0.1 Simulation error of Wang-Landau results for black Statistical Images

```
import numpy as np
from scipy import interpolate, special
import os, h5py, hickle
import matplotlib.pyplot as plt
import pprint
```

```
import sys
if 'src' not in sys.path: sys.path.append('src')
import wanglandau as wl
from statistical_image import exact_bw_gs
```

```
datadir = 'data/black-images'
paths = [os.path.join(datadir, f) for f in os.listdir(datadir)]
len(paths)
```

```
1024
```

```
with h5py.File(paths[0], 'r') as f:
    result = hickle.load(f)
    imp = result['parameters']['system']['StatisticalImage']
    N = len(imp['I0'])
    M = imp['M']
    Es = result['results']['Es'][:-1]
```

```
pprint.pprint(result['parameters'])
```

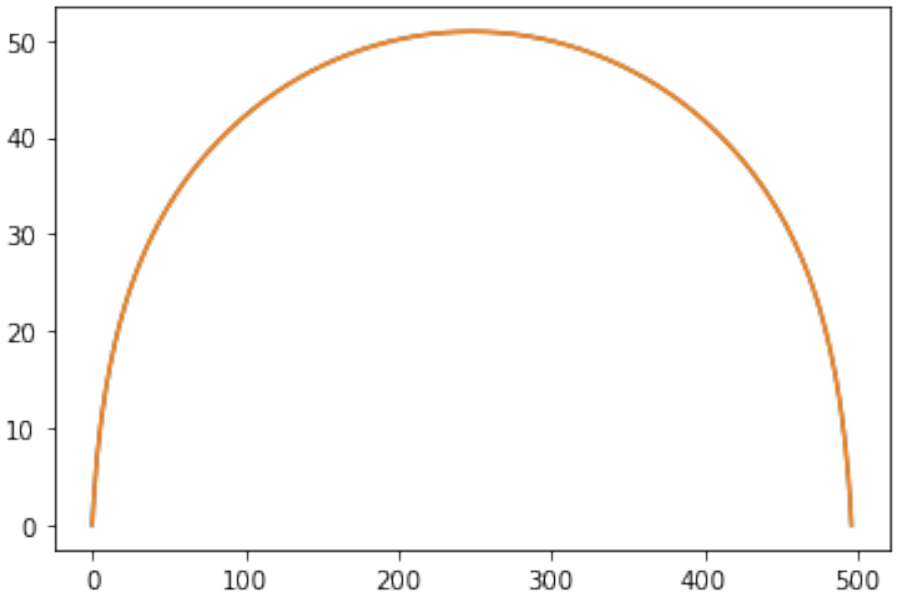
```
{'log': True,
 'simulation': {'eps': 1e-08,
                'flat_sweeps': 10000,
                'flatness': 0.2,
                'logf0': 1,
                'max_sweeps': 100000000},
 'system': {'StatisticalImage': {'I': array([10,  7,  1, 10,  6,  0,  0,  0, 28,  0,  7,  2,  1,  1,  1, 11]),
                                 'I0': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]),
                                 'M': 31}}}
```

```
def file_lngs(path):
    with h5py.File(path, 'r') as f:
        result = hickle.load(f)
        S = result['results']['S']
        # Shift for computing exponentials
        S -= min(S)
        # Set according to the correct total number of states ((M+1)**N)
        S += N*np.log(M+1) - np.log(np.sum(np.exp(S)))
        # Set according to leftmost value
#         S -= S[0]
        return S
```

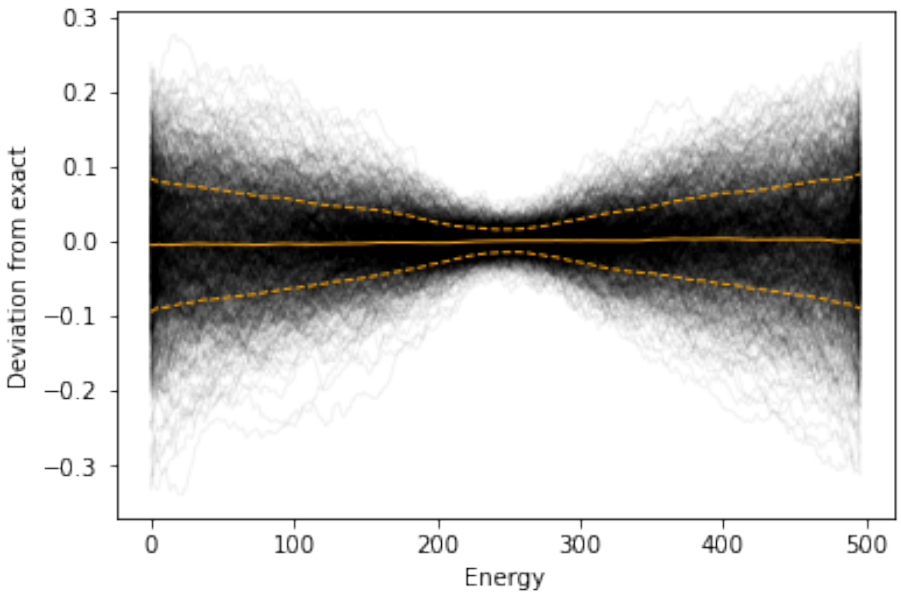
```
xEs, xgs = exact_bw_gs(N, M)
xlng = np.log(xgs)
```

```
mean_lng = np.zeros(len(Es))
std_lng = np.zeros(len(Es))
for lng in map(file_lngs, paths):
    mean_lng += lng
mean_lng /= len(paths)
for lng in map(file_lngs, paths):
    std_lng += (mean_lng - lng)**2
std_lng = np.sqrt(std_lng / (len(paths) - 1))
```

```
plt.plot(xEs, np.log(xgs))
plt.plot(Es, mean_lng);
```

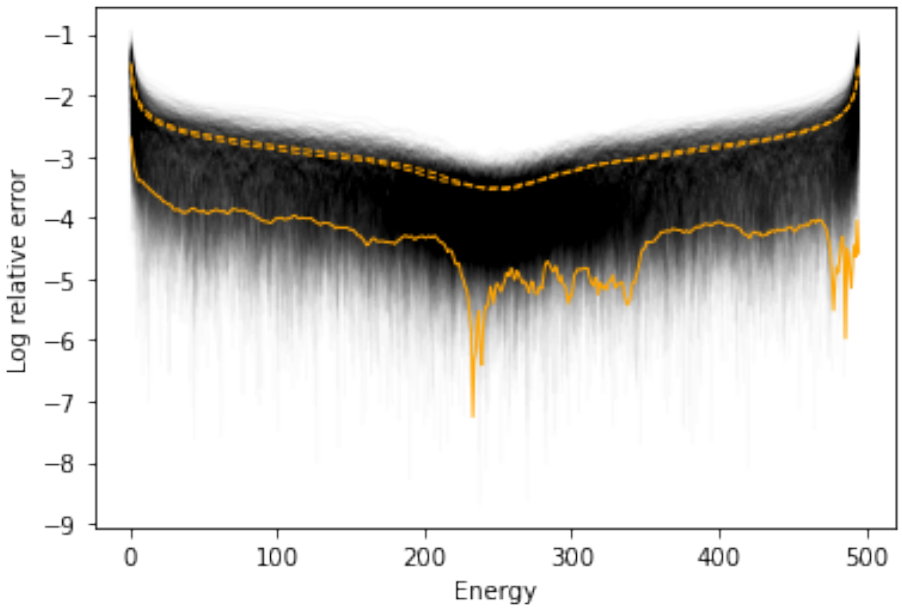



```
for lng in map(file_lngs, paths):
    plt.plot(Es, lng - xlng, 'black', alpha=0.05, linewidth=1)
plt.plot(Es, mean_lng - xlng, 'orange', linewidth=1)
plt.plot(Es, (mean_lng - std_lng) - xlng, 'orange', linestyle='dashed', linewidth=1)
plt.plot(Es, (mean_lng + std_lng) - xlng, 'orange', linestyle='dashed', linewidth=1)
plt.xlabel('Energy')
plt.ylabel('Deviation from exact');
```




```
def relative_error(sim, exact):
    if exact == 0.0:
        return np.inf
    else:
        return np.abs(sim - exact) / exact
def log_relerror(sim):
    return np.log10(np.vectorize(relative_error)(sim, xlng))
```

```
for lng in map(file_lngs, paths):
    plt.plot(Es, log_relerror(lng), 'black', alpha=0.02, linewidth=1)
plt.plot(Es, log_relerror(mean_lng), 'orange', linewidth=1)
plt.plot(Es, log_relerror(mean_lng - std_lng), 'orange', linestyle='dashed', linewidth=1)
plt.plot(Es, log_relerror(mean_lng + std_lng), 'orange', linestyle='dashed', linewidth=1)
plt.xlabel('Energy')
plt.ylabel('Log relative error');
```


−1
−2
−3
−4
−5
−6
−7
−8
−9
Log relative error
0
100
200
300
400
500
Energy