 'Ez']
)

DiagFields(
    every = 10 * t_0 / dt,
    fields = ["Et_mode_1"]
)

def my_filter(particles):
        if Main.iteration * Main.timestep < 230*2*pi:
                return (particles.y < 1.)*(particles.x < 1.)
        elif Main.iteration * Main.timestep > 400*2*pi:
                return np.sqrt(1.0 + particles.px**2 +  particles.py**2
                                + particles.pz**2) > 2
        else:
                return (particles.y > -100.)*(particles.y < 100.)
                    *(particles.z > -100.)*(particles.z < 100.)

DiagTrackParticles(
    species = "electronN",
    every = diag_every,
    filter = my_filter,
    attributes = ["x", "y", "z", "px", "py", "pz", "w"]
)

DiagTrackParticles(
    species = "electronAl",
    every = diag_every,
    filter = my_filter,
    attributes = ["x", "y", "z", "px", "py", "pz", "w"]
)
```

```
DiagParticleBinning(
    #name = "energy spectrum"
    deposited_quantity = "weight",
    every = diag_every,
    time_average = 1,
    species = ["electronN"],
    axes = [ ["ekin", 0, 70, 500] ]
)

DiagParticleBinning(
    deposited_quantity = "weight",
    every = diag_every,
    time_average = 1,
    species = ["electronN"],
    axes = [ ["x", 4650, 4900, 500],
             ["y", 0, 200, 500]]
)

DiagParticleBinning(
    deposited_quantity = "weight",
    every = diag_every,
    time_average = 1,
    species = ["electronN"],
    axes = [ ["x", 4650, 4900, 500],
             ["z", 0, 200, 500]]
)

DiagParticleBinning(
    deposited_quantity = "weight",
    every = diag_every,
    time_average = 1,
    species = ["electronN"],
    axes = [ ["y",  -200, 200, 500],
             ["z", -200, 200, 500]]
)

DiagParticleBinning(
    #name = "longitudinal phase space"
    deposited_quantity = "weight",
    every = diag_every,
    time_average = 1,
    species = ["electronN"],
    axes = [ ["x",    4650, 4900,   500],
             ["px",    0,   100,   500]]
)
```

```
DiagParticleBinning(
    #name = "energy spectrum"
    deposited_quantity = "weight",
    every = diag_every,
    time_average = 1,
    species = ["electronAl"],
    axes = [ ["ekin", 0, 70, 500] ]
)

DiagParticleBinning(
    #name = "longitudinal phase space"
    deposited_quantity = "weight",
    every = diag_every,
    time_average = 1,
    species = ["electronAl"],
    axes = [ ["x",    4650, 4900,   500],
             ["px",    0,   100,   500]]
)

DiagParticleBinning(
    deposited_quantity = "weight",
    every = diag_every,
    time_average = 1,
    species = ["electronAl"],
    axes = [ ["x",  4650, 4900, 500],
             ["y", -200, 200, 500]]
)

DiagParticleBinning(
    deposited_quantity = "weight",
    every = diag_every,
    time_average = 1,
    species = ["electronAl"],
    axes = [ ["x",  4650, 4900, 500],
             ["z", -200, 200, 500]]
)

DiagParticleBinning(
    deposited_quantity = "weight",
    every = diag_every,
    time_average = 1,
    species = ["electronAl"],
    axes = [ ["y",  -200, 200, 500],
             ["z", -200, 200, 500]]
)
```