wave_pattern_0_smear_beta_13

November 6, 2024

[2]:
```
%load_ext autoreload
%autoreload 2
from modules import read_and_write
from modules import polyakov
from modules import utility
from modules import fourier_surface
from modules import surface_amplitudes as sf
from modules import globals
import numpy as np
import matplotlib.pyplot as plt
import os
import glob
```

1 Load data

[33]:
```
folders = utility.list_all_folders(globals.data_path,"60-60-86-6")
```

```
../data/output-measure-surface/su4-60-60-86-6/beta-10.80-twist-1-60-60-86-6 ,
index: 0
../data/output-measure-surface/su4-60-60-86-6/beta-10.80-twist-2-60-60-86-6 ,
index: 1
../data/output-measure-surface/su4-60-60-86-6/beta-10.82-twist-1-60-60-86-6 ,
index: 2
../data/output-measure-surface/su4-60-60-86-6/beta-10.82-twist-2-60-60-86-6 ,
index: 3
../data/output-measure-surface/su4-60-60-86-6/beta-10.83-twist-1-60-60-86-6 ,
index: 4
../data/output-measure-surface/su4-60-60-86-6/beta-10.85-twist-1-60-60-86-6 ,
index: 5
../data/output-measure-surface/su4-60-60-86-6/beta-10.85-twist-2-60-60-86-6 ,
index: 6
../data/output-measure-surface/su4-60-60-86-6/beta-10.90-twist-1-60-60-86-6 ,
index: 7
../data/output-measure-surface/su4-60-60-86-6/beta-10.90-twist-2-60-60-86-6 ,
index: 8
../data/output-measure-surface/su4-60-60-86-6/beta-10.95-twist-1-60-60-86-6 ,
index: 9
```

```
../data/output-measure-surface/su4-60-60-86-6/beta-10.95-twist-2-60-60-86-6 ,
index: 10
../data/output-measure-surface/su4-60-60-86-6/beta-11.5-twist-1-60-60-86-6 ,
index: 11
../data/output-measure-surface/su4-60-60-86-6/beta-11.5-twist-2-60-60-86-6 ,
index: 12
../data/output-measure-surface/su4-60-60-86-6/beta-12-twist-1-60-60-86-6 ,
index: 13
../data/output-measure-surface/su4-60-60-86-6/beta-12-twist-2-60-60-86-6 ,
index: 14
../data/output-measure-surface/su4-60-60-86-6/beta-13-twist-1-60-60-86-6 ,
index: 15
```

[34]:
```
smooth_surfaces= {}
choose_folder = 15
folder = folders[choose_folder]
files = glob.glob(os.path.join(folder, "surface_smooth_0"))
for file in files:
    file_name = file.split("/")[-1]
    smearing_level = file_name.split("_")[-1]
    volume, surface = read_and_write.read_surface_data(folder, file_name)
    smooth_surfaces[smearing_level] = surface
```

[35]:
```
smooth_surfaces = dict(sorted(smooth_surfaces.items(), key=lambda item:␣
 ↪int(item[0])))
smooth_surfaces
```

[35]:
```
{'0': array([[[ 0.00000e+00,  0.00000e+00,  2.01738e+01],
              [ 1.00000e+00,  0.00000e+00,  1.37780e+01],
              [ 2.00000e+00,  0.00000e+00, -2.89725e+01],
              …,
              [ 5.70000e+01,  5.90000e+01,  1.76311e+00],
              [ 5.80000e+01,  5.90000e+01,  8.85013e+00],
              [ 5.90000e+01,  5.90000e+01, -5.35053e+00]],

             [[ 0.00000e+00,  0.00000e+00,  2.05935e+01],
              [ 1.00000e+00,  0.00000e+00,  1.78206e+01],
              [ 2.00000e+00,  0.00000e+00,  4.20981e+01],
              …,
              [ 5.70000e+01,  5.90000e+01,  2.03520e+01],
              [ 5.80000e+01,  5.90000e+01,  1.16436e+01],
              [ 5.90000e+01,  5.90000e+01,  2.62330e+00]],

             [[ 0.00000e+00,  0.00000e+00, -4.09283e+00],
              [ 1.00000e+00,  0.00000e+00,  1.91456e-02],
              [ 2.00000e+00,  0.00000e+00, -3.30735e+01],
              …,
```

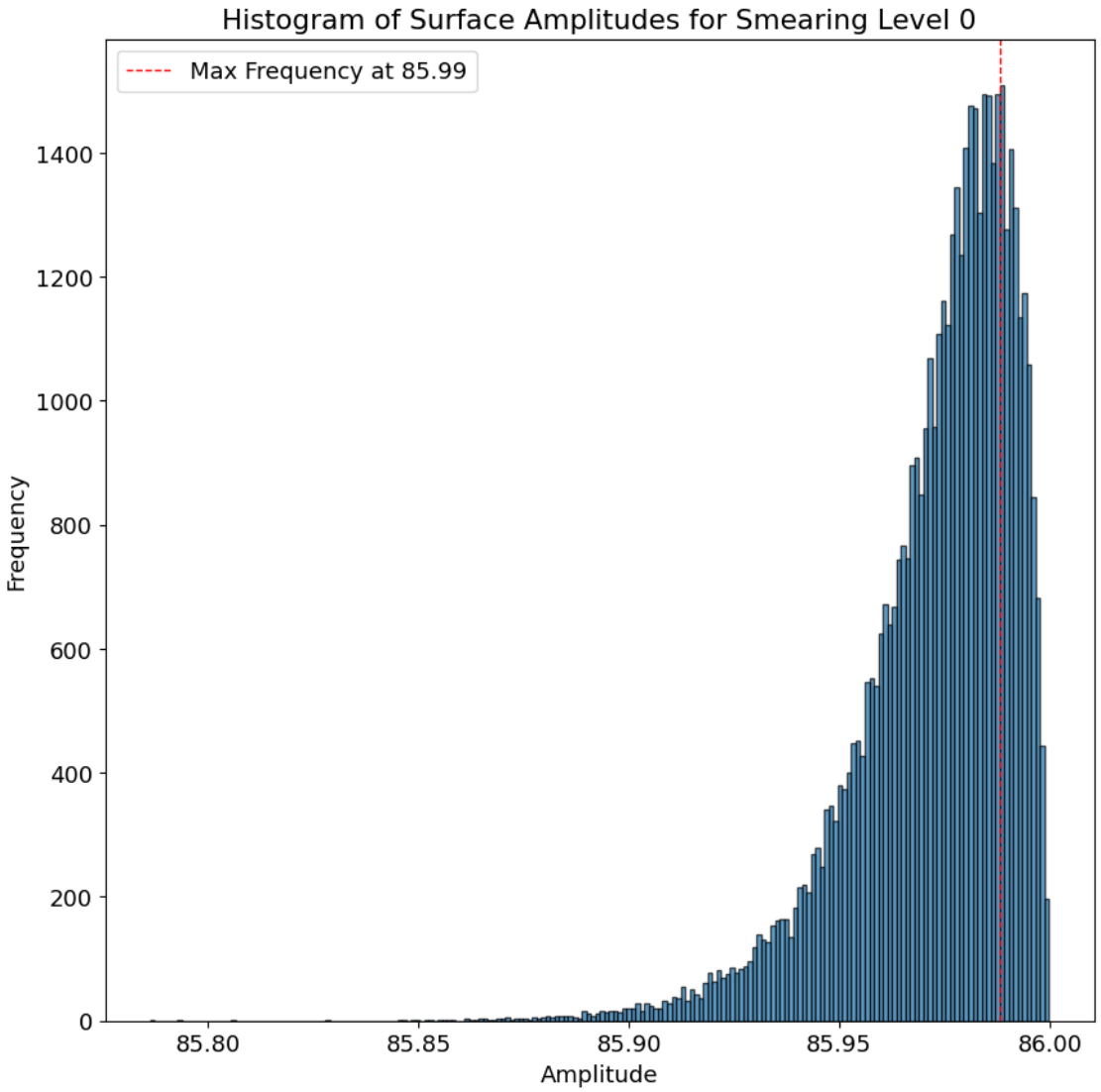
```
 [ 5.70000e+01,  5.90000e+01,  1.09540e+01],
 [ 5.80000e+01,  5.90000e+01, -7.86593e+00],
 [ 5.90000e+01,  5.90000e+01,  3.09539e+01]],

…,

[[ 0.00000e+00,  0.00000e+00,  3.18706e+01],
 [ 1.00000e+00,  0.00000e+00, -5.93565e+00],
 [ 2.00000e+00,  0.00000e+00, -5.04743e+00],
 …,
 [ 5.70000e+01,  5.90000e+01,  2.39470e+01],
 [ 5.80000e+01,  5.90000e+01, -2.30392e+01],
 [ 5.90000e+01,  5.90000e+01,  5.84352e+00]],

[[ 0.00000e+00,  0.00000e+00, -7.79116e+00],
 [ 1.00000e+00,  0.00000e+00,  1.98384e+00],
 [ 2.00000e+00,  0.00000e+00,  3.11109e+01],
 …,
 [ 5.70000e+01,  5.90000e+01,  3.98292e+01],
 [ 5.80000e+01,  5.90000e+01, -2.22241e+01],
 [ 5.90000e+01,  5.90000e+01,  3.31200e+01]],

[[ 0.00000e+00,  0.00000e+00,  4.69331e+01],
 [ 1.00000e+00,  0.00000e+00,  4.31416e+01],
 [ 2.00000e+00,  0.00000e+00, -1.83722e+01],
 …,
 [ 5.70000e+01,  5.90000e+01,  4.15494e+00],
 [ 5.80000e+01,  5.90000e+01, -1.80622e+01],
 [ 5.90000e+01,  5.90000e+01,  2.20175e+01]]])}
```

[36]:
```
utility.display_markdown_title(folder)
```

2 SU(4), V = ['60', '60', '86', '6'], β = 13, twist coeff = 1

[37]:
```
indices = sf.surface_amplitudes(smooth_surfaces=smooth_surfaces,␣
 ↪return_threshold=40,thermalization=10)
```

```
Smearing Level: 0
```



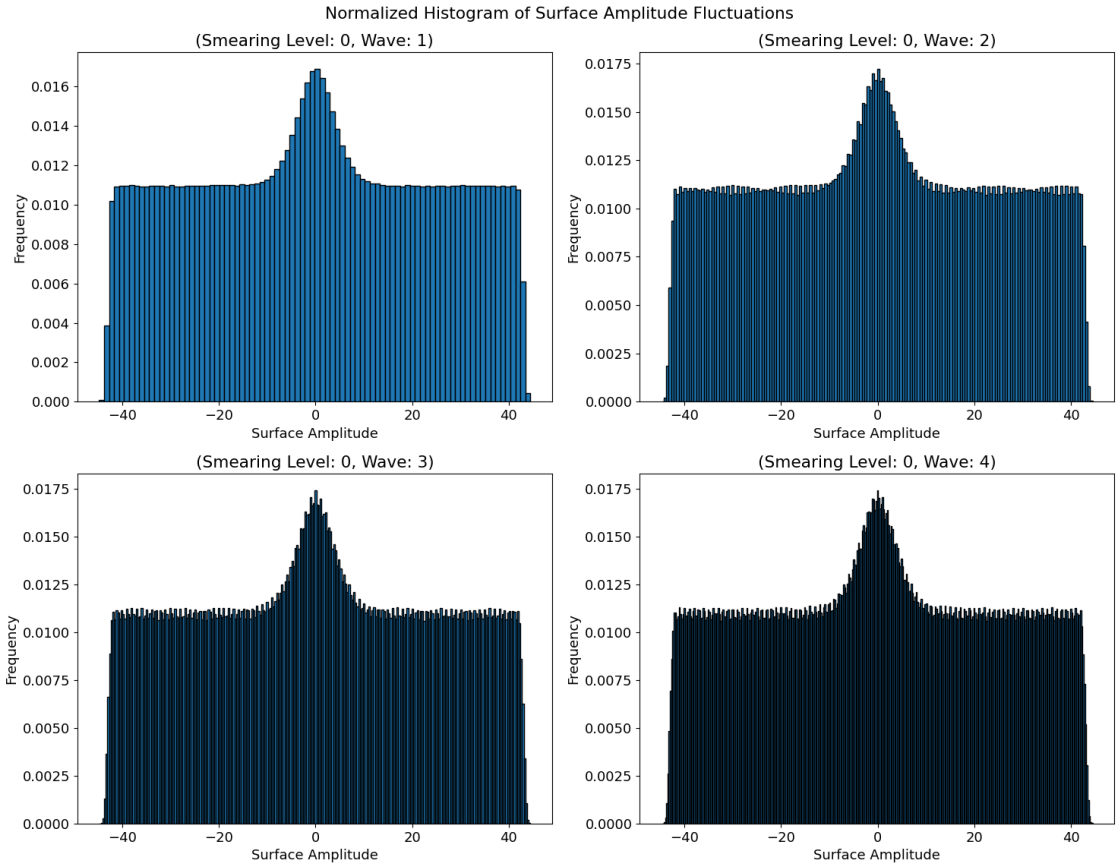

```
{'0': (85.97313600920184, 85.7862, 85.9999)}
```

[ ]:
```
# Create a dictionary to store fluctuations for each smearing level
fluctuations_dict = {}
thermalization = 1000
plt.rcParams.update({'font.size': 13})

for smearing_level, surface_data in smooth_surfaces.items():
    post_thermalization_data = surface_data[thermalization:thermalization+10000]
    mean_z_values = np.mean(post_thermalization_data[:, :, 2], axis=1)
    fluctuations = post_thermalization_data[:, :, 2] - mean_z_values[:, np.
 ↪newaxis]
    fluctuations_dict[smearing_level] = fluctuations.flatten()
```

```
# Plotting all histograms in subplots
num_plots = len(fluctuations_dict)
fig, axes = plt.subplots(num_plots, 1, figsize=(10,15))

# If there is only one histogram to be plotted, axes is not an array
if num_plots == 1:
    axes = [axes]

for ax, (smearing_level, fluctuations) in zip(axes, fluctuations_dict.items()):
    ax.hist(fluctuations, edgecolor='black', bins=86*100, density=True)
    ax.set_xlabel('Surface Amplitude')
    ax.set_ylabel('Frequency')
    ax.set_title(f'Normalized Histogram of Surface Amplitude Fluctuations␣
 ↪(Smearing Level: {smearing_level})')

plt.tight_layout()
plt.show()
```

[56]:
```
# Create a dictionary to store fluctuations for each smearing level
fluctuations_dict = {}
thermalization = 1000
plt.rcParams.update({'font.size': 13})
num_waves = 4
for smearing_level, surface_data in smooth_surfaces.items():
    post_thermalization_data = surface_data[thermalization:thermalization+10000]
    mean_z_values = np.mean(post_thermalization_data[:, :, 2], axis=1)
    fluctuations = post_thermalization_data[:, :, 2] - mean_z_values[:, np.
 ↪newaxis]
    fluctuations_dict[smearing_level] = fluctuations.flatten()

# Plotting all histograms in subplots
num_plots = len(fluctuations_dict) * num_waves
num_cols = 2
num_rows = (num_plots + num_cols - 1) // num_cols  # Calculate the number of␣
 ↪rows needed
fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, 11.7))  # A4 size in␣
 ↪inches (landscape)

# Flatten axes array for easy iteration
axes = axes.flatten()

plot_index = 0
for smearing_level, fluctuations in fluctuations_dict.items():
    for wave in range(1, num_waves + 1):
        ax = axes[plot_index]
        ax.hist(fluctuations, edgecolor='black', bins=86 * wave, density=True)
```

```
        ax.set_xlabel('Surface Amplitude')
        ax.set_ylabel('Frequency')
        ax.set_title(f'(Smearing Level: {smearing_level}, Wave: {wave})')
        plot_index += 1
fig.suptitle('Normalized Histogram of Surface Amplitude Fluctuations')
# Hide any unused subplots
for ax in axes[plot_index:]:
    ax.axis('off')

plt.tight_layout()
plt.show()
```



[ ]: