Script 1 (LAMMPS): NVT simulations to equilibrate the system

```
variable T equal 94.0

units metal
boundary p p p
atom_style atomic

lattice fcc 5.78015
region simulation_box block 0 5 0 5 0 5 # 500 particles
create_box 1 simulation_box
create_atoms 1 region simulation_box
mass 1 39.948002
velocity all create $T 20000802 dist gaussian

neighbor 0.1 bin
neigh_modify delay 10 check no

pair_style lj/cut 15.0
pair_coeff 1 1 0.0103 3.405 15.0

fix 1 all nvt temp $T $T 0.5
thermo 1
thermo_style custom step temp pe ke etotal press density
dump 1 all custom 1 nvt.dump id type x y z vx vy vz
dump_modify 1 sort id
run 100000
unfix 1
```

Script 2 (LAMMPS): NVE simulations to calculate g(r)

```
variable T equal 94.0

units metal
boundary p p p
atom_style atomic

read_data nve.lmp

neighbor 0.1 bin
neigh_modify delay 10 check no

pair_style lj/cut 15.0
pair_coeff 1 1 0.0103 3.405 15.0

thermo 1000
thermo_style custom step temp pe ke etotal press density

# Calculate the kinetic energy
compute g2 all rdf 1000
fix 1 all nve
fix temp_1 all ave/time 1 10000 10000 c_thermo_temp file temp.dat
fix g2_1 all ave/time 1 5000000 5000000 c_g2[*] file rdf.dat mode vector

dump 1 all custom 1000 argon.dump id type x y z vx vy vz

run 5000000
```

Script 3 (Python): Plot g(r) and S(q)

```python
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import rcParams
from matplotlib.ticker import AutoMinorLocator

rcParams['font.family'] = 'sans-serif'
rcParams['font.size'] = '16'
rcParams['font.sans-serif'] = 'Arial'
rcParams['mathtext.fontset'] = 'custom'
rcParams['mathtext.rm'] = 'Arial'
rcParams['mathtext.it'] = 'Arial:italic'
rcParams['mathtext.bf'] = 'Arial:bold'
rcParams['xtick.direction'] = 'in'
rcParams['ytick.direction'] = 'in'

def structure_factor(q, r, g_r, rho):
    dr = r[1] - r[0]
    integrand = 4 * np.pi * r * rho * (g_r - 1) * np.sin(q * r) / q
    return 1 + np.sum(integrand) * dr

g2_exp = np.loadtxt('g2.dat', unpack=True)
Sq_exp = np.loadtxt('Sq.dat', unpack=True)

fig, ax = plt.subplots(figsize=(10, 8))
rho = 0.0207
q_values = np.linspace(0.01, 10, 500)
ax.set_xlabel('$q$ (Å$^{-1}$)')
ax.set_ylabel('$S(q)$')

clist = ['C1', 'C2', 'C4', 'C5', 'C0']
num_list = ['108', '192', '256', '400', '500']

ax.plot(Sq_exp[0], Sq_exp[1], c='k', label = 'Yarnell et al. (1973)', alpha = 0.5)
for index, i in enumerate(num_list):
    data = np.loadtxt('./' + i + '/rdf.dat', skiprows=4, unpack=True)
    r = data[1]
    g_r= data[2]
    s_q_values = [structure_factor(q, r, g_r, rho) for q in q_values]
    ax.plot(q_values, s_q_values, c = clist[index], label = '$N$ = ' + i, alpha = 0.8)

plt.legend(fancybox=False, edgecolor='black', fontsize=14)
ax.xaxis.set_minor_locator(AutoMinorLocator())
ax.yaxis.set_minor_locator(AutoMinorLocator())
plt.savefig('../Sq.pdf', bbox_inches='tight')

plt.cla()
ax.plot(g2_exp[0], g2_exp[1], c='k', label = 'Yarnell et al. (1973)', alpha = 0.5)
```

```
for index, i in enumerate(num_list):
    data = np.loadtxt('./' + i + '/rdf.dat', skiprows=4, unpack=True)
    r = data[1]
    g_r = data[2]
    ax.plot(r, g_r, c = clist[index], label = '$N$ = ' + i, alpha = 0.8)
ax.set_xlabel('$r$ (Å)')
ax.set_ylabel('$g_2(r)$')
ax.xaxis.set_minor_locator(AutoMinorLocator())
ax.yaxis.set_minor_locator(AutoMinorLocator())

plt.legend(fancybox=False, edgecolor='black', fontsize=14)
plt.savefig('../g2.pdf', bbox_inches='tight')
```

Script 4 (LAMMPS): NVE simulations to calculate D

```
units metal
boundary p p p
atom_style atomic

read_data nve.lmp

neighbor 0.1 bin
neigh_modify delay 10 check no

pair_style lj/cut 15.0
pair_coeff 1 1 0.0103 3.405 15.0

thermo 100
thermo_style custom step temp pe ke etotal press density

variable i loop 1000

# Define lable for the loop
label loop_start

reset_timestep 0
compute msd_Ar_${i} all msd
compute vacf_Ar_${i} all vacf
fix ${i} all nve
fix store_msd_Ar_${i} all vector 1 c_msd_Ar_${i}[4]
fix store_vacf_Ar_${i} all vector 1 c_vacf_Ar_${i}[4]
fix msd_${i} all ave/time 1 1 1 c_msd_Ar_${i}[4] file ./msd/msd.${i}.dat
fix vacf_${i} all ave/time 1 1 1 c_vacf_Ar_${i}[4] file ./vacf/vacf.${i}.dat
fix tmp_${i} all ave/time 1 1000 1000 c_thermo_temp file ./temp/temp.${i}.dat

dump d_${i} all custom 1000 ./dump/argon.${i}.dump id type x y z vx vy vz

run 10000

undump d_${i}
unfix store_msd_Ar_${i}
unfix store_vacf_Ar_${i}
unfix msd_${i}
unfix vacf_${i}
unfix tmp_${i}
unfix ${i}

next i

# Repeat simulation 10 times
jump SELF loop_start
```

Script 5 (Python): Plot the MSD and VACF

```python
import numpy as np
import pickle as pkl
from matplotlib import pyplot as plt
from matplotlib import rcParams
from matplotlib.ticker import AutoMinorLocator
from matplotlib.ticker import LogLocator

rcParams['font.family'] = 'sans-serif'
rcParams['font.size'] = '16'
rcParams['font.sans-serif'] = 'Arial'
rcParams['mathtext.fontset'] = 'custom'
rcParams['mathtext.rm'] = 'Arial'
rcParams['mathtext.it'] = 'Arial:italic'
rcParams['mathtext.bf'] = 'Arial:bold'
rcParams['xtick.direction'] = 'in'
rcParams['ytick.direction'] = 'in'

def vacftoD(vacf, t):
    dt = t[1] - t[0]
    integrand = vacf
    return np.sum(integrand * dt)

def fit(x, y, deg):
    z = np.polyfit(x, y, deg=deg, full=True)
    pn = np.poly1d(z[0])
    R_square = 1 - z[1][0] / sum((y - np.mean(y))** 2)
    return pn, R_square

fig, ax = plt.subplots(figsize=(10, 8))

q_values = np.linspace(0.01, 10, 500)
ax.set_xlabel('$t$ (fs)')
ax.set_ylabel('$\Delta R^2$ (Å$^{2}$)')

clist = ['C1', 'C2', 'C4', 'C5', 'C0']
num_list = ['108', '192', '256', '400', '500']
D = dict()
for index, i in enumerate(num_list):
    msd = np.zeros(10001)
    t = np.arange(len(msd))
    D[i] = dict()
    D[i]['msd'], D[i]['vacf'], D[i]['temp'] = [], [], []
    for j in range(1, 1001):
        msd_tmp = np.loadtxt('./' + i + '/nve/msd/msd.' + str(j) + '.dat',\
        skiprows=2, unpack=True)[1]
        temp = np.loadtxt('./' + i + '/nve/temp/temp.' + str(j) + '.dat',\
        skiprows=2, unpack=True)[1]
        pn, Rs = fit(t[2000:], msd_tmp[2000:], 1)
        D[i]['msd'].append(pn[1]/6 * 1e-5)
        D[i]['temp'].append(np.mean(temp))
        msd += msd_tmp
    msd /= 1000
    ax.loglog(t, msd, c = clist[index], label = '$N$ = ' + i, alpha = 0.8)

ax.legend(fancybox=False, edgecolor='black', fontsize=14)
ll = LogLocator(base=10.0,subs=([0.1 * i for i in range(1,10)]),numticks=12)
ax.xaxis.set_minor_locator(ll)
ax.yaxis.set_minor_locator(ll)
plt.savefig('../msd.pdf', bbox_inches='tight')

vacflist = []
plt.cla()
for index, i in enumerate(num_list):
    vacf = np.zeros(10001)
    t = np.arange(len(vacf))
    for j in range(1, 1001):
        vacf_tmp = np.loadtxt('./' + i + '/nve/vacf/vacf.' + str(j) + '.dat',\
        skiprows=2, unpack=True)[1]
        D[i]['vacf'].append(vacftoD(vacf_tmp[:5000], t[:5000]) / 3 * 1e-11)
        vacf += vacf_tmp
    vacf /= 1000
    vacflist.append(vacf)
    ax.plot(t[:5000], vacf[:5000], c = clist[index], label = '$N$ = ' + i, alpha = 0.8)

ax.legend(fancybox=False, edgecolor='black', fontsize=14)
ax.set_xlabel('$t$ (fs)')
ax.set_ylabel('VACF ($10^{-6}$ Å$^{2}$fs$^{-2}$)')
ax.xaxis.set_minor_locator(AutoMinorLocator())
ax.yaxis.set_minor_locator(AutoMinorLocator())
plt.savefig('../vacf.pdf', bbox_inches='tight')

with open('data.pkl', 'wb') as file:
    pkl.dump(D, file)
with open('vacf.pkl', 'wb') as file:
    pkl.dump(vacflist, file)
```

Script 6 (Python): Plot D

```python
import numpy as np
from matplotlib import pyplot as plt
import pickle as pkl
from matplotlib import rcParams
from matplotlib.ticker import AutoMinorLocator

rcParams['font.family'] = 'sans-serif'
rcParams['font.size'] = '16'
rcParams['font.sans-serif'] = 'Arial'
rcParams['mathtext.fontset'] = 'custom'
rcParams['mathtext.rm'] = 'Arial'
rcParams['mathtext.it'] = 'Arial:italic'
rcParams['mathtext.bf'] = 'Arial:bold'
rcParams['xtick.direction'] = 'in'
rcParams['ytick.direction'] = 'in'

def fit(x, y, deg):
    z = np.polyfit(x, y, deg=deg, full=True)
    pn = np.poly1d(z[0])
    R_square = 1 - z[1][0] / sum((y - np.mean(y))** 2)
    return pn, R_square

def vacftoPS(vacf, w, t):
    dt = t[1] - t[0]
    integrand = vacf * np.cos(w * t)
    return np.sum(integrand * dt)

with open('data.pkl', 'rb') as file:
    D = pkl.load(file)

with open('vacf.pkl', 'rb') as file:
    vacflist = pkl.load(file)

num_list = ['108', '192', '256', '400', '500']
clist = ['C1', 'C2', 'C4', 'C5', 'C0']
l = [17.34045, 23.1206, 28.90075]
length = np.array([(l[i]*l[j]*l[k])**(1/3) for i, j, k in [(0,0,0),\
(1,1,0),(1,1,1),(2,2,1),(2,2,2)]])
k_B = 1.380649e-23 #J/K

# for i in num_list:
#     p = [np.mean(D[i][j]) for j in ['msd', 'vacf', 'temp']]
#     print(i, p[0], p[1], p[2])

for i in num_list:
    p = [np.mean(D[i][j]) for j in ['msd', 'vacf', 'temp']]
    p = [np.std(D[i][j]) for j in ['msd', 'vacf', 'temp']]
```

```python
T = np.mean([np.mean(D[i]['temp']) for i in num_list])

fig, ax = plt.subplots(figsize=(8, 8))

for sys in ['msd', 'vacf']:
    Diffusion = np.array([np.mean(D[i][sys]) for i in num_list]) * 1e9
    err = np.array([2 * np.std(D[i][sys]) / np.sqrt(len(D[i][sys])) for i in num_list])
    ax.scatter(1/length, Diffusion, marker='s', s=50, c='C0',\
    edgecolors=(0.122, 0.467, 0.706, 0.6), linewidths=2, zorder=10)
    ax.errorbar(1/length, Diffusion, marker='s', yerr=err, ls='none',\
    ecolor='k', elinewidth=1, capsize=4, capthick=1, zorder=0)
    linear, Rs = fit(1/length, Diffusion, deg=1)
    x = np.linspace(0, 1 / np.min(length), 1000)
    ax.plot(x, linear(x), ls='--', c = 'gray')
    ax.set_xlim(0, 0.06)
    ax.set_ylim(2.02, 2.72)
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.set_xlabel('$1/L$ ($\mathrm{\AA^{-1}}$)')
    ax.set_ylabel(r'$D$ ($10^{-9}$ m$^2$/s)')
    plt.savefig('../D_' + sys + '.pdf', bbox_inches='tight')
    plt.cla()
    eta = - k_B * T * 2.837 / (6 * np.pi * linear[1] * 1e-19) * 1e6
    print(sys, eta, 'uPa·s', linear[0], '*1e-9 m2/s')

# msd  183.9 uPa·s 2.702 *1e-9 m2/s
# vacf 192.3 uPa·s 2.699 *1e-9 m2/s

w = np.linspace(0, 0.04, 10000)

for index, i in enumerate(num_list):
    vacf = vacflist[index][:5000]
    t = np.arange(len(vacf))
    ps = np.array([vacftoPS(vacf, j, t) for j in w]) / (300 * Diffusion[index])
    ax.plot(w, ps, c = clist[index], label = '$N$ = ' + i, alpha = 0.8)
ax.plot(w, 1e-4 / (w**2 + 1e-4), c='k', ls='--', label='Brownian fluid', alpha=0.5)
ax.xaxis.set_minor_locator(AutoMinorLocator())
ax.yaxis.set_minor_locator(AutoMinorLocator())
ax.set_xlabel('$\omega$ (fs$^{-1}$)')
ax.set_ylabel('$f(\omega$)')
ax.legend(fancybox=False, edgecolor='black', fontsize=14)
plt.savefig('../ps.pdf', bbox_inches='tight')
```

Script 7 (LAMMPS): NVE simulations to calculate Cv

```
variable T equal 94.0

units metal
boundary p p p
atom_style atomic

read_data nve.lmp

neighbor 0.1 bin
neigh_modify delay 10 check no

pair_style lj/cut 15.0
pair_coeff 1 1 0.0103 3.405 15.0

thermo 1000
thermo_style custom step temp pe ke etotal press density

# Calculate the kinetic energy
compute ke_Ar all ke
fix 1 all nve
fix temp_1 all ave/time 1 10000 10000 c_thermo_temp file temp.dat
fix ke_1 all ave/time 1 1 1 c_ke_Ar file ke.dat

dump 1 all custom 1000 argon.dump id type x y z vx vy vz

run 1000000
```

Script 6 (Python): Plot Cv

```python
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import rcParams
from matplotlib.ticker import AutoMinorLocator

rcParams['font.family'] = 'sans-serif'
rcParams['font.size'] = '16'
rcParams['font.sans-serif'] = 'Arial'
rcParams['mathtext.fontset'] = 'custom'
rcParams['mathtext.rm'] = 'Arial'
rcParams['mathtext.it'] = 'Arial:italic'
rcParams['mathtext.bf'] = 'Arial:bold'
rcParams['xtick.direction'] = 'in'
rcParams['ytick.direction'] = 'in'

k_B = 8.617333262e-5 #eV/K

num_list = [108, 192, 256, 400, 500]
Cv, err = [], []

for index, n in enumerate(num_list):
    ke_square = np.loadtxt('./' + str(n) + '/ke.dat', skiprows=2, unpack=True)[1]
    ke_blocks = np.split(ke_square[1:], 100)
    t = np.loadtxt('./' + str(n) + '/temp.dat', skiprows=2, unpack=True)[1]
    avg_ke = np.mean(ke_blocks, axis=1)
    dK = np.var(ke_blocks, axis=1)
    cv = 3 * n * k_B / (2 - 4 * dK / (3 * k_B**2 * t**2 * n))
    Cv.append(np.mean(cv))
    err.append(2 * np.std(cv) / np.sqrt(len(cv)))

fig, ax = plt.subplots(figsize=(8, 8))
ax.scatter(num_list, Cv, marker='s', s=50, c='C0',\
edgecolors=(0.122, 0.467, 0.706, 0.6), linewidths=2, zorder=10)
ax.errorbar(num_list, Cv, marker='s', yerr=err, ls='none', ecolor='k',\
elinewidth=1, capsize=4, capthick=1, zorder=0)
ax.xaxis.set_minor_locator(AutoMinorLocator())
ax.yaxis.set_minor_locator(AutoMinorLocator())
ax.set_xlabel('$N$')
ax.set_ylabel(r'$C_V$ (eV/K)')
plt.savefig('../Cv.pdf', bbox_inches='tight')
```