Deep-sea Settling of Plastic Debris

Original Code for Microplastic Vertical Transport Model

Xiangfei Sun, Eddy Y. Zeng\*, Mengyi Xie, Lei Mai, Yaoqiu Kuang

\*Corresponding author. E-mail: [eddyzeng@jnu.edu.cn](mailto:eddyzeng@jnu.edu.cnx)

*r"""*

*This model primaryly estimates the vertical transport velocity for microplastics*

*December 7th, 2021*

*Code by Xiangfei Sun, Jinan University*

*Please see the associated paper:*

*Sun X., Zeng E.Y., Xie M., Mai L., Song X.*

*2022*

*Microplastic Deposition in Ocean*

*Science*

*"""*

***# Functional Package Import***

import os

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import random

import re

import copy

import math

import netCDF4

#from mpl\_toolkits.basemap import Basemap

import numpy.ma as ma

from pylab import \*

import csv

import random

import copy

from scipy.interpolate import make\_interp\_spline

***# Data file reader***

# .xlsx reader

def readexcel(file\_name):

data = pd.read\_excel(file\_name)

train\_data = np.array(data) # np.ndarray()

excel\_list = train\_data.tolist() # list

return excel\_list

# .csv reader

def csvimport(file\_name):

smiles = []

with open(file\_name,'r',encoding='UTF-8') as csvfile:

csv\_reader = csv.reader(csvfile)

for row in csv\_reader:

smiles.append(row)

smiles[0][0] = 'No.'

return(smiles)

def csvimport1(file\_name):

smiles = []

with open(file\_name) as csvfile:

csv\_reader = csv.reader(csvfile)

for row in csv\_reader:

smiles.append(row)

smiles[0][0] = 'No.'

return(smiles)

*r"""*

*The functions are defined for data import from .csv file (csvimport and csvimport1) and .xlsx file (readexcel)*

*"""*

***# Random size generator***

def exponential\_rand(lam):

if lam <= 0:

return -1

u = random.random()

sign = 1

while sign == 1:

try:

value = (-1.0/lam) \* np.log(u)

sign = 0

except:

sign = 1

return value

***# Gravity Adjustment according to Latitude***

def local\_g(lat,depth):

A = 0.0053024

B = 0.0000058

C = 3.086e-06

lat\_value = float(lat[:-1])

loc\_g = 9.780327\*((1+A\*(math.sin(lat\_value))\*\*2-B\*(math.sin(2\*lat\_value))\*\*2)-C\*(-depth))\*3600\*\*2\*24\*\*2

return(loc\_g)

*r"""*

*Calculates local gravity at given Latitude and water depth.*

*Parameters*

*----------*

*lat : float*

*Latitude in degree*

*depth : float*

*Seawater depth in minus meter*

*Returns*

*-------*

*loc\_g, the local gravity in m2/day*

*References*

*----------*

*International Gravity Formula, 2008, A Dictionary of Earth Sciences (3 ed.),*

*Oxford University Press.*

*"""*

***# Seawater Density according to depth, salinity, and temperature***

def water\_density(t,S):

c1 = 9.999e02

c2 = 2.034e-02

c3 = -6.162e-03

c4 = 2.261e-05

c5 = -4.657e-08

b1 = 8.020e02

b2 = -2.001

b3 = 1.677e-02

b4 = -3.060e-05

b5 = -1.613e-05

den\_f = c1 + c2\*t + c3\*t\*\*2 + c4\*t\*\*3 + c5\*t\*\*4 + b1\*S + b2\*S\*t + b3\*S\*t\*\*2 + b4\*S\*t\*\*3 + b5\*(S\*\*2)\*(t\*\*2)

value = den\_f/1000

return(value)

*r"""*

*Calculates seawater density at given temperature and salinity*

*using Eq. (8) given by Sharqawy et. al [1]. Values at temperature higher*

*than the normal boiling temperature are calculated at the saturation*

*pressure.*

*Parameters*

*----------*

*t : float*

*Temperature must be in Celsius for this emperical equation to work*

*S : float*

*Salinity must be expressed in kg of salt per kg of solution (ppt).*

*Returns*

*-------*

*value, the density of water/seawater in [kg/m3]*

*Notes*

*-----*

*T must be in C, and S in g of salt per kg of phase, or ppt (parts per thousand)*

*VALIDITY: 0 < T < 180 C; 0 < S < 0.16 kg/kg;*

*ACCURACY: 0.1 %*

*References*

*----------*

*[1] Sharqawy M. H., Lienhard J. H., and Zubair, S. M., Desalination and*

*Water Treatment, 2010.*

*"""*

***# Microplastic Shape, Size, and Weight***

# 1. Sphere

def MP\_sphere(r\_pl):

V\_pl = 4\*pi\*(r\_pl\*\*3)/3

S\_pl = 4\*pi\*(r\_pl\*\*2)

return (V\_pl, S\_pl)

r"""

Calculates dimensions, sizes, and volume of microplastics in near sphere shape.

Parameters

----------

r\_pl : float

The radius of original microplastic particle near to sphere shape in m.

Returns

-------

V\_pl: float

The volume of original microplastic particle near to sphere shape in m3.

S\_pl: float

The surface area of original microplastic particle near to sphere shape in m2.

*Notes*

*----------*

*The microplastic particle, which is close to sphere shape, is assumed as a perfect*

*sphere to simplify the analysis. This function can be altered or adjusted for an*

*irregular sphere. However, it may require a simultaneous change for the shape factor*

*estimation according to real-time measurements or modeling assumptions.*

*"""*

# 2. Film:

def MP\_film(r\_pl,t\_pl):

V\_pl = pi\*(r\_pl\*\*2)\*t\_pl

S\_pl = 2\*pi\*(r\_pl\*\*2) + 2\*pi\*r\_pl\*t\_pl

return (V\_pl, S\_pl)

*r"""*

*Calculates dimensions, sizes, and volume of microplastics in film shape.*

*Parameters*

*----------*

*r\_pl : float*

*The radius of the surface of original microplastic film in m.*

*t\_pl : float*

*The thickness of original microplastic film in m.*

*Returns*

*-------*

*V\_pl: float*

*The volume of original microplastic film in m3.*

*S\_pl: float*

*The surface area of original microplastic film in m2.*

*Notes*

*----------*

*The microplastic particle, which is close to fibrous shape, is assumed as a perfect*

*long cylinder to simplify the analysis. This function can be altered or adjusted for an*

*irregular or uneven cross-section. However, it may require a simultaneous change for the*

*shape factor estimation according to real-time measurements or modeling assumptions.*

*"""*

# 3. Fiber:

def MP\_fiber(r\_pl,l\_pl):

V\_pl = pi\*(r\_pl\*\*2)\*l\_pl

S\_pl = 4\*pi\*(r\_pl\*\*2) + 2\*pi\*r\_pl\*l\_pl

return (V\_pl, S\_pl)

*r"""*

*Calculates dimensions, sizes, and volume of microplastics fiber.*

*Parameters*

*----------*

*r\_pl : float*

*The radius of the cross section of original microplastic fiber in m.*

*l\_pl : float*

*The thickness of original microplastic in film shape in m.*

*Returns*

*-------*

*V\_pl: float*

*The volume of original microplastic fiber in m3.*

*S\_pl: float*

*The surface area of original microplastic fiber in m2.*

*Notes*

*----------*

*The microplastic particle, which is close to thin film shape, is assumed as an even thickness*

*and a circular shape to simplify the analysis. This function can be altered or adjusted for an*

*irregular or uneven thickness. However, it may require a simultaneous change for the*

*shape factor estimation according to real-time measurements or modeling assumptions.*

*"""*

***# The Enconunter Kernel Rate***

def collision(d\_ESD,T,k,r\_A,w\_v,Wt,shear):

# 1 Bronian motion

D\_pl = k\*(T+273.16)/(6\*pi\*w\_v\*d\_ESD\*0.5)

D\_A = k\*(T+273.16)/(6\*pi\*w\_v\*r\_A)

Beta\_BM = 4\*pi\*(D\_pl+D\_A)\*(r\_A+d\_ESD/2)

# 2 Differential settling

Beta\_DS = 0.5\*pi\*(0.5\*d\_ESD)\*\*2\*Wt

# 3 Advective shear

Beta\_AS = 1.3\*shear\*(0.5\*d\_ESD+r\_A)\*\*3

Beta\_A = Beta\_BM + Beta\_DS + Beta\_AS

return Beta\_A

*r"""*

*Calculates the particle/algae encounter kernel rate at given temperature, seawater viscosity,*

*veritical speed, and shear using reference [1], [2], and [3].*

*Parameters*

*----------*

*T : float*

*The seawater temperature at given depth in C.*

*k : float*

*The Boltzmann constant in m2 kg d-2 K-1.*

*w\_v : float*

*The seawater viscosity at given depth in kg m-1 d-1.*

*Wt : float*

*The vertical transport speed at given depth in m d-1.*

*shear: float*

*The shear rate in day-1*

*Returns*

*-------*

*Beta\_A: float*

*The enconunter kernal rate.*

*References*

*----------*

*[1] Jackson, G. A., A model of the formation of marine algal flocs by physical coagulation processes.*

*Deep-Sea Res. I: Oceanogr. Res. 1990, 37, 1197-1211.*

*[2] Burd, A. B.; Jackson, G. A., Particle aggregation. Annu. Rev. Mar. Sci. 2009, 1, 65-90.*

*[3] Farley, K. J.; Morel, F. M. M., Role of coagulation in the kinetics of sedimentation. Environ.*

*Sci. Technol. 1986, 20, 187-195.*

*"""*

***# Selecting parameters for surface Chl\_a concentration***

def position\_chl\_a(Latitude,Longtiude,Lat\_bins,Lon\_bins,dataset):

try:

# Latitude Reposition

identifier = Latitude[-1]

lat\_value = float(Latitude[:-1])

if identifier == 'S':

lat\_value = -1\*lat\_value

elif identifier == 'N':

lat\_value = lat\_value

else:

print('The lattitude input is wrong!!')

# Longtitude Reposition

identifier = Longtitude[-1]

lon\_value = float(Longtitude[:-1])

if identifier == 'E':

lon\_value = lon\_value

elif identifier == 'W':

lon\_value = -1\*lon\_value

else:

print('The longtitude input is wrong!!')

posi\_lat = np.digitize(lat\_value, Lat\_bins) - 1

posi\_lon = np.digitize(lon\_value, Lon\_bins) - 1

chl\_a = dataset[posi\_lat,posi\_lon]

except:

print("The given position is on land! Calculation terminates!")

return(chl\_a)

*r"""*

*Find the surface Chl\_a concentraiton at given latitude and longtitude*

*using reference [1] provided by NASA.*

*Parameters*

*----------*

*Latitude : string*

*The latitude of the given position in degrees and orientation sign.*

*Longtitude : string*

*The longtitude of the given position in degrees and orientation sign.*

*Lat\_bins : float*

*The latitude of given surface chlorophyll-a profile in degrees.*

*Lon\_bins : float*

*The longtitude of given surface chlorophyll-a profile in degrees.*

*dataset: float*

*The surface chlorophyll-a concentration matrix in mg m-3.*

*Returns*

*-------*

*chl\_a: float*

*The surface chlorophyll-a concentration in mg m-3.*

*References*

*----------*

*[1] NASA, Chlorophyll concentration, NASA earth observatory,*

*https://earthobservatory.nasa.gov/global-maps/MY1DMM\_CHLORA.*

*(accessed Feb 2, 2022)*

*"""*

***# The chlorophyll-a concentration***

def Chloro\_a(parameter\_list, chl\_a\_bins, chl\_a, z):

posi\_chl\_a = np.digitize(chl\_a, chl\_a\_bins) - 1

C\_b = parameter\_list[posi\_chl\_a][0]

s = parameter\_list[posi\_chl\_a][1]

C\_max = parameter\_list[posi\_chl\_a][2]

Z\_max = parameter\_list[posi\_chl\_a][3]

delta\_z = parameter\_list[posi\_chl\_a][4]

dep\_eupz = parameter\_list[posi\_chl\_a][9]

Chl\_a\_base = parameter\_list[posi\_chl\_a][5]

Chl\_a = (C\_b-s\*z/dep\_eupz+C\_max\*exp(-((z-Z\_max)/delta\_z)\*\*2))\*Chl\_a\_base

if Chl\_a < 0:

Chl\_a = 0

return Chl\_a

*r"""*

*Calculates the chlorophyll-a concentration at given depth based on references [1] and [2].*

*Parameters*

*----------*

*C\_b : float*

*The normalized surface concentration.*

*s : float*

*The slope in m-1.*

*z: float*

*The seawater depth in m.*

*C\_max: float*

*The normalized maximum concentration.*

*Z\_max: float*

*The depth at which the maximum concentration can be found in m.*

*delta\_z: float*

*The width of the peak in m.*

*Chl\_a\_base: float*

*The average chlorophyll-a concentration of the vertical profile in mg m-3*

*Returns*

*-------*

*Chl\_a: float*

*The chlorophyll-a concentration at depth z in mg m-3.*

*Notes*

*----------*

*Chl\_a concentration must be a non-negative value.*

*References*

*----------*

*[1] Uitz, J.; Claustre, H.; Morel, A.; Hooker, S. B., Vertical distribution of*

*phytoplankton communities in open ocean: An assessment based on surface chlorophyll.*

*J. Geophys. Res. Oceans 2006, 111.*

*[2] Ardyna, M.; Babin, M.; Gosselin, M.; Devred, E.; Bélanger, S.; Matsuoka, A.;*

*Tremblay, J. É., Parameterization of vertical chlorophyll a in the Arctic Ocean:*

*Impact of the subsurface chlorophyll maximum on regional, seasonal, and annual primary*

*production estimates. Biogeosciences 2013, 10, 4383-4404.*

*"""*

***# Algae concentraition at given depth z***

def algae\_c(Chl\_a,T,L\_z,V\_A):

C\_conc = Chl\_a / (0.003+1.0154\*exp(0.05\*T)\*exp(-0.059\*L\_z/10\*\*6))

A\_conc = C\_conc\*10\*\*9 / (0.216\*((V\_A\*10\*\*18)\*\*0.939))

return A\_conc,C\_conc

*r"""*

*Calculates the algae concentration at given depth according to chlorophyll-a concentration*

*and carbon ratio given by reference [1] and [2].*

*Parameters*

*----------*

*Chl\_a : float*

*The vertical chlorophyll-a profile at given depth z in mg m-3.*

*T : float*

*The temperature at given depth in C.*

*L\_z : float*

*The light intensity at given depth z in (uE m-2 day-1).*

*V\_A: float*

*The algae volume in m3.*

*Returns*

*-------*

*C\_conc: float*

*The carbon concentration in mg m-3*

*A\_conc: float*

*The algae concentration at given temperature and light intensity L\_z in cell m-3.*

*References*

*----------*

*[1] Cloern, J. E.; Grenz, C.; Vidergar-Lucas, L., An empirical model of the phytoplankton*

*chlorophyll: carbon ratio-the conversion factor between productivity and growth rate.*

*Limnol. Oceanogr. 1995, 40, 1313-1321.*

*[2] Menden-Deuer, S.; Lessard, E. J., Carbon to volume relationships for dinoflagellates,*

*diatoms, and other protist plankton. Limnol. Oceanogr. 2000, 45, 569-579.*

*"""*

***# Light intensity***

def light\_intensity(z,Chl\_a,o\_w,o\_p,t\_ac,I\_m,Latitude):

o\_t = o\_w + o\_p \* Chl\_a

I\_0 = I\_m \* sin(2\*t\_ac\*pi)

if I\_0 < 0:

I\_0 = 0

L\_z = I\_0 \* exp(-o\_t\*z)

if L\_z < 0:

L\_z = 0

return L\_z, o\_t

*r"""*

*Calculates the light intensity at given depth and time based on the law of Lambert-Beer.*

*Parameters*

*----------*

*z : float*

*Seawater depth in m.*

*Chl\_a : float*

*The vertical chlorophyll-a profile at given depth z in mg m-3.*

*o\_w : float*

*The extinction coefficients for water.*

*o\_p : float*

*The extinction coefficients for chhlorophyll.*

*t: float*

*The time spot.*

*I\_m: float*

*The light intensity at noon (uE m-2 day-1)*

*Returns*

*-------*

*L\_z: float*

*The light intensity at given depth z in (uE m-2 day-1).*

*"""*

***# Algae Growth***

def algae\_growth(L\_z,u\_max,I\_opt,alpha,T,T\_max,T\_min,T\_opt):

u\_opt = u\_max\*L\_z/(L\_z+u\_max\*(L\_z/I\_opt-1)\*\*2/alpha)

fy\_T = ((T-T\_max)\*(T-T\_min)\*\*2)/((T\_opt-T\_min)\*((T\_opt-T\_min)\*(T-T\_opt)-(T\_opt-T\_max)\*(T\_opt+T\_min-2\*T)))

A\_g = u\_opt \* fy\_T

return A\_g

*r"""*

*Calculates the algae growth rate at given depth and temperature based on reference [1].*

*Parameters*

*----------*

*L\_z : float*

*The light intensity at given depth z in uE m-2 day-1.*

*u\_max : float*

*The maximum growth rate under optimal conditions in day-1.*

*I\_opt : float*

*The optimal light intensity for algae growth uE m-2 day-1.*

*alpha : float*

*The initial slope in day-1.*

*T: float*

*The temperature at depth z in C.*

*T\_max: float*

*The maximum temperature to sustain algae growth in C.*

*T\_min: float*

*The minimum temperature to sustain algae growth in C.*

*T\_opt: float*

*The optimal temperature to sustain algae growth in C.*

*Returns*

*-------*

*A\_g: float*

*The algae growth rate in day-1.*

*References*

*----------*

*[1] Bernard, O.; Rémond, B., Validation of a simple model accounting for light and temperature*

*effect on microalgal growth. Bioresour. Technol. 2012, 123, 520-527.*

*"""*

***# The Biofilm development rate***

def biofilm\_growth(S\_pl, A\_g, A\_conc, Beta\_A, m\_A, T, Q\_10, R\_20, A, phy\_att):

B\_g = Beta\_A\*A\_conc/S\_pl\*phy\_att + A\_g\*A - m\_A\*A - Q\_10\*\*((T-20)/10)\*A\*R\_20

return B\_g

*r"""*

*Calculates the biofilm development rate at given depth and temperature.*

*Parameters*

*----------*

*A\_g : float*

*The algae growth rate in day-1.*

*A\_conc : float*

*The algae concentration at given temperature and light intensity L\_z in cell m3.*

*S\_pl : float*

*The surface area of original microplastic film in m2.*

*Beta\_A : float*

*The enconunter kernal rate in m3 day-1.*

*m\_A: float*

*The mortality rate in day-1.*

*T: float*

*The temperature at depth z in C.*

*Q\_10: float*

*The respiration coefficient.*

*R\_20: float*

*The respiration rate of algae in day-1.*

*A: float*

*Current algae attached on microplatic surface in cell m-2.*

*Returns*

*-------*

*B\_g: float*

*The biofilm growth rate in cell m-2 day-1.*

*"""*

***# The biofirm volume***

def bio\_volume(B\_g, V\_A, t, A, S\_pl):

A = A + B\_g \* t

V\_bf = A \* V\_A \* S\_pl

return V\_bf,A

*r"""*

*Calculates the biofilm volume on microplastic at given depth and temperature.*

*Parameters*

*----------*

*V\_A : float*

*The volume of single algae in m3.*

*S\_pl : float*

*The microplastic surface area in m2.*

*A : float*

*The number of algae in cell m-2.*

*V\_M : float*

*The volume of minerals on microplastics particle in m3.*

*t: float*

*The time length in days.*

*Returns*

*-------*

*V\_bf: float*

*The accumulated volume of algae in the biofirm in m3.*

*Notes*

*-------*

*The algae volume is a non-negative value.*

*"""*

***# The mineral growth***

def minal\_volume(Ca\_g,L\_z,I\_m,t,m\_b,dep,V\_A,A,S\_pl):

proj\_S\_A = A \* S\_pl \* ((3 \* V\_A / 4 / pi)\*\*(1/3))\*\*2 \* pi

# Calcite precipitation caused by photosynthesis

if proj\_S\_A < S\_pl:

m\_b\_ph = 0.5\*Ca\_g\*3600\*24\*exp(L\_z/I\_m)/10\*\*6\*100/1000\*proj\_S\_A\*t

else:

m\_b\_ph = 0.5\*Ca\_g\*3600\*24\*exp(L\_z/I\_m)/10\*\*6\*100/1000\*S\_pl\*t

# Calcite precipitation caused by passive deposition

if dep < 3000:

m\_b\_pa = 0\*3600\*24\*t\*S\_pl/10\*\*6\*100/1000\*t

else:

m\_b\_pa = 0

if m\_b\_ph >= m\_b\_pa:

m\_b = m\_b + m\_b\_ph

else:

m\_b = m\_b + m\_b\_pa

# Calcite precipitation based on photosynthesis

V\_M = m\_b/2630

return m\_b, V\_M

*r"""*

*Calculates the calcium precipitation rate on microplastic at given depth based on*

*references [1] -- [4], assuming seafloor is always above the calcite compensation depth.*

*Parameters*

*----------*

*Ca\_g : float*

*The calcite precipitation rate in umol m-2 s-1*

*dep : float*

*The seawater depth in m.*

*o\_t : float*

*The coefficient for light reduction.*

*t: float*

*The time length in days.*

*m\_b : float*

*The previous mineral build-ups in kg.*

*V\_A : float*

*The algae volume in m3.*

*A : float*

*Number of algae attached on microplastic surface in No.*

*Returns*

*-------*

*m\_b : float*

*The accumulated mass of calcite in the biofirm in kg.*

*V\_M : float*

*The voulume current mineral build-ups in m3.*

*References*

*----------*

*[1] Dong, S.; Subhas, A. V.; Rollins, N. E.; Naviaux, J. D.; Adkins, J. F.; Berelson,*

*W. M., A kinetic pressure effect on calcite dissolution in seawater. Geochim.*

*Cosmochim. Acta 2018, 238, 411-423.*

*[2] Šovljanski, O.; Pezo, L.; Tomić, A.; Ranitović, A.; Cvetković, D.; Markov, S.,*

*Contribution of bacterial cells as nucleation centers in microbiologically induced*

*CaCO3 precipitation—A mathematical modeling approach. J. Basic Microbiol.*

*2021, 61, 835-848.*

*[3] Price, N. N.; Hamilton, S. L.; Smith, J. E.; Tootell, J. S., Species-specific*

*consequences of ocean acidification for the calcareous tropical green algae Halimeda.*

*In Mar. Ecol. Prog. Ser., PANGAEA: 2011; Vol. 440, pp 67-78.*

*[4] Hartley, A. M.; House Wa Fau - Leadbeater, B. S. C.; Leadbeater Bsc Fau - Callow, M. E.;*

*Callow, M. E., The use of microelectrodes to study the precipitation of calcite upon algal*

*biofilms. J. Colloid Interface Sci. 1996, 183, 498-505.*

"""

***# Microplastic volume with included biofilm***

def total\_volume(V\_M, V\_bf, V\_pl):

V\_T = V\_pl + V\_bf + V\_M

return V\_T

*r"""*

*Calculates the total volume of microplastic inclusion in m3*

*V\_M : float*

*The volume of minerals on microplastics particle in m3.*

*V\_pl : float*

*The microplastic volume in m3.*

*V\_bf: float*

*The volume of biota on microplastic inclusion in m3.*

*Returns*

*-------*

*V\_T : float*

*The total volume of inclusion in m3.*

*"""*

***# The equavalent sphaerical diameter***

def ESD(V\_T):

d\_ESD = (3\*V\_T/(4\*pi))\*\*(1/3)\*2

return d\_ESD

*r"""*

*Calculates the equavalent sphaerical diameter (ESD) in m.*

*Parameters*

*----------*

*V\_T : float*

*The total volume of inclusion in m3.*

*Returns*

*-------*

*d\_ESD: float*

*The equavalent sphaerical diameter (ESD) in m.*

*References*

*----------*

*[1] Van Melkebeke, M.; Janssen, C.; De Meester, S., Characteristics and sinking behavior of*

*typical microplastics including the potential effect of biofouling: implications for*

*remediation. Environ. Sci. Technol. 2020, 54, 8668-8680.*

*[2] Dioguardi, F.; Mele, D.; Dellino, P., A new one-equation model of fluid drag for irregularly*

*shaped particles valid over a wide range of reynolds number. J. Geophys. Res. Solid Earth,*

*2018, 123, 144-156.*

*"""*

***# The shape factor***

#1. Sphere

def shape\_factor\_sphere():

X = 1

Theta = 1

shape\_f = Theta/X

return shape\_f

#2. Fiber

def shape\_factor\_fiber(V\_T,l\_pl):

r\_T = ((V\_T/l\_pl)/pi)\*\*0.5

S\_p = r\_T\*2\*l\_pl\*pi + 2\*pi\*r\_T\*\*2

r\_sp = (3\*V\_T/(4\*pi))\*\*(1/3)

S\_sp = 4\*pi\*r\_sp\*\*2

Theta = S\_sp / S\_p

P\_p = (r\_T\*2+l\_pl)\*2

S\_proj = 2\*r\_T\*l\_pl

r\_proj = (S\_proj/pi)\*\*0.5

P\_proj = 2\*pi\*r\_proj

X = P\_p/P\_proj

shape\_f = Theta/X

return shape\_f

#3. Film

def shape\_factor\_film(V\_T,r\_pl):

t\_T = V\_T/(pi\*r\_pl\*\*2)

S\_p = r\_pl\*2\*t\_T\*pi + 2\*pi\*r\_pl\*\*2

r\_sp = (3\*V\_T/(4\*pi))\*\*(1/3)

S\_sp = 4\*pi\*r\_sp\*\*2

Theta = S\_sp / S\_p

S\_proj = r\_pl\*\*2\*pi\*0.5

r\_proj = (S\_proj/(pi))\*\*0.5

P\_proj = 2\*pi\*r\_proj

P\_p = 2\*pi\*r\_pl

X = P\_p/P\_proj

shape\_f = Theta/X

return shape\_f

*r"""*

*Calculates the shape factor of microplastic particle during vertical movement.*

*Parameters*

*----------*

*V\_T : float*

*The total volume of inclusion in m3.*

*l\_pl : float*

*The length of fiber in m.*

*t\_pl : float*

*The thickness of film in m.*

*Returns*

*-------*

*Theta : float*

*The sphericity*

*X : float*

*The circularity*

*shape\_f : float*

*The shape factor of the inculusion.*

*Notes*

*----------*

*[1] The sphericity is never greater than 1, being 1 for a perfect sphere.*

*[2] The circularity is never smaller than 1, being 1 for a perfect circular contour.*

*[3] The shape factor is never greater than 1, being 1 for a perfect sphere.*

*[4] The shape factor is specifically designed for the vertical velocity calculation*

*formula of Eq. 14 given in reference [1].*

*References*

*----------*

*[1] Dioguardi, F.; Mele, D.; Dellino, P., A new one-equation model of fluid drag for irregularly*

*shaped particles valid over a wide range of reynolds number. J. Geophys. Res. Solid Earth,*

*2018, 123, 144-156.*

*"""*

***# The microplastic inclusion density***

def mp\_in\_density(V\_T,den\_pl,den\_A,m\_b,V\_bf,V\_pl):

in\_den = (V\_pl\*den\_pl + V\_bf\*den\_A + m\_b)/V\_T

return in\_den

*r"""*

*Calculates the shape factor of vertical movement.*

*Parameters*

*----------*

*V\_T : float*

*The total volume of inclusion in m3.*

*V\_pl : float*

*The microplastic volume in m3.*

*den\_pl : float*

*The miroplastic density in kg/m3.*

*den\_A : float*

*The algae density in kg/m3.*

*m\_b : float*

*The accumulated mineral mass in kg.*

*V\_bf : float*

*The current biofilm volume in m3.*

*Returns*

*-------*

*in\_den : float*

*The microplastic inclusion density in kg m-3.*

*"""*

***# Seawater Viscosity***

def water\_viscosity(pore\_T,pore\_salinity):

T = pore\_T

try:

S = pore\_salinity

except:

S = 0

TC = T

S = S/1000

a1 = 1.5700386464E-01

a2 = 6.4992620050E+01

a3 = -9.1296496657E+01

a4 = 4.2844324477E-05

mu\_w = a4 + 1/(a1\*(TC+a2)\*\*2+a3)

a5 = 1.5409136040E+00

a6 = 1.9981117208E-02

a7 = -9.5203865864E-05

a8 = 7.9739318223E+00

a9 = -7.5614568881E-02

a10 = 4.7237011074E-04

A = a5 + a6\*T + a7\*T\*\*2

B = a8 + a9\*T + a10\*T\*\*2

mu\_sw = mu\_w\*(1 + A\*S + B\*S\*\*2)

value = mu\_sw \* 3600 \* 24

return value

*r"""*

*Calculates viscosity of pure water or seawater at atmospheric pressure*

*using Eq. (22-23) given by Sharqawy et. al [1]. Values at temperature higher*

*than the normal boiling temperature are calculated at the saturation*

*pressure.*

*Parameters*

*----------*

*pore\_temperature : float*

*Temperature must be in Celsius for this emperical equation to work*

*pore\_salinity : float*

*Salinity must be expressed in kg of salt per kg of solution (ppt).*

*Returns*

*-------*

*mu\_sw, the viscosity of water/seawater in kg m-1 day-1.*

*Notes*

*-----*

*T must be in C, and S in g of salt per kg of phase, or ppt (parts per thousand)*

*VALIDITY: 0 < T < 180 C; 0 < S < 150 g/kg;*

*ACCURACY: 1.5 %*

*References*

*----------*

*[1] Sharqawy M. H., Lienhard J. H., and Zubair, S. M., Desalination and*

*Water Treatment, 2010.*

*"""*

***# Vertical settlement velocity solver***

def vertical\_velocity(shape\_f,den\_f,in\_den,g,d\_p,u\_f):

# Set initial tolerance and guess of Reynolds number for solver

tolerance = 1

Re = 100

# Calculating the vertical settlement velocity

if dep > 0:

if in\_den > den\_f:

while tolerance > 0.01:

# Update current Reynolds number

init\_Re = Re

# Calculating partical drag coefficient using Dioguardi et al., 2018

Cd = 24/Re\*((1-shape\_f)/Re+1)\*\*0.25+24/Re\*(0.1806\*Re\*\*0.6459)/(shape\_f\*\*(Re\*\*0.08))+0.4251/(1+6880.95/Re\*shape\_f\*\*5.05)

# Calculating settling velocity

Wt = (4\*g\*(in\_den-den\_f)\*d\_p/(3\*Cd\*den\_f))\*\*0.5

# Calculating Reynolds number

Re = den\_f\*Wt\*d\_p/u\_f

# Estimating the new tolerance value under current result

tolerance = abs(init\_Re-Re)/Re

else:

while tolerance > 0.01:

# Update current Reynolds number

init\_Re = Re

# Calculating partical drag coefficient using Dioguardi et al., 2018

Cd = 24/Re\*((1-shape\_f)/Re+1)\*\*0.25+24/Re\*(0.1806\*Re\*\*0.6459)/(shape\_f\*\*(Re\*\*0.08))+0.4251/(1+6880.95/Re\*shape\_f\*\*5.05)

# Calculating settling velocity

Wt = (4\*g\*(den\_f-in\_den)\*d\_p/(3\*Cd\*den\_f))\*\*0.5

# Calculating Reynolds number

Re = den\_f\*Wt\*d\_p/u\_f

# Estimating the new tolerance value under current result

tolerance = abs(init\_Re-Re)/Re

Wt = -1 \* Wt

else:

if in\_den > den\_f:

while tolerance > 0.01:

# Update current Reynolds number

init\_Re = Re

# Calculating partical drag coefficient using Dioguardi et al., 2018

Cd = 24/Re\*((1-shape\_f)/Re+1)\*\*0.25+24/Re\*(0.1806\*Re\*\*0.6459)/(shape\_f\*\*(Re\*\*0.08))+0.4251/(1+6880.95/Re\*shape\_f\*\*5.05)

# Calculating settling velocity

Wt = (4\*g\*(in\_den-den\_f)\*d\_p/(3\*Cd\*den\_f))\*\*0.5

# Calculating Reynolds number

Re = den\_f\*Wt\*d\_p/u\_f

# Estimating the new tolerance value under current result

tolerance = abs(init\_Re-Re)/Re

else:

Wt = 0

Re = Re

Cd = 0

return(Re, Wt, Cd)

*r"""*

*Calculates settling velocity of microplastics with specific density in seawater Eq. (2),(3),(14)*

*given by Dioguardi et al. 2018*

*Parameters*

*----------*

*shape : float*

*The shape factor is a dimensionless parameter to describe the ratio between sphericity and circularity.*

*den\_f : float*

*The density of seawater in kg m-3.*

*in\_den : float*

*The density of paricles in kg m-3.*

*g : float*

*The gravitational acceleration in m day-2.*

*d\_p: float*

*The particle size in m.*

*u\_f: float*

*The seawater viscosity in Pa\*day.*

*Returns*

*-------*

*Re : float*

*The Reynolds number.*

*Wt : float*

*The particle settling velocity in m day-1.*

*Cd : float*

*The particle drag coefficient.*

*Notes*

*-----*

*R2: 96.0%*

*References*

*----------*

*[1] Dioguardi, F.; Mele, D.; Dellino, P., A new one-equation model of fluid drag for irregularly*

*shaped particles valid over a wide range of reynolds number. J. Geophys. Res. Solid Earth,*

*2018, 123, 144-156.*

*[2] Van Melkebeke, M.; Janssen, C.; De Meester, S., Characteristics and sinking behavior of typical*

*microplastics including the potential effect of biofouling: implications for remediation.*

*Environ. Sci. Technol. 2020, 54, 8668-8680.*

*"""*

def cal\_given\_posi(Latitude,Longtiude,date,tolerance):

*# Latitude Reposition*

identifier = Latitude[-1]

lat\_value = float(Latitude[:-1])

if identifier == 'S':

lat\_value = -1\*lat\_value

elif identifier == 'N':

lat\_value = lat\_value

else:

print('The lattitude input is wrong!!')

*# Longtitude Reposition*

identifier = Longtitude[-1]

lon\_value = float(Longtitude[:-1])

if identifier == 'E':

if lon\_value > 74.16:

lon\_value = lon\_value

else:

lon\_value = lon\_value + 360

elif identifier == 'W':

lon\_value = 360 - lon\_value

else:

print('The longtitude input is wrong!!')

*# Data import*

url = 'F:\RTOFS\Original\_data\\' + date + '\_rtofs\_glo\_3dz\_n024\_daily\_3ztio.nc'

file = netCDF4.Dataset(url)

lat = file.variables['Latitude'][:]

lon = file.variables['Longitude'][:]

dep = file.variables['Depth'][:]

file.close()

for i in range(3298):

for j in range(4500):

lat\_diff = abs(lat[i,j] - lat\_value)

lon\_diff = abs(lon[i,j] - lon\_value)

new\_diff = (lat\_diff\*\*2 + lon\_diff\*\*2)\*\*0.5

if new\_diff < tolerance:

posi\_lat = i

posi\_lon = j

else:

continue

*# Data import*

url = 'F:\RTOFS\Original\_data\\' + date + '\_rtofs\_glo\_3dz\_n024\_daily\_3ztio.nc'

file = netCDF4.Dataset(url)

*# Extract data from given location*

temp = file.variables['temperature'][0,:,posi\_lat,posi\_lon]

file.close()

*# Data import*

url = 'F:\RTOFS\Original\_data\\' + date + '\_rtofs\_glo\_3dz\_n024\_daily\_3zsio.nc'

file = netCDF4.Dataset(url)

*# Extract data from given location*

sat = file.variables['salinity'][0,:,posi\_lat,posi\_lon]

file.close()

Sea\_den = []

for i in range(len(sat)):

try:

new\_point = water\_density(temp.data[i],sat.data[i]/1000)

if new\_point > 0 and new\_point < 1.2:

Sea\_den.append(new\_point)

else:

continue

except:

continue

Sea\_den = np.array(Sea\_den)

return(Sea\_den,sat,temp)

*r"""*

*Localize the given latitude and longtitude on the RTOFS datasets and extract the vertical profiles of*

*seawater temperature and salinity at given date. Calculate the seawater density profiles accordingly.*

*Parameters*

*----------*

*Latitude : string*

*The proposed Latitude in degrees with orientation.*

*Longtitude : string*

*The proposed Longtiude in degrees with orientation.*

*date : float*

*The proposed date.*

*tolerance : float*

*The difference between proposed location and avaliable data points.*

*Returns*

*-------*

*Sea\_den : float*

*The vertical seawater density at given location in kg m-3.*

*sat : float*

*The vertical seawater salinity at given location in ppt.*

*temp : float*

*The vertical seawater temperature at given location in C.*

*"""*

***# Density, Salinity and Temperature at given depth***

def position(dep,bins,Sea\_den,sat,temp):

posi\_depth = max\_depth\_check(sat) - 1

max\_depth = bins[posi\_depth]

if dep < max\_depth:

cats = np.digitize(dep, bins)

posi\_den = Sea\_den[cats] - (Sea\_den[cats] - Sea\_den[cats-1])/(bins[cats] - bins[cats-1])\*(bins[cats]-dep)

posi\_sat = sat[cats] - (sat[cats] - sat[cats-1])/(bins[cats] - bins[cats-1])\*(bins[cats]-dep)

posi\_temp = temp[cats] - (temp[cats] - temp[cats-1])/(bins[cats] - bins[cats-1])\*(bins[cats]-dep)

sign = 0

else:

sign = 0

#print("The maximum depth is apporached, which is: " + str(max\_depth) + 'm at given location.')

posi\_den = Sea\_den[posi\_depth]

posi\_sat = sat[posi\_depth]

posi\_temp = temp[posi\_depth]

dep = max\_depth

return(sign, posi\_den, posi\_sat, posi\_temp, dep)

*r"""*

*Calculate density, sanlinity, and temperature at given depth with last known position and vertical settling velocity.*

*Parameters*

*----------*

*dep : float*

*The current depth in m.*

*Sea\_den : float*

*The vertical profiles of seawater density in kg m-3.*

*sat : float*

*The vertical profiles of seawater salinity in ppt.*

*temp : float*

*The vertical profiles of seawater temperature in ppt.*

*bins : float*

*The given depth in m.*

*Returns*

*-------*

*sign : float*

*The value which indicates whether the particle approaches the seafloor.*

*posi\_den : float*

*The surrounding seawater density at given depth in m.*

*posi\_temp : float*

*The surrounding seawater temperature at given depth in C.*

*posi\_sat : float*

*The surrounding seawater salinity at given depth in ppt.*

*dep : float*

*The fixed depth in m.*

*Notes*

*-------*

*The depth never excceds the maximum avaliable seawater depth at given location.*

*"""*

***# Check maximum depth at given location***

def max\_depth\_check(sat):

count = 0

for i in range(len(sat.data)):

if sat.data[i] < 1.2676506e+30:

count = count + 1

else:

break

return count

*r"""*

*Estimate the maximum seawater depth at given location.*

*"""*

def evaluation\_same\_volume(r\_pl,t\_pl,l\_pl):

*# Given the same size with different shape*

V\_pl = 4/3\*pi\*r\_pl\*\*3

r\_fiber = (V\_pl/l\_pl/pi)\*\*0.5

r\_film = (V\_pl/t\_pl/pi)\*\*0.5

return(r\_fiber,r\_film)

***# Random Particle Generator***

def random\_mp\_pro(shape\_type,Latitude):

base\_inf = []

shape\_p = random.randint(0,2)

shape\_type = shape\_type[shape\_p]

Lat = float(Latitude[:-1])

r\_pl = (exponential\_rand(4.8)\*1000)\*0.000001\*0.5

l\_pl = random.randrange(57,13000)\*0.000001

t\_pl = random.randrange(10,50)\*0.000001

den\_pl = random.randint(900,1000)

#time\_len = random.randint(20,300)

time\_len = -1

while time\_len < 1:

time\_len = int(np.random.normal(200,80,1))

if shape\_type == 'fiber':

r\_fiber,r\_film = evaluation\_same\_volume(r\_pl,t\_pl,l\_pl)

elif shape\_type == 'film':

r\_fiber,r\_film = evaluation\_same\_volume(r\_pl,t\_pl,l\_pl)

if shape\_type == 'sphere':

V\_pl, S\_pl = MP\_sphere(r\_pl)

V\_T = V\_pl

#shape\_f = shape\_factor\_sphere()

shape\_f = 1

elif shape\_type == 'fiber':

V\_pl, S\_pl = MP\_fiber(r\_fiber,l\_pl)

V\_T = V\_pl

shape\_f = shape\_factor\_fiber(V\_T,l\_pl)

elif shape\_type == 'film':

V\_pl, S\_pl = MP\_film(r\_film,t\_pl)

V\_T = V\_pl

shape\_f = shape\_factor\_film(V\_T,r\_pl)

base\_inf.append(shape\_type)

base\_inf.append(Lat)

base\_inf.append(r\_pl)

base\_inf.append(l\_pl)

base\_inf.append(t\_pl)

base\_inf.append(V\_pl)

base\_inf.append(S\_pl)

base\_inf.append(V\_T)

base\_inf.append(shape\_f)

base\_inf.append(den\_pl)

base\_inf.append(time\_len)

return base\_inf

***# Main Program***

***# Positioning***

date = '20211208'

# California

Latitude = '36.8N'

Longtitude = '122.35W'

tolerance = 0.08

Sea\_den,sat,temp = cal\_given\_posi(Latitude,Longtitude,date,tolerance)

***# Surface chlorophyll-a database import***

chl\_a\_file\_name = r"C:\Users\JePhyllis\MP\_Vertical\_Transport\_Model\Chl\_a\_surface concentration.csv"

chl\_a\_file = csvimport(chl\_a\_file\_name)

chl\_a\_file = np.array(chl\_a\_file)

Lon\_bins = chl\_a\_file[0,1:]

Lat\_bins = chl\_a\_file[1:,0]

Lat\_bins = Lat\_bins.astype('float')

Lon\_bins = Lon\_bins.astype('float')

dataset = chl\_a\_file[1:,1:].astype('float')

chl\_a\_cal\_file\_name = r'C:\Users\JePhyllis\MP\_Vertical\_Transport\_Model\chl\_a\_parameter.xlsx'

chl\_a\_cal = readexcel(chl\_a\_cal\_file\_name)

chl\_a\_cal = np.array(chl\_a\_cal)

parameter\_list = chl\_a\_cal

***# Save Path***

path1 = r'E:\暨南大学\碳排放与碳中和\Proposal\Biofouling\Paper\Test Results\Trajectory\\'

***# Plastic Material Properties***

record = []

dep\_record\_all = []

for i in range(6000):

shape\_type = ('sphere','fiber','film')

base\_inf = random\_mp\_pro(shape\_type,Latitude)

record.append(base\_inf)

Ca\_g = 0.2

phy\_att = 0.5

den\_A = 1380

V\_A = 1e-15

m\_A = 0.39

R\_20 = 0.1

k = 1.0306e-013

Q\_10 = 2

u\_max = 1.85

alpha = 0.12

T\_min = 0.2

T\_opt = 26.7

T\_max = 33.3

I\_opt = 1.75392e13

t = 0.02

o\_w = 0.2

o\_p = 0.02

I\_m = 1.2e08

bins = np.array([0.,10.,20.,30.,50.,75.,100.,125.,150.,200.,250.,300.,400.,500.,600.,700.,800.,900.,1000.,1100.,1200.,1300.,1400.,1500.,1750.,2000.,2500.,3000.,3500.,4000.,4500.,5000.,5500.])

shear = 1.7e05

r\_A = (3\*V\_A/(4\*pi))\*\*(1/3)

chl\_a\_bins = np.array([0,0.04,0.08,0.12,0.2,0.3,0.4,0.8,2.2,4])

dep = 0

m\_b = 0

W\_t = 0

V\_bf = 0

V\_M = 0

d\_ESD = ESD(base\_inf[7])

sign, posi\_den, posi\_sat, posi\_temp, dep = position(dep,bins,Sea\_den,sat.data,temp.data)

w\_v = water\_viscosity(posi\_temp,posi\_sat)

Beta\_A = collision(d\_ESD,posi\_temp,k,r\_A,w\_v,W\_t,shear)

A = 0

sign = 0

t\_total = 0

t\_ac = 0

count\_mark = 0

dep\_record = []

time\_lenth = base\_inf[10]

round\_sim = int(time\_lenth/t)

#print('The initial shape factor of given plastic particle is: '+str(base\_inf[8])+'.' )

while sign == 0 and t\_total < round\_sim:

sign, posi\_den, posi\_sat, posi\_temp, dep = position(dep,bins,Sea\_den,sat.data,temp.data)

chl\_a = position\_chl\_a(Latitude,Longtitude,Lat\_bins,Lon\_bins,dataset)

if chl\_a > 4:

chl\_a = 3.9

Chl\_a = Chloro\_a(parameter\_list, chl\_a\_bins, chl\_a, dep)

L\_z, o\_t = light\_intensity(dep,Chl\_a,o\_w,o\_p,t\_ac,I\_m,base\_inf[1])

A\_conc,C\_conc = algae\_c(Chl\_a,posi\_temp,L\_z,V\_A)

A\_g = algae\_growth(L\_z,u\_max,I\_opt,alpha,posi\_temp,T\_max,T\_min,T\_opt)

w\_v = water\_viscosity(posi\_temp,posi\_sat)

Beta\_A = collision(d\_ESD,posi\_temp,k,r\_A,w\_v,W\_t,shear)

B\_g = biofilm\_growth(base\_inf[6], A\_g, A\_conc, Beta\_A, m\_A, posi\_temp, Q\_10, R\_20, A, phy\_att)

m\_b, V\_M = minal\_volume(Ca\_g,L\_z,I\_m,t,m\_b,dep,V\_A, A, base\_inf[6])

V\_bf, A = bio\_volume(B\_g, V\_A, t, A, base\_inf[6])

V\_T = total\_volume(V\_M, V\_bf, base\_inf[5])

m\_pl = base\_inf[5] \* base\_inf[9]

m\_bf = V\_bf \* den\_A

d\_ESD = ESD(base\_inf[7])

if base\_inf[0] == 'sphere':

base\_inf[6] = 4\*(((3\*base\_inf[7]/(4\*pi))\*\*(1/3))\*\*2)\*pi

elif base\_inf[0] == 'fiber':

base\_inf[8] = shape\_factor\_fiber(base\_inf[7],base\_inf[3])

base\_inf[6] = 2\*pi\*((base\_inf[7]/base\_inf[3]\*pi)\*\*(1/2))\*base\_inf[3]

elif base\_inf[0] == 'film':

base\_inf[8] = shape\_factor\_film(base\_inf[7],base\_inf[2])

g = local\_g(Latitude,dep)

in\_den = mp\_in\_density(base\_inf[7],base\_inf[9],den\_A,m\_b,V\_bf,base\_inf[5])

Re, Wt, Cd = vertical\_velocity(base\_inf[8],posi\_den\*1000,in\_den,g,d\_ESD,w\_v)

mark = posi\_den\*1000\*(1+0.02)

if in\_den < mark and int(dep) == 0:

Wt = 0

dep = dep + Wt\*t

posi\_depth = max\_depth\_check(sat) - 1

max\_depth = bins[posi\_depth]

if dep < 0:

dep = 0

Wt = 0

elif dep > max\_depth:

dep = max\_depth

Wt = 0

elif np.isnan(dep):

dep = dep\_record[-1][1] + 0.001

t\_total = t\_total + 1

#if int(t\_total) % int(1/t) == 0:

t\_ac = t\_ac + t

new\_line = []

new\_line.append(t\_ac)

new\_line.append(dep)

dep\_record.append(new\_line)

if int(dep) != 0 and count\_mark == 0:

#print(in\_den,t\_ac)

count\_mark = 1

dep\_record = np.array(dep\_record)

dep\_record\_all.append(dep\_record[-1][1])

plastic\_type = []

for i in range(len(dep\_record\_all)):

new\_record = []

x\_axis = random.random()\*100

new\_record.append(x\_axis)

new\_record.append(dep\_record\_all[i])

new\_record.append(record[i][0])

plastic\_type.append(new\_record)

np.save(path1+'Random\_Sim\_7\_Info\_normal\_200\_80.npy',record)

np.save(path1+'Random\_Sim\_7\_Depth\_normal\_200\_80.npy',dep\_record\_all)

dataframe = pd.DataFrame(plastic\_type)

dataframe.to\_excel(path1+'plastic\_type\_7\_200\_80.xls')