```
# Set parameters and run the experiment!!!
Ns = [10]
J = 1.0
H = 0.0
Ts = np.linspace(1.8,2.26,40)
sweeps = 12000
equilibrating_sweeps = 2000

magnetisation, energy, magnetisation_per_site, energy_per_site, run_time =
Metropolis_algorithm(Ns,J,H,Ts,sweeps)


# In[ ]:


# Plot time evolution
N_indices = range(len(Ns))
T_indices = [0,19,39]
startpt = 0
stoppt = sweeps
plot_time_series(magnetisation_per_site,Ns,N_indices,Ts,T_indices,startpt,stoppt,'Sweeps'
,'Magnetisation per site')


# In[ ]:


# Plot run time vs N to investigate program complexity
run_time_data = np.zeros(len(Ns))
for N_index in N_indices:
    run_time_data[N_index] = np.average(run_time[N_index,:])
plt.plot(Ns,run_time_data,'-o')
plt.xlabel('Lattice size N')
plt.ylabel('Run time (s)')
plt.savefig('plots/Run time vs Lattice size.pdf')


# In[ ]:


# How total magnetisation fluctuates with time when system is in equilibrium
magnetisation_autocovariance =
calculate_autocovariation(magnetisation,Ns,Ts,sweeps,equilibrating_sweeps)


# In[ ]:


# Plot autocovariance of total magnetisation
N_indices = range(len(Ns))
T_indices = range(26,31)
startpt = 0
stoppt = sweeps
plot_time_series(magnetisation_autocovariance,Ns,N_indices,Ts,T_indices,startpt,stoppt,'t
au','Autocorrelation')


# In[ ]:


# Calculate thermodynamic variables
average_magnetisation =
calculate_thermodynamic_variable(np.abs(magnetisation),Ns,Ts,equilibrating_sweeps)
average_energy = calculate_thermodynamic_variable(energy,Ns,Ts,equilibrating_sweeps)
average_susceptibility =
calculate_derivative_thermodynamic_average(np.abs(magnetisation),Ns,Ts,1,equilibrating_sw
eeps)
```