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

```python
import numpy as np
from scipy import interpolate, special
import os, h5py, hickle
import matplotlib.pyplot as plt
import pprint
plt.rcParams['font.size'] = 12
```

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

0.1.1 The setup

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

```
1024
```

```python
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]
```

```python
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}}}
```

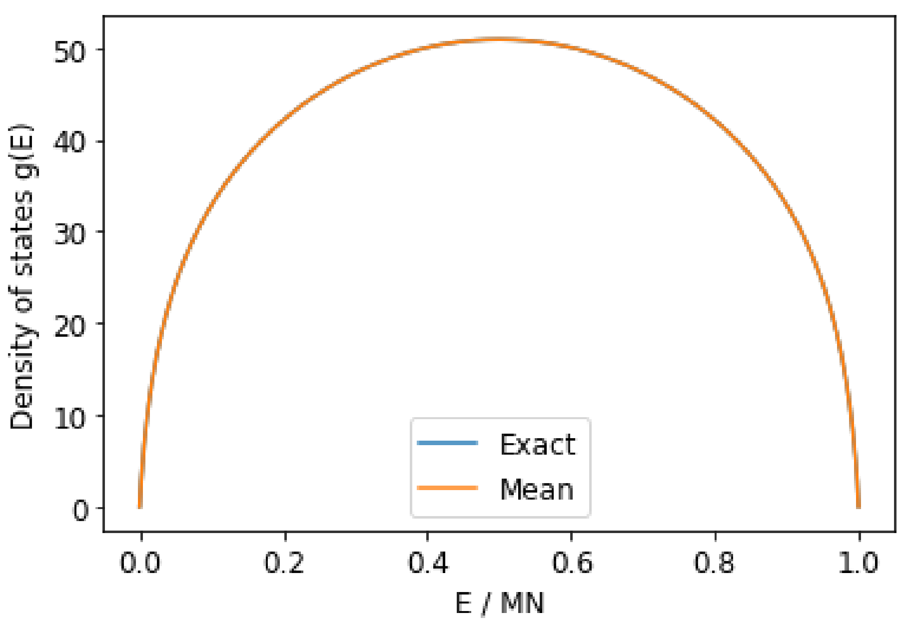
```python
def file_lngs(path):
    with h5py.File(path, 'r') as f:
        result = hickle.load(f)
        S = result['results']['S']
        # Shift for computing exponentials
        S -= min(S)
        # Set according to the correct total number of states ((M+1)**N)
        S += N*np.log(M+1) - np.log(np.sum(np.exp(S)))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

0.1.2 Error in the log density of states

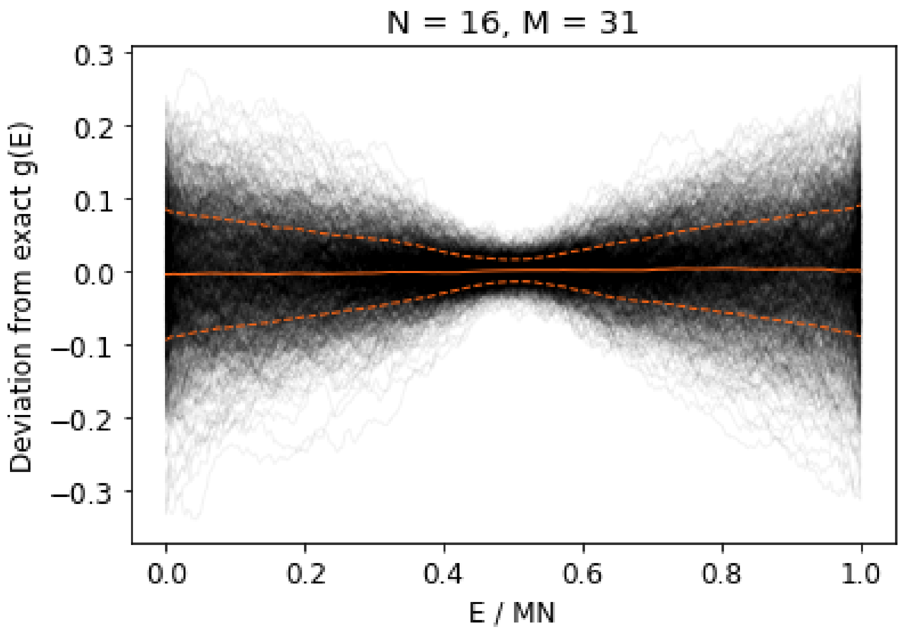
```
xEs, xgs = exact_bw_gs(N, M)
xlng = np.log(xgs)
```

```
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 / (M*N), np.log(xgs), label='Exact')
plt.plot(Es / (M*N), mean_lng, label='Mean')
plt.xlabel('E / MN')
plt.ylabel('Density of states g(E)')
plt.legend();
```

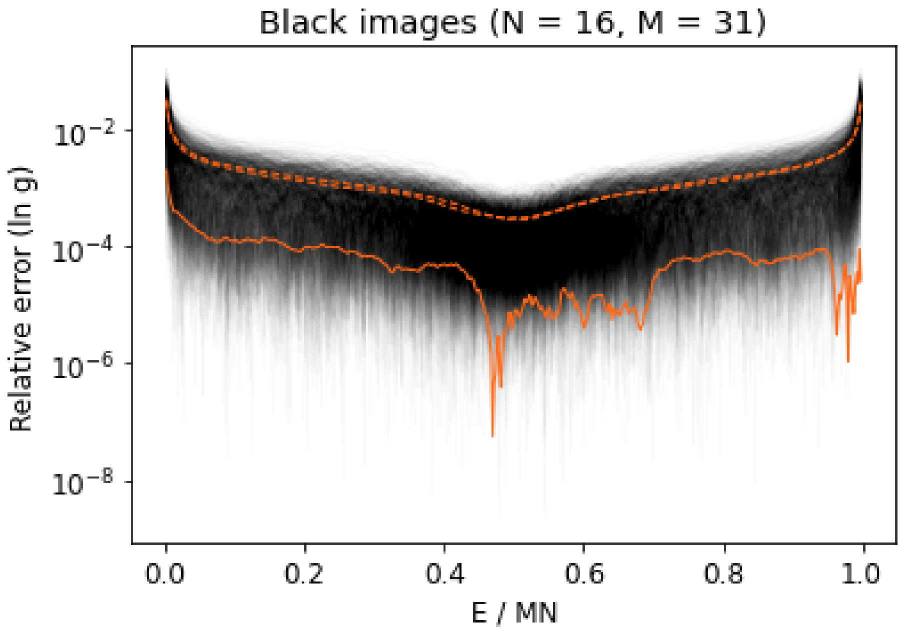



```
for lng in map(file_lngs, paths):
    plt.plot(Es / (M*N), lng - xlng, 'black', alpha=0.05, linewidth=1)
plt.plot(Es / (M*N), mean_lng - xlng, '#ff6716', linewidth=1)
plt.plot(Es / (M*N), (mean_lng - std_lng) - xlng, '#ff6716', linestyle='dashed', linewidth=1)
plt.plot(Es / (M*N), (mean_lng + std_lng) - xlng, '#ff6716', linestyle='dashed', linewidth=1)
plt.title('N = {}, M = {}'.format(N, M))
plt.xlabel('E / MN')
plt.ylabel('Deviation from exact g(E)')
plt.savefig('wanglandau-bw-deviation.png', dpi=600);
```




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

```python
for lng in map(file_lngs, paths):
    plt.plot(Es / (M*N), relerror(lng), 'black', alpha=0.02, linewidth=1)
plt.plot(Es / (M*N), relerror(mean_lng), '#ff6716', linewidth=1)
plt.plot(Es / (M*N), relerror(mean_lng - std_lng), '#ff6716', linestyle='dashed', linewidth=1)
plt.plot(Es / (M*N), relerror(mean_lng + std_lng), '#ff6716', linestyle='dashed', linewidth=1)
plt.title('Black images (N = {}, M = {})'.format(N, M))
plt.xlabel('E / MN')
plt.ylabel('Relative error (ln g)')
plt.yscale('log')
plt.savefig('wanglandau-bw-relerror.png', dpi=600);
```

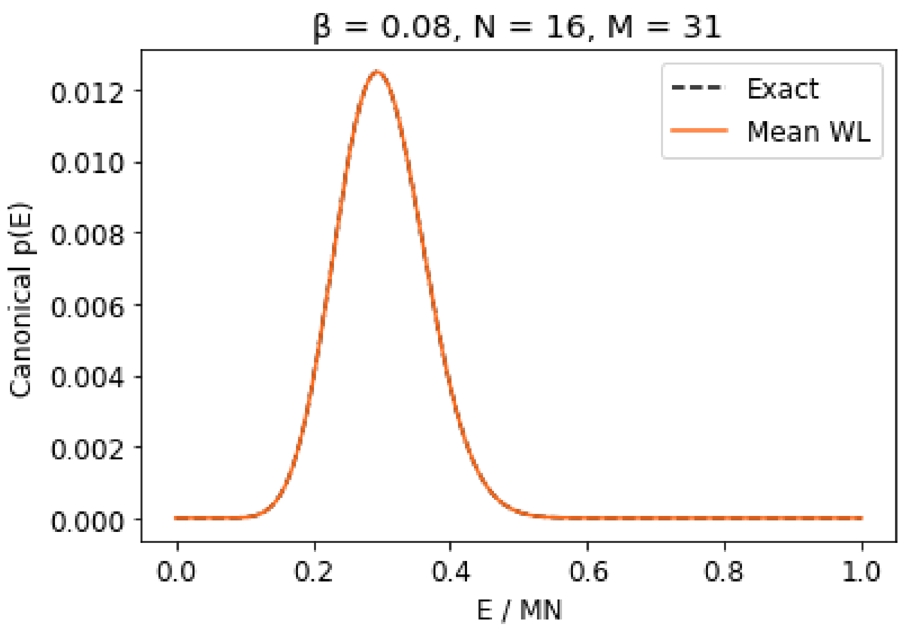


0.1.3 Error in canonical ensemble variables

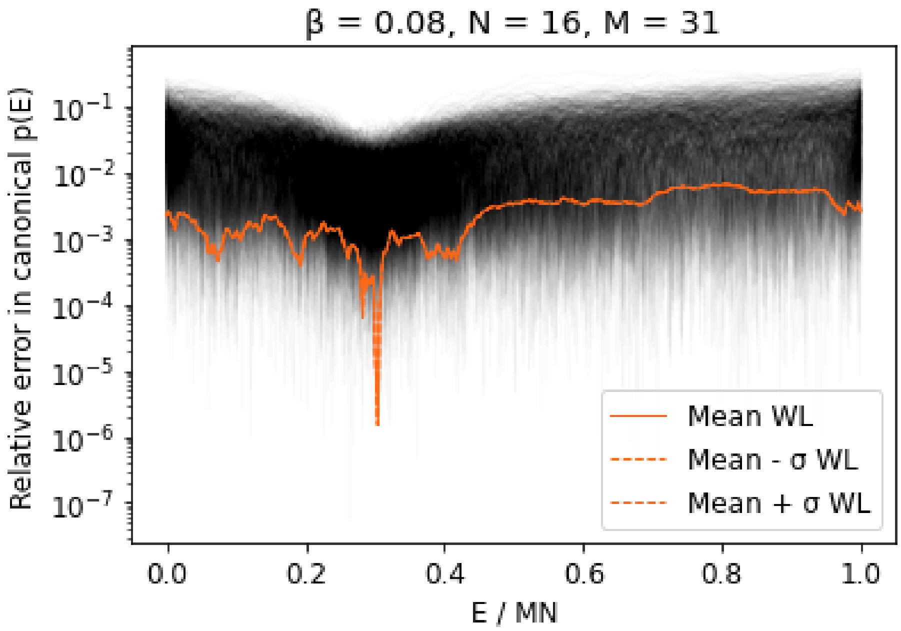
```
βs = np.exp(np.linspace(-7, 4, 500))
exact_ens = canonical.Ensemble(Es, xlng, 'Exact')
mean_ens = canonical.Ensemble(Es, mean_lng, 'Mean WL')
mσ_ens = canonical.Ensemble(Es, mean_lng, 'Mean - σ WL')
pσ_ens = canonical.Ensemble(Es, mean_lng, 'Mean + σ WL')
```

The canonical distribution for fixed β.

```
βc = 8e-2
plt.plot(Es / (M*N), exact_ens.p(βc), 'black', label=exact_ens.name, linestyle='dashed')
plt.plot(Es / (M*N), mean_ens.p(βc), '#ff6716', label=mean_ens.name)
plt.title('β = {}, N = {}, M = {}'.format(βc, N, M))
plt.xlabel("E / MN")
plt.ylabel("Canonical p(E)")
plt.legend();
```

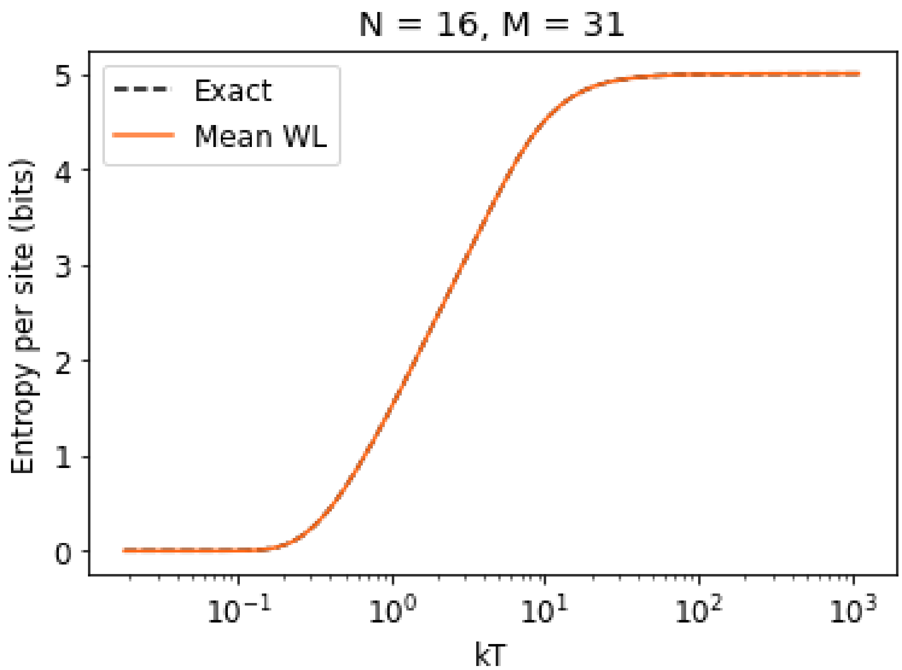



```
for lng in map(file_lngs, paths):
    ens = canonical.Ensemble(Es, lng)
    plt.plot(Es / (M*N), relerror(ens.p(βc), exact_ens.p(βc)),
             'black', alpha=0.02, linewidth=1)
plt.plot(Es / (M*N), relerror(mean_ens.p(βc), exact_ens.p(βc)),
             '#ff6716', linewidth=1, label=mean_ens.name)
plt.plot(Es / (M*N), relerror(mσ_ens.p(βc), exact_ens.p(βc)),
             '#ff6716', linewidth=1, linestyle='dashed', label=mσ_ens.name)
plt.plot(Es / (M*N), relerror(pσ_ens.p(βc), exact_ens.p(βc)),
             '#ff6716', linewidth=1, linestyle='dashed', label=pσ_ens.name)
plt.title('β = {}, N = {}, M = {}'.format(βc, N, M))
plt.xlabel("E / MN")
plt.ylabel("Relative error in canonical p(E)")
plt.yscale('log')
plt.legend();
```



Entropy

```
plt.plot(1 / βs, exact_ens.entropy(βs) / (N*np.log(2)), 'black', label=exact_ens.name, linestyle='dashed')
plt.plot(1 / βs, mean_ens.entropy(βs) / (N*np.log(2)), '#ff6716', label=mean_ens.name)
plt.xlabel("kT")
plt.xscale('log')
plt.ylabel("Entropy per site (bits)")
plt.title('N = {}, M = {}'.format(N, M))
plt.legend()
plt.savefig('wanglandau-bw-S.png', dpi=600)
```




```
exact_ens.entropy(0) / N
```

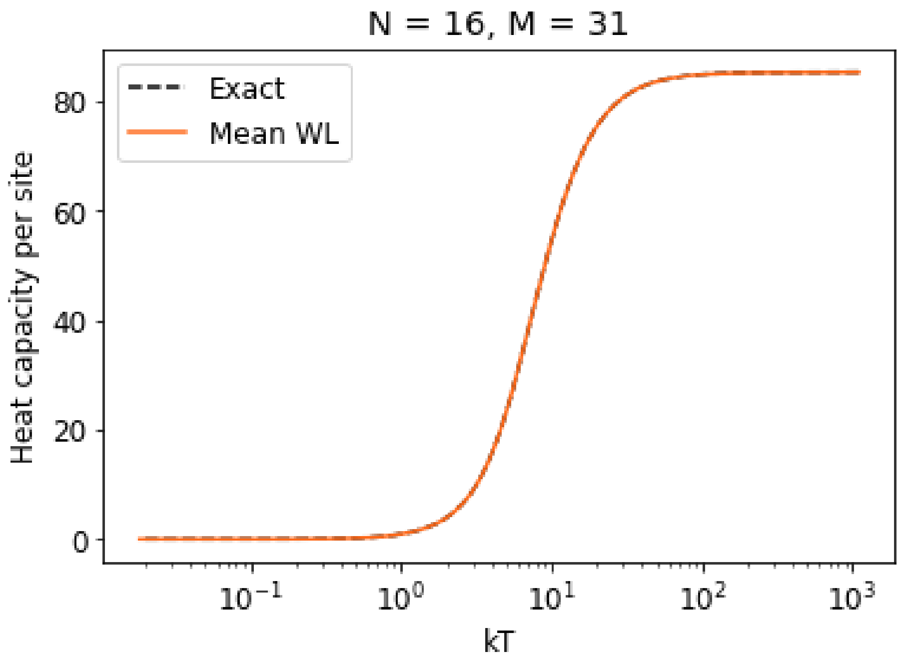
```
array([3.4657359])
```

```
exact_ens.entropy(0) / (N*np.log(2))
```

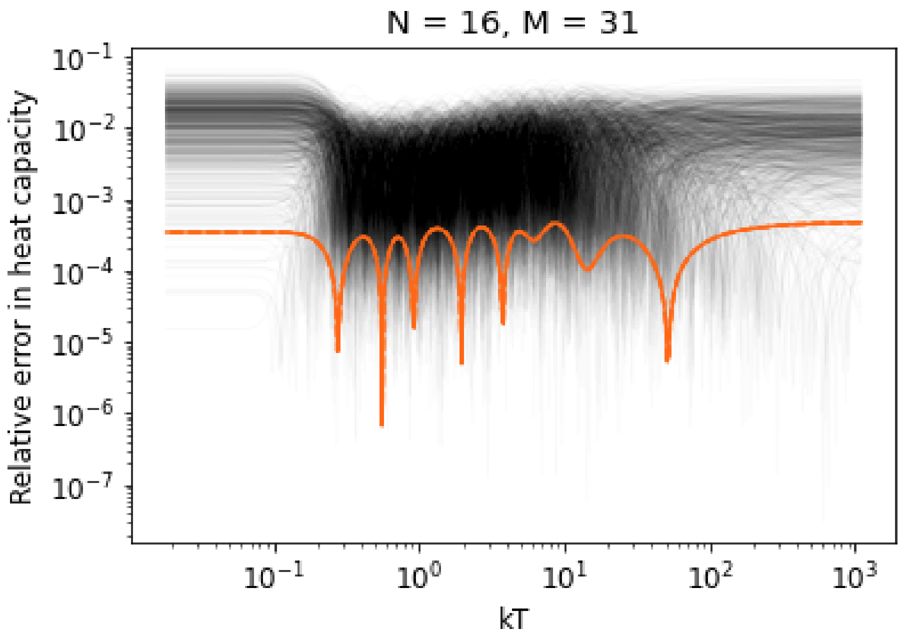
```
array([5.])
```

The relative error in the heat capacity provides a stringent test of the results.

```
plt.plot(1 / βs, exact_ens.heat_capacity(βs) / N, 'black', label=exact_ens.name, linestyle='dashed')
plt.plot(1 / βs, mean_ens.heat_capacity(βs) / N, '#ff6716', label=mean_ens.name)
plt.xlabel("kT")
plt.xscale('log')
plt.ylabel("Heat capacity per site")
plt.title('N = {}, M = {}'.format(N, M))
plt.legend()
plt.savefig('wanglandau-bw-C.png', dpi=600)
```




```python
for lng in map(file_lngs, paths):
    ens = canonical.Ensemble(Es, lng)
    plt.plot(1 / βs, relerror(ens.heat_capacity(βs), exact_ens.heat_capacity(βs)),
             'black', alpha=0.02, linewidth=1)
plt.plot(1 / βs, relerror(mean_ens.heat_capacity(βs), exact_ens.heat_capacity(βs)),
             '#ff6716', label=mean_ens.name)
plt.plot(1 / βs, relerror(mσ_ens.heat_capacity(βs), exact_ens.heat_capacity(βs)),
             '#ff6716', linestyle='dashed', label=mσ_ens.name)
plt.plot(1 / βs, relerror(pσ_ens.heat_capacity(βs), exact_ens.heat_capacity(βs)),
             '#ff6716', linestyle='dashed', label=pσ_ens.name)
plt.xlabel('kT')
plt.xscale('log')
plt.ylabel('Relative error in heat capacity')
plt.yscale('log')
plt.title('N = {}, M = {}'.format(N, M))
plt.savefig('wanglandau-bw-C-relerror.png', dpi=600)
```


N = 16, M = 31
Relative error in heat capacity
kT
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{0}
10^{1}
10^{2}
10^{3}