Titre: Title:	Pheromones impregnated on porous PLA fibers for integrated pest management. Supplément
Auteurs: Authors:	Darius Klassen, Catherine Pouchet, William Simon, Isabella Smith, Noémie Lemoine, Bruno Blais, Mikaël Larose, Gérald Chouinard, Adya Karthikeyan, Marie-Josée Dumont, & Jason Robert Tavares
Date:	2026
Type:	Article de revue / Article
Référence: Citation:	Klassen, D., Pouchet, C., Simon, W., Smith, I., Lemoine, N., Blais, B., Larose, M., Chouinard, G., Karthikeyan, A., Dumont, M.-J., & Tavares, J. R. (2026). Pheromones impregnated on porous PLA fibers for integrated pest management. Journal of Polymers and the Environment, 34(5), 99 (16 pages). https://doi.org/10.1007/s10924-026-03827-1

Pheromones impregnated on porous PLA fibers for integrated pest management (IPM)

Journal of Polymer and the Environment

Darius Klassen[a], Catherine Pouchet[c], William Simon[a], Isabella Smith[a], Noémie Lemoine[c], Bruno Blais[a,d], Mikaël Larose[c], Gérald Chouinard[c], Adya Karthikeyan[e], Marie-Josée Dumont[b], Jason R. Tavares[a*]

[a]CREPEC, Department of Chemical Engineering Polytechnique Montréal, Montreal, Canada
[b]CREPEC, Department of Chemical Engineering, Université Laval, Quebec, Canada
[c]Research and development institute for the agri-environment (IRDA), Saint-Bruno-de-Montarville, Canada
[d]CHAOS laboratory, Department of Chemical Engineering Polytechnique Montréal, Montreal, Canada
[e]Department of Chemical and Biological Engineering, University of Ottawa, Ottawa, Ontario K1N 6N5, Canada

*Co-corresponding author: jason.tavares@polymtl.ca

Supplementary information: Python codes for Model

N-dodecane release prediction constant Windspeed

```
import numpy as np
import matplotlib.pyplot as plt
import scipy

def capillary_function1(X): # Rate of loss without degradation
    import math
    rate=-1.11*math.log(1-X) #units of rate are [1/h]
    return rate

def capillary_function2(X): # Rate of loss without degradation
    import math
    rate=0.6*(1-X) #units of rate are [1/h]
    return rate

# Calculate diffusivity for two gases at atmospheric pressure (p = 1 atm)
def diffusivity(T,P,MA,MB,EA,Tc,GA,Pc):
    import math
    GB = 2.44*math.pow(Tc/Pc,1/3) # approximate collision diameter of evaporating liquid from critical temperature and pressure
    EB = 0.77*Tc # approximate attractive energy of evaporating liquid from critical temperature
```

```
    GAB = 0.5*(GA+GB) # calculate average collision diameter
    EAB = math.sqrt(EA*EB) # calculate average maximum attractive energy
between the two molecules
    tAB = T/EAB # dimensionless term
    Omega =
(1.06036/math.pow(tAB,0.15610))+(0.19300/math.exp(0.47635*tAB))+(1.03587/math.ex
p(1.52996*tAB))+(1.76474/math.exp(3.89411*tAB)) # collision integral solution
    DAB=0.0018583*math.pow(T,3/2)*math.sqrt((1/MA)+(1/MB))*(1/(math.pow(GA
B,2)*Omega*P)) # calculate diffusivity
    DAB = DAB/(100*100) #convert to m2/s
    return DAB
    print()


# calculate viscosity using Sutherland's formula
def Sutherland(T):
    import math
    mu_ref = 1.716*math.pow(10,-5)
    T_ref = 273
    Suth = 111 # Sutherland's constant for air
    mu = mu_ref*math.pow(T/T_ref,3/2)*((T_ref+Suth)/(T+Suth))
    return mu # Dynamic viscosity (N*s/m2)
    print()
```

```
# calculate Reynolds number for air (density=1.1614 kg/m3 viscosity=184.6*10^-7
N*s/m2 at 300K)
def Reynold(V,L,T):
        import math
        density = 1.1614 # density of air
        viscosity = Sutherland(T)
        re = density*V*L/viscosity
        return re
        print()

# calculate schmidt number for air (kinematic viscosity=15.89*10^-6 m2/s at 300k)
def schmidt(DAB,T):
        import math
        viscosity = Sutherland(T)
        density = 1.1614 # density of air
        v = viscosity/density # kinematic viscosity
        sc=v/DAB
        return sc
        print()
```

```
# calculate sherwood number for flow over a flat plate where the flowing fluid is air and
pr>0.2
# Experimentally, with no wind the mass loss due only to natural convection corresponds
to a sherwood number of 14.5
def sherwood(V,L,DAB,T):
        import math
        re = Reynold(V,L,T)
        sc = schmidt(DAB,T)
        if re<=2000:
                sh = 14.5+0.65*math.pow(re,1/2)*math.pow(sc,1/3)
        else:
                sh = 14.5+0.428*math.pow(re,0.574)*math.pow(sc,1/3)
        #Sherwood correlation for vertical plate taken from Incopera and DeWitt p.411
        return sh
        print()


# calculate mass transfer coefficent due to convection (cm/s)
def mtc(V,L,T,P,MA,MB,EA,Tc,GA,Pc):
        import math
        DAB = diffusivity(T,P,MA,MB,EA,Tc,GA,Pc)
        sh = sherwood(V,L,DAB,T)
        hm = sh*DAB/L
        return hm
```

```
    print()

# calculate total mass transfer over the whole surface (mass fraction/h)
def MassTrans(Xnd,P,V,L,T,MA,MB,EA,GA,Tc,Pc):
    import math
    Cmol = Xnd*(P/(8.205736*math.pow(10,-5)*T)) # convert mol fraction to
mol/m3 using ideal gas law
    Cg = Cmol*MB # convert from mol/m3 to g/m3
    hm = mtc(V,L,T,P,MA,MB,EA,Tc,GA,Pc)
    Q1 = hm*Cg
    Q2 = Q1*math.pi*math.pow(L/2,2)
    Q3 = Q2*60*60 # convert from mass g/s to mass g/h
    return Q3
    print()

# calculate latent heat of EBF at given temperature
def LatentH(T,Tboil,Tc,MB):
    import math
    Hvap =
4.1868*Tboil*(9.08+4.36*math.log10(Tboil)+0.0068*Tboil/MB+0.0009*math.pow(Tboi
l,2)/MB)
    # Vetere approximation for latent heat of hydrocarbons at their boiling point
    Tr = T/Tc
```

```
        Trb = Tboil/Tc
        TR = (1-Tr)/(1-Trb)
        Hvap = Hvap*math.pow(TR,0.38)
        # Watson equation correction for latent heat at given temperature
        return Hvap
        print()


# calculate mass loss rate of n-dodecane
def Rate(L,MA,MB,EA,GA,Tc,Pc,SA,Vkmh):
        import math
        P = 1 # atmospheric pressure in atm
        Time = 0 # Start time step at O hours
        HexCodes =
['#ffffcc','#ffeda0','#fed976','#feb24c','#fd8d3c','#fc4e2a','#e31a1c','#b10026']
        Temp = [299, 301, 303, 305, 307, 309, 311, 313]
        Mass_dodecane =
[0.178093333,0.163433333,0.181313333,0.18408,0.157893333,0.16504,0.166766667,0.
163446667] # starting mass n-dodecane
        Experimental=[0.13368,0.097486667,0.050586667,0.03352,0.02832,0.02258666
7,0.019053333,0.01752,0.015353333,0.013553333,0.009053333]#experimental mass
loss data
        ExpTime = [0,1,2,3,4,5,6,7,8,9,24] # time in hours
```

```
    for x in range (0,8):
        T = Temp[x]
        Tboil = 489 # boiling point of n-dodecane (K)
        Pboil  = 1 # pressure equal to atmospheric pressure at boiling point
        LatentHeat = LatentH(T,Tboil,Tc,MB)
        R = 8.314
        clausius = -1*(LatentHeat/R)*(1/T-1/Tboil) # calculate maximum vapor
fraction n-dodecane with Clausius Clapeyron equation
        Xnd = Pboil*math.exp(clausius)# atm
        V = Vkmh/3.6 # convert from km/h to m/s
        mass = [Mass_dodecane[x]]
        percent = [100]
        Time = [ExpTime[0]]
        import os.path # create path to text file
        save_path = 'C:/Users/Darius'
        name_of_file = "masslossrate_"+str(Temp[x])+"K.txt"
        name_of_file = os.path.join(save_path, name_of_file)
        open(name_of_file,"w").close()
        count = 0
        end = len(ExpTime)-1
        while mass[count]>0 and Time[count]<=ExpTime[end]:
            f = open(name_of_file, "a") # write value to text file
            f.write(str(mass[count]))
```

```
                    f.write('\n')
                    f.close()
                    loss1 = MassTrans(Xnd,P,V,L,T,MA,MB,EA,GA,Tc,Pc)# calculate
mass loss rate in g/h
                    check = mass[count]/SA
                    inflection = 12/(100*100)   #12 g/m2 to g/cm2
                    if count>=1:
                        loss = loss1*capillary_function1(mass[count]/mass[0])
                        mass1 = mass[count]-loss*(Time[count]-Time[count-1])
                        finalloss=loss
                    if check/inflection<1:
                        loss=finalloss*capillary_function2(mass[count]/mass[0])
                        mass1 = mass[count]-loss*(Time[count]-Time[count-1])
                    if count<1:
                        mass1 = mass[count]-loss1
                    Time.append(Time[count]+1) # increase time step by 1 hour
                    mass.append(mass1)
                    percent.append(100*(mass1/Mass_dodecane[x]))
                    count = count+1
                xdata = Time
                ydata = percent
                plt.plot(xdata, ydata, HexCodes[x], linestyle='-',label=str(Temp[x]-273) +
' C')
```

```
            plt.xlabel('Time (h)')
            plt.ylabel('Mass retention n-dodecane (%)')
            plt.xscale('log', base=10)
        #plt.plot(ExpTime, Experimental,'x',label='Experimental 299K')
        #plt.title('Model predictions for n-dodecane evaporation at different
temperatures')
        #plt.legend()
        plt.show()


Rate(0.0382,28.9647,170.33,97,3.617,658.2,17.864,26.27,2.2)



# velocity (V) of air in fumehood is 105 ft/min or 18 km/h
# velocity (V) of air in environmental chamber is 2.2 km/h (225 cfm fan will recirculate
all air in 0.05m3 box approximately twice every second)
# length of flat disk (L) 0.0382 m
# Pressure (P) is 1 atm
# Molar masses or air (MA) and n-dodecane (MB) 28.9647 g/mol and 170.33 g/mol
respectively
# maximum attractive energy or air (EA) is 97 K
# Average collision diameter of air (GA) id 3.617 A
# critical temperature (Tc) and pressure (Pc) of n-dodecane are 658.2 K and 18.1 bar
(17.864 atm) respectively
```

 # mass of disk is 3.2 g (massdisk)

 # initial mass of n-dodecane on disk is 0.146273 g (mass)

 # mass of n-dodecane below which all n-dodecane must be in pores is 0.001269 g/cm2 (Vcrit) based on max uptake tests on disks with excess n-dodecane removed

 # step size for time (Time) is in seconds

 # SA is the surface area of the sample in cm2 (26.27 cm2)

N-dodecane release prediction constant Temperature

```
import numpy as np
import matplotlib.pyplot as plt
import scipy

def capillary_function1(X): # Rate of loss without degradation
    import math
    rate=-1.11*math.log(1-X) #units of rate are [1/h]
    return rate

def capillary_function2(X): # Rate of loss without degradation
    import math
    rate=0.6*(1-X) #units of rate are [1/h]
    return rate
```

```
# Calculate diffusivity for two gases at atmospheric pressure (p = 1 atm)
def diffusivity(T,P,MA,MB,EA,Tc,GA,Pc):
    import math
    GB = 2.44*math.pow(Tc/Pc,1/3) # approximate collision diameter of evaporating liquid from critical temperature and pressure
    EB = 0.77*Tc # approximate attractive energy of evaporating liquid from critical temperature
    GAB = 0.5*(GA+GB) # calculate average collision diameter
    EAB = math.sqrt(EA*EB) # calculate average maximum attractive energy between the two molecules
    tAB = T/EAB # dimensionless term
    Omega = (1.06036/math.pow(tAB,0.15610))+(0.19300/math.exp(0.47635*tAB))+(1.03587/math.exp(1.52996*tAB))+(1.76474/math.exp(3.89411*tAB)) # collision integral solution
    DAB=0.0018583*math.pow(T,3/2)*math.sqrt((1/MA)+(1/MB))*(1/(math.pow(GAB,2)*Omega*P)) # calculate diffusivity
    DAB = DAB/(100*100) #convert to m2/s
    return DAB
    print()


# calculate viscosity using Sutherland's formula
def Sutherland(T):
```

```
    import math
    mu_ref = 1.716*math.pow(10,-5)
    T_ref = 273
    Suth = 111 # Sutherland's constant for air
    mu = mu_ref*math.pow(T/T_ref,3/2)*((T_ref+Suth)/(T+Suth))
    return mu # Dynamic viscosity (N*s/m2)
    print()


# calculate Reynolds number for air (density=1.1614 kg/m3 viscosity=184.6*10^-7
N*s/m2 at 300K)
def Reynold(V,L,T):
    import math
    density = 1.1614 # density of air
    viscosity = Sutherland(T)
    re = density*V*L/viscosity
    return re
    print()


# calculate schmidt number for air (kinematic viscosity=15.89*10^-6 m2/s at 300k)
def schmidt(DAB,T):
    import math
    viscosity = Sutherland(T)
```

```
    density = 1.1614 # density of air
    v = viscosity/density # kinematic viscosity
    sc=v/DAB
    return sc
    print()

# calculate sherwood number for flow over a flat plate where the flowing fluid is air and pr>0.2
# Experimentally, with no wind the mass loss due only to natural convection corresponds to a sherwood number of 14.5
def sherwood(V,L,DAB,T):
    import math
    re = Reynold(V,L,T)
    sc = schmidt(DAB,T)
    if re<=2000:
        sh = 14.5+0.65*math.pow(re,1/2)*math.pow(sc,1/3)
    else:
        sh = 14.5+0.428*math.pow(re,0.574)*math.pow(sc,1/3)
    #Sherwood correlation for vertical plate taken from Incopera and DeWitt p.411
    return sh
    print()

# calculate mass transfer coefficent due to convection (cm/s)
```

```
def mtc(V,L,T,P,MA,MB,EA,Tc,GA,Pc):
    import math
    DAB = diffusivity(T,P,MA,MB,EA,Tc,GA,Pc)
    sh = sherwood(V,L,DAB,T)
    hm = sh*DAB/L
    return hm
    print()

# calculate total mass transfer over the whole surface (mass fraction/h)
def MassTrans(Xnd,P,V,L,T,MA,MB,EA,GA,Tc,Pc):
    import math
    Cmol = Xnd*(P/(8.205736*math.pow(10,-5)*T)) # convert mol fraction to
mol/m3 using ideal gas law
    Cg = Cmol*MB # convert from mol/m3 to g/m3
    hm = mtc(V,L,T,P,MA,MB,EA,Tc,GA,Pc)
    Q1 = hm*Cg
    Q2 = Q1*math.pi*math.pow(L/2,2)
    Q3 = Q2*60*60 # convert from mass g/s to mass g/h
    return Q3
    print()

# calculate latent heat of EBF at given temperature
def LatentH(T,Tboil,Tc,MB):
```

```
    import math
    Hvap =
4.1868*Tboil*(9.08+4.36*math.log10(Tboil)+0.0068*Tboil/MB+0.0009*math.pow(Tboi
l,2)/MB)
    # Vetere approximation for latent heat of hydrocarbons at their boiling point
    Tr = T/Tc
    Trb = Tboil/Tc
    TR = (1-Tr)/(1-Trb)
    Hvap = Hvap*math.pow(TR,0.38)
    # Watson equation correction for latent heat at given temperature
    return Hvap
    print()



# calculate mass loss rate of n-dodecane
def Rate(L,MA,MB,EA,GA,Tc,Pc,SA):
    import math
    P = 1 # atmospheric pressure in atm
    Time = 0 # Start time step at O hours
    T = 299
    Vkmh = [0,1.3,1.6,1.8,4,5.8,8.3,12.5,16] # wind speed in km/h
```

```
HexCodes =
['#fff7fb','#ece2f0','#d0d1e6','#a6bddb','#67a9cf','#3690c0','#02818a','#016c59','#014636
']
Mass_dodecane = [0.155,0.155,0.155,0.155,0.155,0.155,0.155,0.155,0.155] #
starting mass n-dodecane
ExpTime = [0,1,2,3,4,5,6,7,8,9,24] # time in hours
for x in range (0,len(Mass_dodecane)):
V = Vkmh[x]
Tboil = 489 # boiling point of n-dodecane (K)
Pboil  = 1 # pressure equal to atmospheric pressure at boiling point
LatentHeat = LatentH(T,Tboil,Tc,MB)
R = 8.314
clausius = -1*(LatentHeat/R)*(1/T-1/Tboil) # calculate maximum vapor
fraction n-dodecane with Clausius Clapeyron equation
Xnd = Pboil*math.exp(clausius)# atm
V = V/3.6 # convert from km/h to m/s
mass = [Mass_dodecane[x]]
percent = [100]
Time = [ExpTime[0]]
import os.path # create path to text file
save_path = 'C:/Users/Darius'
name_of_file = "masslossrate_"+str(Vkmh[x])+"K.txt"
name_of_file = os.path.join(save_path, name_of_file)
```

```
        open(name_of_file,"w").close()
        count = 0
        end = len(ExpTime)-1
        while mass[count]>0 and Time[count]<=ExpTime[end]:
            f = open(name_of_file, "a") # write value to text file
            f.write(str(mass[count]))
            f.write('\n')
            f.close()
            loss1 = MassTrans(Xnd,P,V,L,T,MA,MB,EA,GA,Tc,Pc)# calculate
mass loss rate in g/h
            check = mass[count]/SA
            inflection = 12/(100*100)   #12 g/m2 to g/cm2
            if count>=1:
                loss = loss1*capillary_function1(mass[count]/mass[0])
                mass1 = mass[count]-loss*(Time[count]-Time[count-1])
                finalloss=loss
            if check/inflection<1:
                loss=finalloss*capillary_function2(mass[count]/mass[0])
                mass1 = mass[count]-loss*(Time[count]-Time[count-1])
            if count<1:
                mass1 = mass[count]-loss1
            Time.append(Time[count]+1) # increase time step by 1 hour
            mass.append(mass1)
```

```
                percent.append(100*(mass1/Mass_dodecane[x]))
                count = count+1
            xdata = Time
            ydata = percent
            plt.plot(xdata, ydata, HexCodes[x], linestyle='-',label=str(Vkmh[x]) + '
km/h')
            plt.xlabel('Time (h)')
            plt.ylabel('Mass retention n-dodecane (%)')
            plt.xscale('log', base=10)
        #plt.plot(ExpTime, Mass_experimental,'x',label='Experimental 2.2 km/h')
        #plt.legend()
        plt.show()

Rate(0.0382,28.9647,170.33,97,3.617,658.2,17.864,26.27)


# velocity (V) of air in fumehood is 105 ft/min or 18 km/h
# velocity (V) of air in environmental chamber is 2.2 km/h (225 cfm fan will recirculate
all air in 0.05m3 box approximately twice every second)
# length of flat disk (L) 0.0382 m
# Pressure (P) is 1 atm
# Molar masses or air (MA) and n-dodecane (MB) 28.9647 g/mol and 170.33 g/mol
respectively
```

```
# maximum attractive energy or air (EA) is 97 K
# Average collision diameter of air (GA) id 3.617 A
# critical temperature (Tc) and pressure (Pc) of n-dodecane are 658.2 K and 18.1 bar
(17.864 atm) respectively
# mass of disk is 3.2 g (massdisk)
# initial mass of n-dodecane on disk is 0.146273 g (mass)
# mass of n-dodecane below which all n-dodecane must be in pores is 0.001269 g/cm2
(Vcrit) based on max uptake tests on disks with excess n-dodecane removed
# step size for time (Time) is in seconds
# SA is the surface area of the sample in cm2 (26.27 cm2)
```

EBF release prediction constant Windspeed

```
import numpy as np
import matplotlib.pyplot as plt
import scipy

def capillary_function1(X): # Rate of loss without degradation
    import math
    rate=-1.11*math.log(1-X) #units of rate are [1/h]
    return rate

def capillary_function2(X): # Rate of loss without degradation
    import math
```

```
    rate=0.6*(1-X) #units of rate are [1/h]
    return rate

def degredation_function(Time): # Rate of loss without degradation
    import math
    rate=5.33/Time #units of rate are [1/h]
    return rate

# Calculate diffusivity for two gases at atmospheric pressure (p = 1 atm)
def diffusivity(T,P,MA,MB,EA,Tc,GA,Pc):
    import math
    GB = 2.44*math.pow(Tc/Pc,1/3) # approximate collision diameter of evaporating liquid from critical temperature and pressure
    EB = 0.77*Tc # approximate attractive energy of evaporating liquid from critical temperature
    GAB = 0.5*(GA+GB) # calculate average collision diameter
    EAB = math.sqrt(EA*EB) # calculate average maximum attractive energy between the two molecules
    tAB = T/EAB # dimensionless term
    Omega = (1.06036/math.pow(tAB,0.15610))+(0.19300/math.exp(0.47635*tAB))+(1.03587/math.exp(1.52996*tAB))+(1.76474/math.exp(3.89411*tAB)) # collision integral solution
```

```
        DAB=0.0018583*math.pow(T,3/2)*math.sqrt((1/MA)+(1/MB))*(1/(math.pow(GA
B,2)*Omega*P)) # calculate diffusivity
        DAB = DAB/(100*100) #convert to m2/s
        return DAB
        print()


# calculate viscosity using Sutherland's formula
def Sutherland(T):
        import math
        mu_ref = 1.716*math.pow(10,-5)
        T_ref = 273
        Suth = 111 # Sutherland's constant for air
        mu = mu_ref*math.pow(T/T_ref,3/2)*((T_ref+Suth)/(T+Suth))
        return mu # Dynamic viscosity (N*s/m2)
        print()


# calculate Reynolds number for air (density=1.1614 kg/m3 viscosity=184.6*10^-7
N*s/m2 at 300K)
def Reynold(V,L,T):
        import math
        density = 1.1614 # density of air
```

```
    viscosity = Sutherland(T)
    re = density*V*L/viscosity
    return re
    print()

# calculate schmidt number for air (kinematic viscosity=15.89*10^-6 m2/s at 300k)
def schmidt(DAB,T):
    import math
    viscosity = Sutherland(T)
    density = 1.1614 # density of air
    v = viscosity/density # kinematic viscosity
    sc=v/DAB
    return sc
    print()

# calculate sherwood number for flow over a flat plate where the flowing fluid is air and
pr>0.2
# Experimentally, with no wind the mass loss due only to natural convection corresponds
to a sherwood number of 14.5
def sherwood(V,L,DAB,T):
    import math
    re = Reynold(V,L,T)
    sc = schmidt(DAB,T)
```

```
    if re<=2000:
        sh = 14.5+0.65*math.pow(re,1/2)*math.pow(sc,1/3)
    else:
        sh = 14.5+0.428*math.pow(re,0.574)*math.pow(sc,1/3)
    #Sherwood correlation for vertical plate taken from Incopera and DeWitt p.411
    return sh
    print()


# calculate mass transfer coefficent due to convection (cm/s)
def mtc(V,L,T,P,MA,MB,EA,Tc,GA,Pc):
    import math
    DAB = diffusivity(T,P,MA,MB,EA,Tc,GA,Pc)
    sh = sherwood(V,L,DAB,T)
    hm = sh*DAB/L
    return hm
    print()


# calculate total mass transfer over the whole surface (mass fraction/h)
def MassTrans(Xnd,P,V,L,T,MA,MB,EA,GA,Tc,Pc):
    import math
    Cmol = Xnd*(P/(8.205736*math.pow(10,-5)*T)) # convert mol fraction to
mol/m3 using ideal gas law
    Cg = Cmol*MB # convert from mol/m3 to g/m3
```

```
    hm = mtc(V,L,T,P,MA,MB,EA,Tc,GA,Pc)
    Q1 = hm*Cg
    Q2 = Q1*math.pi*math.pow(L/2,2)
    Q3 = Q2*60*60 # convert from mass g/s to mass g/h
    return Q3
    print()

# calculate latent heat of EBF at given temperature
def LatentH(T,Tboil,Tc,MB):
    import math
    Hvap =
4.1868*Tboil*(9.08+4.36*math.log10(Tboil)+0.0068*Tboil/MB+0.0009*math.pow(Tboi
l,2)/MB)
    # Vetere approximation for latent heat of hydrocarbons at their boiling point
    Tr = T/Tc
    Trb = Tboil/Tc
    TR = (1-Tr)/(1-Trb)
    Hvap = Hvap*math.pow(TR,0.38)
    # Watson equation correction for latent heat at given temperature
    return Hvap
    print()
```

```
# calculate mass loss rate of n-dodecane
def Rate(L,MA,MB,EA,GA,Tc,Pc,SA,Vkmh):
    import math
    P = 1 # atmospheric pressure in atm
    Time = 0 # Start time step at O hours
    HexCodes =
['#ffffcc','#ffeda0','#fed976','#feb24c','#fd8d3c','#fc4e2a','#e31a1c','#b10026']
    Temp = [299, 301, 303, 305, 307, 309, 311, 313]
    Mass_EBF = [0.22,0.22,0.22,0.22,0.22,0.22,0.22,0.22] # starting masses of EBF
    Experimental=[0.162833333,0.112933333,0.0474,0.035433333,0.028733333,0.0
263,0.018966667,0.014666667,0.011933333,0.0081,0.0083,0.0073,0.0073]#experimenta
l mass loss data
    ExpTime = [24,48,144,168,192,216,240,312,384,408,528,552,576] # time in
hours
    for x in range (0,8):
        T = Temp[x]
        Tboil = 516.43 # Experimental boiling point of EBF (K)
        Pboil  = 1 # pressure equal to atmospheric pressure at boiling point
        LatentHeat = LatentH(T,Tboil,Tc,MB)
        R = 8.314
        clausius = -1*(LatentHeat/R)*(1/T-1/Tboil) # calculate maximum vapor
fraction n-dodecane with Clausius Clapeyron equation
        print('clausius', clausius)
```

```
                    Xnd = Pboil*math.exp(clausius)# atm
                    #print('Vapor fraction at',T,Xnd)
                    V = Vkmh/3.6 # convert from km/h to m/s
                    mass = [Mass_EBF[x]]
                    percent = [100]
                    Time = [ExpTime[0]]
                    import os.path # create path to text file
                    save_path = 'C:/Users/Darius'
                    name_of_file = "masslossrate_"+str(Temp[x])+"K.txt"
                    name_of_file = os.path.join(save_path, name_of_file)
                    open(name_of_file,"w").close()
                    count = 0
                    end = len(ExpTime)-1
                    while mass[count]>0 and Time[count]<=ExpTime[end]:
                            f = open(name_of_file, "a") # write value to text file
                            f.write(str(mass[count]))
                            f.write('\n')
                            f.close()
                            check = mass[count]/SA
                            inflection = 12/(100*100)   #12 g/m2 to g/cm2
                            loss1 = MassTrans(Xnd,P,V,L,T,MA,MB,EA,GA,Tc,Pc)# calculate
mass loss rate in g/h
                            if count>=1:
```

```
                    loss =
loss1*capillary_function1(mass[count]/mass[0])*degredation_function(Time[count])
                    mass1 = mass[count]-loss*(Time[count]-Time[count-1])
                    finalloss=loss
                if check/inflection<1:

        loss=finalloss*capillary_function2(mass[count]/mass[0])*degredation_function(
Time[count])
                    mass1 = mass[count]-loss*(Time[count]-Time[count-1])
                if count<1:
                    mass1 = mass[count]-loss1
                Time.append(Time[count]+1) # increase time step by 1 hour
                mass.append(mass1)
                percent.append(100*(mass1/Mass_EBF[x]))
                count = count+1
            xdata = Time
            ydata = percent
            plt.plot(xdata, ydata, HexCodes[x], linestyle='-',label=str(Temp[x]-273) +
' C')
            plt.xlabel('Time (h)')
            plt.ylabel('Mass retention EBF (%)')
            plt.xscale('log', base=10)
        #plt.plot(ExpTime, Mass_experimental,'x',label='Experimental 2.2 km/h')
```

```
        #plt.legend()
        plt.show()


Rate(0.0382,28.9647,204.35,97,3.617,730.22,16.37,26.27,2.2)



# velocity (V) of air in fumehood is 105 ft/min or 18 km/h
# velocity (V) of air in environmental chamber is 0.6145 m/s or 2.2 km/h (225 cfm fan
will recirculate all air in 0.05m3 box approximately twice every second)
# length of flat disk (L) 0.0382 m
# Pressure (P) is 1 atm
# Molar masses or air (MA) and EBF (MB) 28.9647 g/mol and 204.35 g/mol respectively
# maximum attractive energy or air (EA) is 97 K
# Average collision diameter of air (GA) id 3.617 A
# critical temperature (Tc) and pressure (Pc) of EBF are 730.22 K and 16.70 bar (16.37
atm) respectively
# critical volume of EBF (Vc) is 0.8 m3/kmol
# mass of disk is 3.2 g (massdisk)
# initial mass of EBF on disk is 0.157389748 g (mass)
# mass of n-dodecane below which all EBF must be in pores is 0.001365444 g/cm2
(Vcrit) based on max uptake tests on disks with excess n-dodecane removed
# step size for time (Time) is in hours
# SA is the surface area of the sample in cm2 (26.27cm2)
```

EBF release prediction constant Temperature

```
import numpy as np
import matplotlib.pyplot as plt
import scipy

def capillary_function1(X): # Rate of loss without degradation
    import math
    rate=-1.11*math.log(1-X) #units of rate are [1/h]
    return rate

def capillary_function2(X): # Rate of loss without degradation
    import math
    rate=0.6*(1-X) #units of rate are [1/h]
    return rate

def degredation_function(Time): # Rate of loss without degradation
    import math
    rate=5.33/Time #units of rate are [1/h]
    return rate

# Calculate diffusivity for two gases at atmospheric pressure (p = 1 atm)
def diffusivity(T,P,MA,MB,EA,Tc,GA,Pc):
```

```
    import math
    GB = 2.44*math.pow(Tc/Pc,1/3) # approximate collision diameter of evaporating
liquid from critical temperature and pressure
    EB = 0.77*Tc # approximate attractive energy of evaporating liquid from critical
temperature
    GAB = 0.5*(GA+GB) # calculate average collision diameter
    EAB = math.sqrt(EA*EB) # calculate average maximum attractive energy
between the two molecules
    tAB = T/EAB # dimensionless term
    Omega =
(1.06036/math.pow(tAB,0.15610))+(0.19300/math.exp(0.47635*tAB))+(1.03587/math.ex
p(1.52996*tAB))+(1.76474/math.exp(3.89411*tAB)) # collision integral solution
    DAB=0.0018583*math.pow(T,3/2)*math.sqrt((1/MA)+(1/MB))*(1/(math.pow(GA
B,2)*Omega*P)) # calculate diffusivity
    DAB = DAB/(100*100) #convert to m2/s
    return DAB
    print()


# calculate viscosity using Sutherland's formula
def Sutherland(T):
    import math
    mu_ref = 1.716*math.pow(10,-5)
```

```
    T_ref = 273
    Suth = 111 # Sutherland's constant for air
    mu = mu_ref*math.pow(T/T_ref,3/2)*((T_ref+Suth)/(T+Suth))
    return mu # Dynamic viscosity (N*s/m2)
    print()


# calculate Reynolds number for air (density=1.1614 kg/m3 viscosity=184.6*10^-7
N*s/m2 at 300K)
def Reynold(V,L,T):
    import math
    density = 1.1614 # density of air
    viscosity = Sutherland(T)
    re = density*V*L/viscosity
    return re
    print()


# calculate schmidt number for air (kinematic viscosity=15.89*10^-6 m2/s at 300k)
def schmidt(DAB,T):
    import math
    viscosity = Sutherland(T)
    density = 1.1614 # density of air
    v = viscosity/density # kinematic viscosity
```

```
    sc=v/DAB
    return sc
    print()

# calculate sherwood number for flow over a flat plate where the flowing fluid is air and
pr>0.2
# Experimentally, with no wind the mass loss due only to natural convection corresponds
to a sherwood number of 14.5
def sherwood(V,L,DAB,T):
    import math
    re = Reynold(V,L,T)
    sc = schmidt(DAB,T)
    if re<=2000:
        sh = 14.5+0.65*math.pow(re,1/2)*math.pow(sc,1/3)
    else:
        sh = 14.5+0.428*math.pow(re,0.574)*math.pow(sc,1/3)
    #Sherwood correlation for vertical plate taken from Incopera and DeWitt p.411
    return sh
    print()

# calculate mass transfer coefficent due to convection (cm/s)
def mtc(V,L,T,P,MA,MB,EA,Tc,GA,Pc):
    import math
```

```
    DAB = diffusivity(T,P,MA,MB,EA,Tc,GA,Pc)
    sh = sherwood(V,L,DAB,T)
    hm = sh*DAB/L
    return hm
    print()

# calculate total mass transfer over the whole surface (mass fraction/h)
def MassTrans(Xnd,P,V,L,T,MA,MB,EA,GA,Tc,Pc):
    import math
    Cmol = Xnd*(P/(8.205736*math.pow(10,-5)*T)) # convert mol fraction to
mol/m3 using ideal gas law
    Cg = Cmol*MB # convert from mol/m3 to g/m3
    hm = mtc(V,L,T,P,MA,MB,EA,Tc,GA,Pc)
    Q1 = hm*Cg
    Q2 = Q1*math.pi*math.pow(L/2,2)
    Q3 = Q2*60*60 # convert from mass g/s to mass g/h
    return Q3
    print()

# calculate latent heat of EBF at given temperature
def LatentH(T,Tboil,Tc,MB):
    import math
```

```
        Hvap =
4.1868*Tboil*(9.08+4.36*math.log10(Tboil)+0.0068*Tboil/MB+0.0009*math.pow(Tboi
l,2)/MB)
        # Vetere approximation for latent heat of hydrocarbons at their boiling point
        Tr = T/Tc
        Trb = Tboil/Tc
        TR = (1-Tr)/(1-Trb)
        Hvap = Hvap*math.pow(TR,0.38)
        # Watson equation correction for latent heat at given temperature
        return Hvap
        print()



# calculate mass loss rate of n-dodecane
def Rate(L,MA,MB,EA,GA,Tc,Pc,SA):
        import math
        P = 1 # atmospheric pressure in atm
        Time = 0 # Start time step at O hours
        T = 299
        Vkmh = [0,1.3,1.6,1.8,4,5.8,8.3,12.5,16] # wind speed in km/h
        HexCodes =
['#fff7fb','#ece2f0','#d0d1e6','#a6bddb','#67a9cf','#3690c0','#02818a','#016c59','#014636
']
```

```
Mass_EBF = [0.22,0.22,0.22,0.22,0.22,0.22,0.22,0.22,0.22] # starting mass n-dodecane
Mass_experimental=[0.1514,0.0726,0.0481,0.0373,0.033133333,0.029133333,0.023,0.019,0.01695,0.011533333,0.011466667,0.009866667,0.009866667]#experimental mass loss data 2.2 km/h
ExpTime = [24,48,144,168,192,216,240,312,384,408,528,552,576] # time in hours
for x in range (0,len(Mass_EBF)):
        V = Vkmh[x]
        Tboil = 516.43 # Experimental boiling point of EBF (K)
        Pboil  = 1 # pressure equal to atmospheric pressure at boiling point
        LatentHeat = LatentH(T,Tboil,Tc,MB)
        R = 8.314
        clausius = -1*(LatentHeat/R)*(1/T-1/Tboil) # calculate maximum vapor fraction n-dodecane with Clausius Clapeyron equation
        Xnd = Pboil*math.exp(clausius)# atm
        V = V/3.6 # convert from km/h to m/s
        mass = [Mass_EBF[x]]
        Time = [ExpTime[0]]
        percent = [100]
        import os.path # create path to text file
        save_path = 'C:/Users/Darius'
        name_of_file = "masslossrate_"+str(Vkmh[x])+"K.txt"
```

```
        name_of_file = os.path.join(save_path, name_of_file)
        open(name_of_file,"w").close()
        count = 0
        end = len(ExpTime)-1
        while mass[count]>0 and Time[count]<=ExpTime[end]:
            f = open(name_of_file, "a") # write value to text file
            f.write(str(mass[count]))
            f.write('\n')
            f.close()
            check = mass[count]/SA
            inflection = 12/(100*100)   #12 g/m2 to g/cm2
            loss1 = MassTrans(Xnd,P,V,L,T,MA,MB,EA,GA,Tc,Pc)# calculate
mass loss rate in g/h
            if count>=1:
                loss =
loss1*capillary_function1(mass[count]/mass[0])*degredation_function(Time[count])
                mass1 = mass[count]-loss*(Time[count]-Time[count-1])
                finalloss=loss
            if check/inflection<1:

        loss=finalloss*capillary_function2(mass[count]/mass[0])*degredation_function(
Time[count])
                mass1 = mass[count]-loss*(Time[count]-Time[count-1])
```

```
                    if count<1:
                        mass1 = mass[count]-loss1
                    Time.append(Time[count]+1) # increase time step by 1 hour
                    mass.append(mass1)
                    percent.append(100*(mass1/Mass_EBF[x]))
                    count = count+1
                xdata = Time
                ydata = percent
                plt.plot(xdata, ydata, HexCodes[x], linestyle='-',label=str(Vkmh[x]) + '
km/h')
                plt.xlabel('Time (h)')
                plt.ylabel('Mass retention EBF (%)')
                plt.xscale('log', base=10)
        #plt.plot(ExpTime, Mass_experimental,'x',label='Experimental 2.2 km/h')
        #plt.legend()
        plt.show()



Rate(0.0382,28.9647,204.35,97,3.617,730.22,16.37,26.27)


# velocity (V) of air in fumehood is 105 ft/min or 18 km/h
# velocity (V) of air in environmental chamber is 0.6145 m/s or 2.2 km/h (225 cfm fan
will recirculate all air in 0.05m3 box approximately twice every second)
```

 # length of flat disk (L) 0.0382 m

 # Pressure (P) is 1 atm

 # Molar masses or air (MA) and EBF (MB) 28.9647 g/mol and 204.35 g/mol respectively

 # maximum attractive energy or air (EA) is 97 K

 # Average collision diameter of air (GA) id 3.617 A

 # critical temperature (Tc) and pressure (Pc) of EBF are 730.22 K and 16.70 bar (16.37 atm) respectively

 # critical volume of EBF (Vc) is 0.8 m3/kmol

 # mass of disk is 3.2 g (massdisk)

 # initial mass of EBF on disk is 0.157389748 g (mass)

 # mass of n-dodecane below which all EBF must be in pores is 0.001365444 g/cm2 (Vcrit) based on max uptake tests on disks with excess n-dodecane removed

 # step size for time (Time) is in hours

 # SA is the surface area of the sample in cm2 (26.27cm2)