In [1]:
```
import sys,argparse
import pandas as pd
from MD_file1 import *
from MD_file2 import *
```

In [2]:
```
df_atoms , box, atomnm, resnm, resnr, elem = read_pdb('np_sol.pdb')
```

In [3]:
```
x = boundary(box, resnr, df_atoms)
```

In [23]:
```
def stability(dt,D,dx,dz,dy):

    Von_neu = ((2*dt*D)/(dx**2)) + ((2*dt*D)/(dy**2)) + ((2*dt*D)/(dz**2))
    if Von_neu <= 1:
        print('Von_Neumann Stability Condition is satisfied\n')
        return 1
    else:
        print('Von_Neumann Stability Condition is not satisfied\n')
        return 0
```

In [41]:
```
def do_timestep(u0, u, dx2, dt, D):
    # Propagate with forward-difference in time, central-difference in space
    u[1:-1, 1:-1, 1:-1] = u0[1:-1, 1:-1, 1:-1] + D[1:-1, 1:-1, 1:-1] * dt * (
          (u0[2:, 1:-1, 1:-1] - 2*u0[1:-1, 1:-1, 1:-1] + u0[:-2, 1:-1, 1:-1])/dx2
          +(u0[1:-1:, 2:, 1:-1] - 2*u0[1:-1, 1:-1, 1:-1] + u0[1:-1, :-2, 1:-1])/dx2
          +(u0[1:-1, 1:-1, 2:] - 2*u0[1:-1, 1:-1, 1:-1] + u0[1:-1, 1:-1, :-2])/dx2)

    #u0 = u.copy()
    return u
```

In [25]:
```
def Boundary(X,T,N):
    for ii in range(N):
        if ii == 0 or ii == (N-1):
            for jj in range(N):
                for kk in range(N):
                    X[ii,jj,kk] = T
```

```
    for jj in range(N):
        if jj == 0 or jj == (N-1):
            for ii in range(N):
                for kk in range(N):
                    X[ii,jj,kk] = T
    for kk in range(N):
        if kk == 0 or kk == (N-1):
            for ii in range(N):
                for jj in range(N):
                    X[ii,jj,kk] = T

    return X
```

In [26]:

```
import numpy as np
def heating(A,B,T1,T2):
    for ii in range(nx):
        for jj in range(ny):
            for kk in range(nz):
                if A[ii,jj,kk] == 10:
                    B[ii,jj,kk] = T2
    return B
```

In [27]:

```
def heat_power(r,per,perm,W,I):
    ln = (((per-perm)/(per+(2*perm)))**2)
    k = ((2*(math.pi)*(math.sqrt(perm)))/W)
    Absorbtion = (((4*(math.pi)*k*(r**3))*math.log(ln))**2)
    Absorbtion = math.sqrt(Absorbtion)
    power = Absorbtion*I
    return power
```

In [28]:

```
def mass(D,rho):
    v = ((math.pi)*(float(D**3)))/6
    m = (rho)*v
    return m
```

In [29]:

```
def temp(P,T,t,Cp,m):
    T_kel = (T + ((t*P)/(Cp*m)))
    return T_kel
```

In [30]:
```
def remove(string):
    return string.replace(" ", "")

N = len(elem)
n = 0
for ll in range(N):
    if remove(elem[ll]) == "AU":
        n +=1
print(n)
```

```
2531
```

In [31]:
```
np_box = []
np_box.append(max((df_atoms['x_coords'].tolist())[:2530]) - min((df_atoms['x_coords'].tolist())[:2530]))
np_box.append(max((df_atoms['y_coords'].tolist())[:2530]) - min((df_atoms['y_coords'].tolist())[:2530]))
np_box.append(max((df_atoms['z_coords'].tolist())[:2530]) - min((df_atoms['z_coords'].tolist())[:2530]))

print(np_box,box)
```

```
[40.78, 40.78, 40.78] [141.0, 141.0, 141.0]
```

In [32]:
```
import numpy as np
def Heat_cap(bx,bxx,A,W,M):
    a = (A[0]/(A[1]*A[2])) * 100
    w = (W[0]/(W[1]*W[2])) * 100
    m = (M[0]/(M[1]*M[2])) * 100
    ll= [a,w,m]
    for i in range(nx):
        for j in range(ny):
            for k in range(nz):
                if bx[i,j,k] == 10:
                    bxx[i,j,k] = m
                elif bx[i,j,k] == 0:
                    bxx[i,j,k] = a
                else:
                    bxx[i,j,k] = w
    print(a,w,m)
    return bxx,ll
```

In [43]:

```
import math
import numpy as np

D = (((sum(np_box))/30) * (10**(-9)))
r = (((sum(np_box))/60) * (10**(-9)))

W = 800*(10**(-9))
per_np = 1.145
per_w  = 1.0
I = 0.2

Cp1 = 129

P1 = 1200

T1 = 310.0

dt = 10
Nt = 100001
Tt = int(Nt/dt)

power = heat_power(r,per_np,per_w,W,I)

N = len(elem)
#mass = mass(D,P1)
mass = (n*197.0)*(1.66*(10**(-27)))

T2 = temp(power,T1,1,Cp1,mass)

print(mass,T2)
```

```
8.2768762e-22 319.64405129808125
```

In [34]:

```
def remove(string):
    return string.replace(" ", "")

import numpy as np
import math

N = len(elem)
dim = nx = ny = nz = 142
dl = dx = dy = dz = 1
dl2 = dl*dl
```

```
R = np.zeros((nx,ny,nz),dtype=np.float128)
Ti = np.ones((nx,ny,nz),dtype=np.float128) * T1

X = (df_atoms['x_coords'].tolist())[:]
Y = (df_atoms['y_coords'].tolist())[:]
Z = (df_atoms['z_coords'].tolist())[:]

for l in range(N):
    xx = math.floor(X[l])
    yy = math.floor(Y[l])
    zz = math.floor(Z[l])

    if remove(elem[l]) == "AU":
        R[xx,yy,zz] += 10
    else:
        R[xx,yy,zz] += 1

spc = np.zeros((nx,ny,nz),dtype=np.float128)
air_cp   = [0.024,1005.0,1.225]
water_cp = [0.6089,4196.0,997.0]
metal_cp = [314.0,129.0,19300.0]
spc,val = Heat_cap(R,spc,air_cp,water_cp,metal_cp)

print("Stability of Air medium:")
tt1 = stability(dt,val[0],dx,dz,dy)
print("Stability of Water Medium:")
tt2 = stability(dt,val[1],dx,dz,dy)
print("Stability of Metal medium:")
tt3 = stability(dt,val[2],dx,dz,dy)
tt = tt1 + tt2 + tt3
```

```
0.0019494364910143161 1.4555104780499744e-05 0.012611961280475559
Stability of Air medium:
Von_Neumann Stability Condition is satisfied

Stability of Water Medium:
Von_Neumann Stability Condition is satisfied

Stability of Metal medium:
Von_Neumann Stability Condition is satisfied
```

In [39]:

```
num = 0
```

```
for q in range(Tt):
    Ti = heating(R,Ti,T1,T2)
    To = Ti
    Ti = do_timestep(To, Ti, dl2, dt, 0.1)
    if num%10 == 0:
        name  = str(num) + ".out"

        with open(name, 'w') as f:
            for ii in range(nx):
                for jj in range(ny):
                    for kk in range(nz):
                        line = str(ii)+str(" ")+str(jj)+str(" ")+str(kk)+str(" ")+str(To[ii,jj,kk])
                        f.write(line)
                        f.write('\n')
    Ti = Boundary(Ti,T1,dim)
    num += 1
```

In [ ]:

```
num = 100001
for q in range(Tt):
    To = Ti
    Ti = do_timestep(To, Ti, dl2, dt, spc)
    if num%1000 == 0:
        name  = str(num) + ".out"

        with open(name, 'w') as f:
            for ii in range(nx):
                for jj in range(ny):
                    for kk in range(nz):
                        line = str(ii)+str(" ")+str(jj)+str(" ")+str(kk)+str(" ")+str(To[ii,jj,kk])
                        f.write(line)
                        f.write('\n')
    Ti = Boundary(Ti,T1,dim)
    num += 1
```

In [40]:

```
f = open("100000.out", "r")
for  mm in range(1973788):
    txt = f.readline()

    x = txt.split(" ")
    i1 = int(x[0])
    j1 = int(x[1])
```

```
    k1 = int(x[2])
    tet = float(x[3])

    Ti[i1,j1,k1] = tet
```

In [ ]: