```python
            plt.plot(range(start_point,stop_point),data[N_index,T_index,start_point:stop_point],label='N = {0}, T = {1:.3f}'.format(N_range[N_index],T_range[T_index]))
        plt.legend(loc='upper right')
        plt.xlabel(xname)
        plt.ylabel(yname)
        plt.savefig('plots/{0} vs {1} N = {2} from {3} to {4}.pdf'.format(yname,xname,N_range[N_index],start_point,stop_point))
        plt.figure()


# In[ ]:


# Function to calculate the average of a thermodynamic variable
def calculate_thermodynamic_variable(data, N_range, T_range, no_of_equilibrating_sweeps):
    var = np.zeros((len(N_range),len(T_range)))

    for N_index in range(len(N_range)):
        for T_index in range(len(T_range)):
            var[N_index,T_index] = np.sum(data[N_index,T_index,no_of_equilibrating_sweeps:])/len(data[N_index,T_index,no_of_equilibrating_sweeps:])
        var[N_index,:] = normalize_data(var[N_index,:],N_range[N_index])

    return var


# In[ ]:


# Function to calculate the average of a derivative thermodynamic variable
def calculate_derivative_thermodynamic_average(data, N_range, T_range, power_of_T, no_of_equilibrating_sweeps):
    var = np.zeros((len(N_range),len(T_range)))

    for N_index in range(len(N_range)):
        for T_index in range(len(T_range)):
            ave = np.sum(data[N_index,T_index,no_of_equilibrating_sweeps:])/len(data[N_index,T_index,no_of_equilibrating_sweeps:])
            squared_ave = np.sum(data[N_index,T_index,no_of_equilibrating_sweeps:]**2)/len(data[N_index,T_index,no_of_equilibrating_sweeps:])
            var[N_index,T_index] = (squared_ave-ave**2)/(T_range[T_index]**power_of_T)
        var[N_index,:] = normalize_data(var[N_index,:],N_range[N_index])

    return var


# In[ ]:


# Function to plot thermodynamic variable against temperature
def plot_temperature_dependence(data, N_range, N_index_range, T_range, T_index_range, xname, yname):
    for N_index in N_index_range:

        plt.plot(T_range[T_index_range[0]:(T_index_range[-1]+1)],data[N_index,T_index_range[0]:(T_index_range[-1]+1)],'-o',label='N = {0}'.format(N_range[N_index]))
        # +1 to include last element
    plt.legend(loc='best')
    plt.xlabel(xname)
    plt.ylabel(yname)
    plt.savefig('plots/{0} vs {1} from {2:.3f} to {3:.3f}.pdf'.format(yname,xname,T_range[T_index_range[0]],T_range[T_index_range[-1]]))
```