Pair Correlation Function in 1D Systems

Introduction

The **pair correlation function** $g(r)$ quantifies how particles in a system are distributed relative to each other. It provides insight into clustering, ordering, or randomness in particle arrangements by calculating the probability of finding two particles separated by a distance r.

Mathematical Definition

For a 1D lattice with N particles and periodic boundary conditions:

$$g(r) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} \delta(|r_i - r_j| - r),$$

where:

- r_i and r_j: Positions of particles i and j.
- $|r_i - r_j|$: Distance between particles i and j, modulo the lattice size for periodic boundary conditions.
- $\delta(x)$: Equal to 1 if $x = 0$, otherwise 0.

Interpretation of $g(r)$

- **Clustering:** A high $g(r)$ for small r indicates that particles tend to form clusters.

- **Ordering:** Peaks at regular intervals in $g(r)$ indicate periodic ordering of particles.
- **Disorder:** A flat $g(r)$ close to 1 suggests a random (disordered) particle distribution.

Examples

Case 1: Ordered Configuration

Consider a lattice with alternating red and blue particles:

$$\text{Red (R): } R, B, R, B, R, B, \ldots$$

For such a configuration:

- $g(r = 1) = 0$: No two red particles are separated by a distance of 1.
- $g(r = 2)$ is high: Red particles are separated by 2 sites.

Case 2: Clustered Configuration

Now consider a lattice where all red particles are clustered together:

$$\text{R, R, R, B, B, B, } \ldots$$

For such a configuration:

- $g(r = 1)$ is high: Adjacent red particles are common.
- $g(r)$ decreases for larger r as particles are confined to clusters.

Case 3: Random Configuration

For a random arrangement of particles:

$$\text{R, B, R, R, B, R, } \ldots$$

- $g(r)$ fluctuates around 1 without distinct peaks, indicating no regular structure.

Python Implementation

The following Python code computes and visualizes $g(r)$ for a given lattice configuration.

Code Snippet

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
L = 50  # Lattice size
max_r = L // 2  # Maximum distance to consider

# Case 1: Ordered lattice (alternating 1s and 0s)
ordered_lattice = np.array([1 if i % 2 == 0 else 0 for i in
    range(L)])

# Case 2: Random lattice (randomly shuffled 1s and 0s)
random_lattice = np.array([1] * (L // 2) + [0] * (L // 2))
np.random.shuffle(random_lattice)

# Case 3: Clustered lattice (all 1s followed by all 0s)
clustered_lattice = np.array([1] * (L // 2) + [0] * (L // 2))


def pair_correlation_function(lattice, max_r):
    """Compute pair correlation function g(r) for a 1D
       lattice."""
    N = len(lattice)
    g = np.zeros(max_r)
    for i in range(N):
        for r in range(1, max_r + 1):
            neighbor = (i + r) % N  # Periodic boundary
                conditions
            if lattice[i] == lattice[neighbor]:  # Same type
                of particle
                g[r - 1] += 1

    g /= (N * 2)  # Normalize by number of particles and
       periodic neighbors
    return g
```

```
# Compute g(r) for each lattice
g_ordered = pair_correlation_function(ordered_lattice, max_r)
g_random = pair_correlation_function(random_lattice, max_r)
g_clustered = pair_correlation_function(clustered_lattice,
    max_r)

# Plot g(r) for all cases
plt.figure(figsize=(12, 6))

# Ordered lattice
plt.plot(range(1, max_r + 1), g_ordered, marker='o',
    linestyle='-', color='blue', label='Ordered')

# Random lattice
plt.plot(range(1, max_r + 1), g_random, marker='x', linestyle
    ='--', color='green', label='Random')

# Clustered lattice
plt.plot(range(1, max_r + 1), g_clustered, marker='s',
    linestyle='-', color='red', label='Clustered')

# Plot details
plt.xlabel("Distance (r)")
plt.ylabel("g(r)")
plt.title("Pair Correlation Function for Different Lattice 
    Configurations")
plt.grid(alpha=0.3)
plt.legend()
plt.show()
```

Expected Results

- For an **ordered lattice**, $g(r)$ shows peaks at regular intervals corresponding to particle spacing.
- For a **clustered lattice**, $g(r)$ is high for small r and decays for larger r.
- For a **random lattice**, $g(r)$ fluctuates around 1 without distinct peaks.

Experimentation

- Test the Python code on different lattice configurations (ordered, clustered, random).
- Adjust the lattice size (N) and interaction parameters to observe changes in $g(r)$.
- Combine $g(r)$ analysis with visualization of lattice configurations for deeper insights.