```
#==========================================================================
#!/usr/bin/python
#
# Python script function to calculate
# the minimum thickness of the gas layer beneath the impacting drop.
#
# Article title: The skating of drops impacting over gas or vapor layers
# Authors: Paula Garcia-Geijo, Guillaume Riboux, Jos\'e Manuel Gordillo
# Journal: Journal of Fluid Mechanics
# Date: 2023, October
#
# References:
# [1] J. M. Gordillo & G. Riboux. 2022 The initial impact of drops cushioned by an air
 or vapour layer with applications to the dynamic Leidenfrost regime, J. Fluid Mech.,
941, A10:1--19.
# [2] Zhang, Peng & Law, Chung K. 2011 An analysis of head-on droplet collision with l
arge deformation in gaseous medium. Physics of Fluids 23 (4), 042102.
# [3] Sharipov, Felix, Cumin, Liliana M. Gramani & Kalempa, Denize. 2007 Heat flux bet
ween parallel plates through a binary gaseous mixture over the whole range of the knud
sen number. Physica A: Statistical Mechanics and its Applications 378 (2), 183â€“193.

#==========================================================================
import numpy as np
#==========================================================================
#
# Inputs:
# prefac (scalar): Prefactor for the hm theoretical expression
# Ts (scalar)    : Solid temperature (degree)
# V (array)      : Impacting drop velocity (m/s)
# icas (integer) : icas=1 (Capillary regime) - icas=2 (Inertial regime)
#
# Output:
# hth (array)    : Theoretical minimum thickness of the gas layour beneath the drop (m
).
# xi (array)     : Value of the variable xi_bar=taus*xi (-), see eq. (3.16).
#                  In this case xi_bar is normalized by the value of taus
#                  in the isothermal case =12.4
#
#===========================================================================
#===========================================================================
def calcul_rhov(rhov0,rhol,T,V,tau,St):
#
# Function to evaluate the overpressure and vapor density
# see equation (D.1) in [1]
#
#===========================================================================

    # Pressure ratio pg/p0 with p0=patm=1e5 and pg=p0+Dpm, see eq. (3.5)
    pg_p0=(1.+rhol*(V**2.)*(3.*St**(2./3.)/(8*tau*1e5)));

    # Definition of the vapor density as function of temperature
    rhov=rhov0*((273+78)/(273+T))*pg_p0;

    return(pg_p0,rhov);

#===========================================================================
def calcul_taus(CC,betas,St):
#
# Determination of the solution of equation (4.13) for tau_start=taus
#
#===========================================================================

    # Equation (4.13)
    aux=(CC**(5./2.))-(12.4**(3./2.))*(CC+betas*(St**(-1./3.)));
```

```
    # Condition in the case there is not solution
    index=np.where(aux<=0.0);
    if len(index[0])>0:
       Caux=CC[index[0][-1]];
    else:
       Caux=float('nan');

    # Asignement of the solution of the equation Eq. (4.13) for taus
    taus=Caux;

    return(taus);

#===========================================================================
def htheoriq_expr(prefac,icas,tau,St,We,y,R):
#
# Solution of the equation (3.17) in the article.
# prefac (scalar): Prefactor for the hm theoretical expression
# If icas==1 - capillary regime
# If icas==2 - inertial regime
#
#===========================================================================

    if icas==1:
       htheoriq=prefac*R*(tau**(2./3.))*(y**(2./3.))*(St**(-10./9.))*(We**(-1./3.));

    if icas==2:
       htheoriq=prefac*R*tau*(St**(-7./6.))*(y**(1./2.));

    return(htheoriq);

#=============================================================================
def htheoriq_gke(prefac,icas,lamb,tau,St,We,DT,L,rhol,rhov,cpv,Prv,R,muv,mua,htheoriq)
:
#
# Function to calculate the solution of the equation (3.17) taking into account
# the gas kinetic effect with equation (4.4)-(4.12). See also ref. [2] and [3]
#
#==============================================================================

    # Minimum relative difference value of haux and htheoriq for the while loop
    dhthmin=0.001;

    # Initialization of haux and dhth
    haux=htheoriq;
    dhth=100;

    while dhth > dhthmin:
          # Calcul of the Knudsen number with the last value of haux
          Kn=lamb/haux;

          # Calcul of the vapor viscosity (Pa.s) - Gas kinetic effect
          # see ref. [2] and also eq. (4.9)
          muv_kn=muv/(1+6.0966*Kn+0.965*Kn*Kn+0.6967*Kn*Kn*Kn);

          # Calcul of beta and betas - Gas kinetic effect
          # see ref. [3] and eq. (4.12)
          beta=((1./Prv)*(cpv*DT)/L)
          betas=beta*(rhol/rhov)*(muv/mua)/(1.+3.91*Kn);

          # Analitical expression for "y", see Eq. (4.6)
          if DT==0:
             betas=0.0;

          y=3.*(muv_kn/mua)*(1.+np.sqrt(1.+(2./3.)*(mua/muv_kn)*betas));
```

```
        # Expression for htheoriq, see eq. (3.17)
        htheoriq=htheoriq_expr(prefac,icas,tau,St,We,y,R);

        # Calculate the difference of haux
        dhth=100*abs(haux-htheoriq)/haux;

        # New iteration for htheoriq
        haux=htheoriq;

        # Condition in the case the solution is smaller
        # than the mean free path of air
        if htheoriq<70e-9:
            htheoriq=float('NaN');
            break;

    return(htheoriq,y);

#===========================================================================
#
#                     MAIN FUNCTION TO CALCULATE hm
#
#===========================================================================
#
def calcul_hth(prefac,Ts,V,icas):
#
#===========================================================================

    Ta=25; # Ambient temperature definition (degree)
    p0=1e5;# Atmospheric Pressure (Pa)

    # Physical properties in the case of ethanol (liquid)
    #-----------------------------------------------------
    R=1.1e-3;     # Impacting drop radius (m)
    rhol=789;     # Liquid drop density (kg/m^3)
    sigma=0.017; # Superficial tension (liquid drop - air) (N/m)
    kl=0.167;     # Liquid thermal conductivity (W/(m.K))
    mul=1.07e-3; # Drop liquid dynamic viscosity (Pa.s)
    L=853e3;      # Latent Heat coefficient (J/kg)
    Tb=78;        # Boiling temperature (degree)
    Prv=0.96;     # Vapor Prandtl number (-)
    Pr=8;         # Prandtl number (-)

    # The gas properties are evaluated at temperature
    # which is the mean of the boiling temperature of ethanol
    # and the solid temperature
    #-------------------------------------------------------------------
    T=0.5*(Tb+Ts);
    DT=Ts-Tb;  # Solid-boiling difference temperature (degree)

    # Physical properties of the ethanol vapor
    #------------------------------------------
    lambv0=50e-9; # Mean free path of ethanol vapor at p_atm (m)
    rhov0=1.43;   # Vapor density (kg/m^3) appearing in the Eq. (D.1) in ref. [1]
    muv0=1.03e-5;
    muv=muv0*(((273+T)/(273+78))**1.2) # Dynamic viscosity of the vapor (Pa.s)
                                       # in Eq. (D.4) in ref. [1]
    cpv=(1830+250*(T-78)/100); # Heat capacity of the vapor (J/(kg. K))
                               # see Eq. (D.3) in ref. [1]
    kv=0.021;                  # Thermal conductivity of the vapor (W/(m.K))

    # Physical properties of the air
    #--------------------------------
    mua0=1.846e-5;
    mua=mua0*(((T+273)/300)**0.7) # Dynamic viscosity of the air (Pa.s)
                                  # in Eq. (D.5) in ref. [1]
```

```
    lamba0=69e-9; # Mean free path of air (m) at p_atm=pa and T=Ta.

    # Condition for the isothermal case
    if Ts==Ta:
       DT=0;
       mua=mua0;
       muv=mua;
       tau=12.4;    # Value of the constant tau, see ref. [1]
       sigma=0.022; # Superficial tension (liquid drop - air) (N/m)

    # Initialization of hth and xi variables
    xi_bar=float('nan')*np.ones((len(V)));
    hth=float('nan')*np.ones((len(V)));

    # Loops to calculate the minimum gas thickness
    # for each impacting velocity V values and fixed solid Temperature Ts
    #
    #-----------------------------------------------------------------------
    for i in range(len(V)):

        St=rhol*V[i]*R/mua;        # Stokes number for each impact drop velocity
        We=rhol*V[i]*V[i]*R/sigma; # Weber number for each impact drop velocity

        #-------------------------------------------------------------------
        # Isothermal cas - Ts=Ta - DT=0
        #-------------------------------------------------------------------
        if DT==0:

           # Calcul of pg_p0 at tau=12.4
           [pg_p0,rhov]=calcul_rhov(rhov0,rhol,T,V[i],tau,St);

           # 1/ Solution of the equation (3.11) with tau=taus and Knudsen Kn=0
               #    In the case Ts=Ta=25 degree, y=6 and tau=12.4
           #-------------------------------------------------------------
           betas=0.0;
           y=3.*(muv/mua)*(1+np.sqrt(1+(2./3.)*(mua/muv)*betas));

           # Expression for htheoriq (see Eq. (3.17) in the article)
           htheoriq0=htheoriq_expr(prefac,icas,tau,St,We,y,R);

           # 2/ Solution of the equation (3.17) with tau=taus and Knudsen Kn>0
           #    where the gas kinetic effect were taken into account.
           #    With the value of htheoriq0, we calculate the new value of Kn
           #    and converge to the solution haux with a while loop for each
           #    impact velocity V
           #-------------------------------------------------------------
           lamb=lamba0*(1./pg_p0); # Mean free path (m)
                                  # function of the pressure ratio pg/p0

           [htheorique,y]=htheoriq_gke(prefac,icas,lamb,tau,St,We,DT,L,rhol,rhov,cpv,P
rv,R,muv,mua,htheoriq0);

        #-------------------------------------------------------------------
        # Leidenfrost - Ts>Ta - DT>0
        #-------------------------------------------------------------------
        else:
           # 1/ Determination of taus - equation (4.13)
           #-----------------------------------------

           # Initialization of the variable CC~taus
           # for the numerical resolution of equation (4.13)
           # The values of taus should be closer to taus=12.4
           CC=np.array(np.arange(10,20,0.1));

           # Calcul of pg_p0, rhov and betas,
```

```
            # see eq. (4.4) in the article and eq. (D.1) in ref. [1]
            [pg_p0,rhov]=calcul_rhov(rhov0,rhol,T,V[i],CC,St);
            beta=((1./Prv)*(cpv*DT)/L);
            betas=beta*(rhol/rhov)*(muv/mua);

            # Calcul of taus eq. (4.13)
            taus=calcul_taus(CC,betas,St);

            # 2/ Solution of the equation (3.17) with tau=taus and Knudsen Kn=0
            #-----------------------------------------------------------------
            tau=taus;

            # Calcul of pg_p0, rhov, betas at taus obtained before
            [pg_p0,rhov]=calcul_rhov(rhov0,rhol,T,V[i],tau,St);
            beta=((1./Prv)*(cpv*DT)/L)
            betas=((1./Prv)*(cpv*DT)/L)*(rhol/rhov)*(muv/mua);

            # Analitical expression for "y" (see Eq. (4.6))
            y=3.*(muv/mua)*(1.+np.sqrt(1.+(2./3.)*(mua/muv)*betas));

            # Expression for htheoriq
            # (see Eq. (3.17) with y and taus obtained before)
            htheoriq0=htheoriq_expr(prefac,icas,tau,St,We,y,R);

            # 3/ Solution of the equation (3.17) with tau=taus and Knudsen Kn>0
            #    where the gas kinetic effect were taken into account.
            #    With the value of htheoriq0, we calculate the new value of Kn
            #    and converge to the solution haux with a while loop for each
            #    impact velocity V
            #-----------------------------------------------------------------
            lamb=lambv0*(1/pg_p0)*((273+T)/(273+Ta)); # Mean free path (m)
                                     # as function of the temperature T and
                                     # the pressure ratio pg/p0

            [htheorique,y]=htheoriq_gke(prefac,icas,lamb,tau,St,We,DT,L,rhol,rhov,cpv,P
rv,R,muv,mua,htheoriq0);

    #-----------------------------------------------------------------------------
        # Final asignement of the solution for hth and xi
        # as function of the impacting drop velocity V and solid temperature Ts
    #-----------------------------------------------------------------------------
        xi_bar[i]=(tau/12.4)*We*(St**(-1./6.))*(y**(-1./2.));
        hth[i]=htheorique;

    return(xi_bar,hth);

#=========================================================================
```