```python
import numpy as np
from constants_lineRT import *

''' Load preset abundances and fraction for collision coefficients
    PARAMS:
      species = string with the particle name
    RETURNS:
      comp_fracs = list with the fraction of total collision partner density
       for each partner
      abundance = overall abundance of the molecule (assume n_mol =
       abundance*rho everywhere)'''
def load_species_info(species):

    if species == 'HCO+':
        comp_fracs = [1.0] # only 1 collision partner in HCO data
        abundance = 1.e-09 # = N_species/N_h2
    elif species == 'H13CO+':
        comp_fracs = [1.0] # only 1 collision partner in HCO data
        abundance = 2.e-11
    elif species == 'N2H+':
        comp_fracs = [1.0] # only 1 collision partner in HCO data
        abundance = 1.e-10
    elif species == 'SiO': # is seems to be unusually slow
        comp_fracs = [1.0] # only 1 collision partner in HCO data
        abundance = 7.7e-12
    elif species == 'HNC':
        comp_fracs = [1.0] # only 1 collision partner in HCO data
        abundance = 3.1e-10
    elif species == 'HCN':
        comp_fracs = [1.0, 0.0] # H2 and e
        abundance = 3.1e-11
    elif species == 'CO':
        comp_fracs = [0.66, 0.33] # para H2 and orhto H2
        #abundance = 1.e-4
        abundance = 1.0 # dummy for filenames
    else:
        print 'ERROR: Unsupported species'
        exit()

    return comp_fracs, abundance

''' use 6 ray technique to get local external radiation field
    rho in kg/m3
    dx in m'''
def local_UV_rad_field(rad_field_out, rho, i, j, k, dx):
    # calc column density in 6 directions from the cell
    rho_col = np.zeros(6)
    rho_col[0] = np.sum(rho[i,j,k+1:])
    rho_col[1] = np.sum(rho[i,j,:k-1])
    rho_col[2] = np.sum(rho[i,j+1:,k])
    rho_col[3] = np.sum(rho[i,:j-1,k])
    rho_col[4] = np.sum(rho[i+1:,j,k])
    rho_col[5] = np.sum(rho[:i-1,j,k])
    # take the minimum as approximation for the biggest contribution of
     incoming radiation
    rho_col = np.min(rho_col) * dx #kg/m2
    kappa_dust = 1.e3 #cm2/g assume solar metallicity
    kappa_dust = kappa_dust * 0.1 #m2/kg
    return rad_field_out * np.exp(-kappa_dust*rho_col)

def get_rad_field(rho, dx):
    rad_field_solar = 1.0 #??? this is solar units?
    (nx, ny, nz) = rho.shape
    rad_field = np.zeros((nx,ny,nz))
    for i in range(nx):
        for j in range(ny):
            for k in range(nz):
                rad_field[i][j][k] = local_UV_rad_field(rad_field_solar,
                 rho, i, j, k, dx)
    return rad_field

''' Z is metallicity in solar units
    n_H is density of the cell in H/cc
    rad_field_UV is the local radiation field in the cell
    returns H2 density in H2/cc '''
def calc_local_H2(n_H, Z, rad_field_UV):
    k2 = 2.5e-17 #cm3 s-1
    rate_H2 = 1.7e-11 * rad_field_UV #s-1
    #print rate_H2, n_H, k2, Z
    factor = rate_H2/(2*n_H*k2*Z)
    X_H2 = 0.5 * 1./(1.+factor)
    return n_H * X_H2 #H2/cc

def get_H2_density(rho, rad_field):
    (nx, ny, nz) = rho.shape
    n_H = rho * 1.e-3 / mH_cgs #H/cc
    n_H2 = np.zeros((nx,ny,nz))
    Z = 1. # solar metallicity
    n_H2 = calc_local_H2(n_H, Z, rad_field) #H2/cc
    return n_H2 * 1.e6 #H2/m3

''' rad_field_out in solar radiation field'''
def calc_local_CO(n_H, n_H2, rad_field):
    rate_CHX = 5.e-10 * rad_field
    rate_CO = 1.e-10 * rad_field
    x0 = 2.e-4
    k0 = 5.e-16 #cm3 s-1
    k1 = 5.e-10 #cm3 s-1
    factor_beta = rate_CHX/(n_H*k1*x0)
    beta = 1./(1.+factor_beta)
    factor_CO = rate_CO/(n_H2*k0*beta)
    X_CO = 1./(1.+factor_CO)
    abundance_Ctot = 1e-4 # n_C/n_H as defined by nucleosynthesis
    return n_H * abundance_Ctot * X_CO # CO/cc

def get_CO_density(rho, grid_n_H2, rad_field):
    (nx, ny, nz) = rho.shape
    n_H = rho * 1.e-3 / mH_cgs #H/cc
    n_H2 = grid_n_H2 * 1e-6 #H2/cc
    n_CO = np.zeros((nx,ny,nz))
```

```
print n_CO.shape, n_H.shape, n_H2.shape, rad_field.shape
for i in range(nx):
    for j in range(ny):
        for k in range(nz):
            n_CO[i][j][k] = calc_local_CO(n_H[i][j][k], n_H2[i][j][k],
             rad_field[i][j][k])#CO/cc
return n_CO * 1.e6 #CO/m3
```