1 Ising images

```
import numpy as np
import numpy.linalg as linalg
import matplotlib.pyplot as plt
from PIL import Image, ImageFilter, ImageOps
import imageio
plt.rcParams['image.cmap'] = 'gray'
```

```
from ipywidgets import IntProgress
from IPython.display import display
import time
```

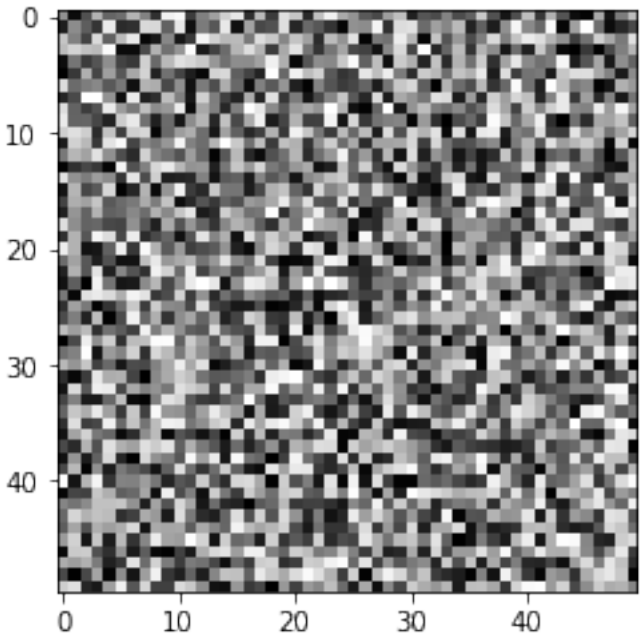
1.1 Standard Ising (on a torus)

In grayscale for fun.

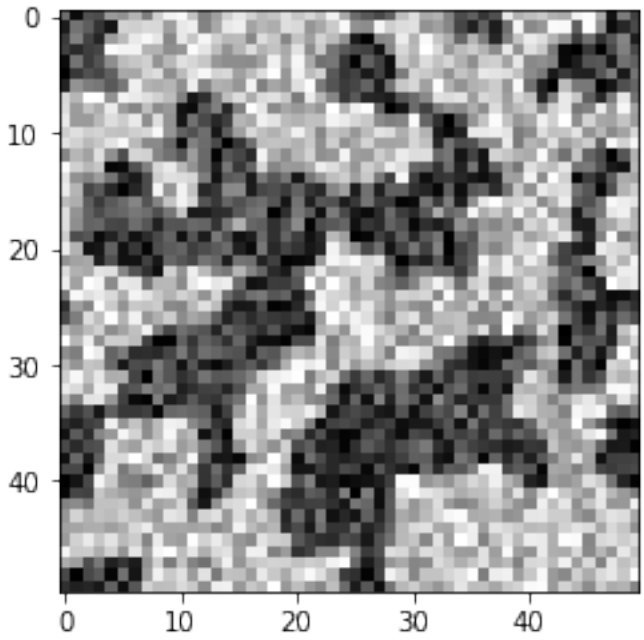
```
def neighbors(a, i, j):
    return np.hstack([a[:,j].take([i-1,i+1], mode='wrap'),
                      a[i,:].take([j-1,j+1], mode='wrap')])
```

```
def energy(img, i, j):
    return -1 + np.sum(np.abs(img[i, j] - neighbors(img, i, j)))

def isingstep(β, img):
    w, h = np.shape(img)
    i = np.random.randint(w)
    j = np.random.randint(h)
    E0 = energy(img, i, j)
    img[i, j] *= -1
    E1 = energy(img, i, j)
    P = np.exp(-β*(E1 - E0)) if E1 > E0 else 1
    if np.random.rand() > P: # Restore old
        img[i, j] *= -1
    return img
```

```
img = 2*np.random.rand(50, 50) - 1
plt.imshow(img);
```




```
n = 100000
for i in range(n):
    isingstep(3 * (np.pi / 2) / np.arctan(n - i), img)
plt.imshow(img);
```



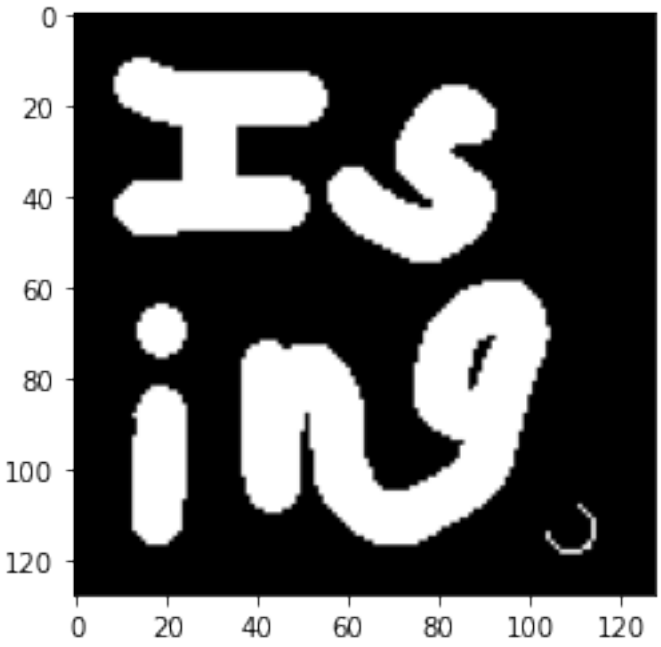
1.2 Image-edge Ising

```
edges = Image.open("ising-edges.png")
edata = np.array(edges) > 128
edges
```

```
def eenergy(img, edges, i, j):
    """Edge-modified Ising energy: 0 on edge."""
    if edges[i, j]:
        return 0
    w, h = np.shape(img)
    c = img[i, j]
    l = img[i-1, j] if i > 0   else img[w-1, j]
    r = img[i+1, j] if i < w-1 else img[0, j]
    t = img[i, j-1] if j > 0   else img[i, h-1]
```

```
    b = img[i, j+1] if j < h-1 else img[i, 0]
    return -img[i, j] * (l + r + t + b)

def nenergy(img, edges, i, j):
    """Neighbor-modified Ising energy: 0 interactions with edges."""
    if edges[i, j]:
        return 0

    w, h = np.shape(img)
    c = img[i, j]
    l = r = t = b = 0
    if i > 0:
        l = img[i-1, j] if not edges[i-1, j] else 0
    else:
        l = img[w-1, j] if not edges[w-1, j] else 0

    if i < w - 1:
        r = img[i+1, j] if not edges[i+1, j] else 0
    else:
        r = img[0, j] if not edges[0, j] else 0

    if j > 0:
        t = img[i, j-1] if not edges[i, j-1] else 0
    else:
        t = img[i, h-1] if not edges[i, h-1] else 0

    if j < h - 1:
        b = img[i, j+1] if not edges[i, j+1] else 0
    else:
        b = img[i, 0] if not edges[i, 0] else 0

    return -img[i, j] * (l + r + t + b)

def eisingstep(β, img, edges):
    w, h = np.shape(img)
    i = np.random.randint(w)
    j = np.random.randint(h)
    E0 = nenergy(img, edges, i, j)
    img[i, j] *= -1
    E1 = nenergy(img, edges, i, j)
    P = np.exp(-β*(E1 - E0)) if E1 > E0 else 1
    if np.random.rand() > P: # Restore old
        img[i, j] *= -1
    return img

def frame(writer, data):
    writer.append_data((255 * ((eimg + 1) / 2)).astype('uint8'))
```

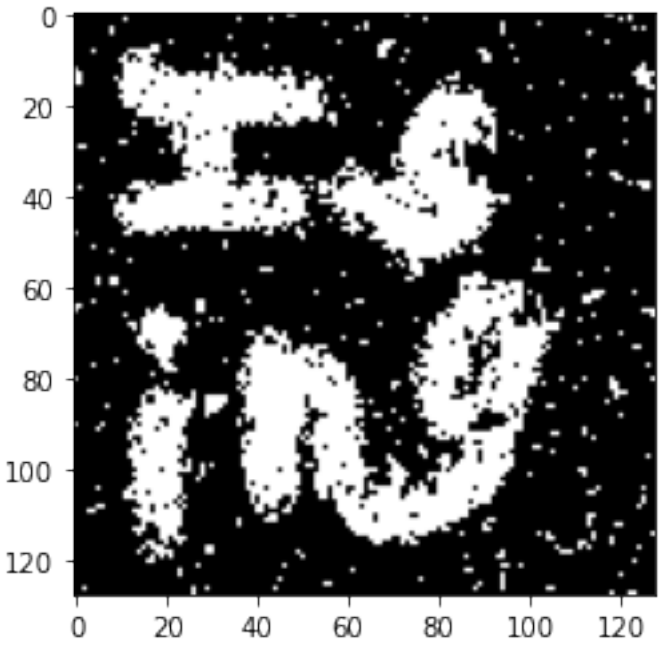
```
img = Image.open("ising-letters.png")
eimg = -1 + 2 * (np.array(img) / 255)
plt.imshow(eimg);
```



movie.gif: Full neighbor Ising.

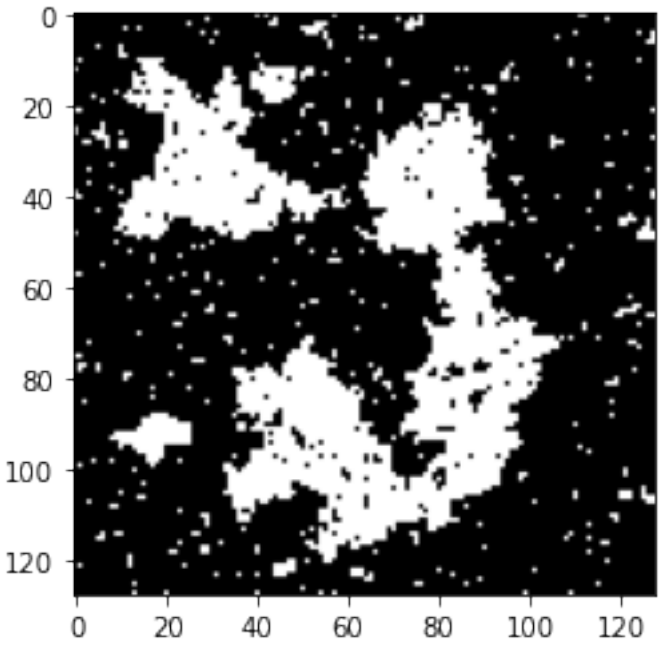
```
n = 1000000
f = IntProgress(min=0, max=1 + (n-1) // 1000) # instantiate the bar
display(f)
with imageio.get_writer('movie.gif', mode='I') as writer:
    frame(writer, eimg)
    for i in range(n):
        eisingstep(0.5 * (np.pi / 2) / np.arctan(n - i), eimg, edata)
        if i % 1000 == 0:
            f.value += 1
            frame(writer, eimg)
plt.imshow(eimg);
```

```
IntProgress(value=0, max=1000)
```



`imovie.gif`: Normal Ising.

```
n = 1000000
img = eimg
with imageio.get_writer('imovie.gif', mode='I') as writer:
    frame(writer, img)
    for i in range(n):
        isingstep(0.5 * (np.pi / 2) / np.arctan(n - i), img)
        if i % 1000 == 0:
            frame(writer, img)
plt.imshow(img);
```



1.3 Image-metric Ising

```
# def takewrap(a, i, j, xs=np.arange(-1, 2), ys=np.arange(-1, 2)):
def takewrap(a, i, j, xs=np.arange(0, 1), ys=np.arange(0, 1)):
    return np.array([x for v in a.take(xs+i, axis=0, mode='wrap')
                       for x in v.take(ys+j, mode='wrap')])
```

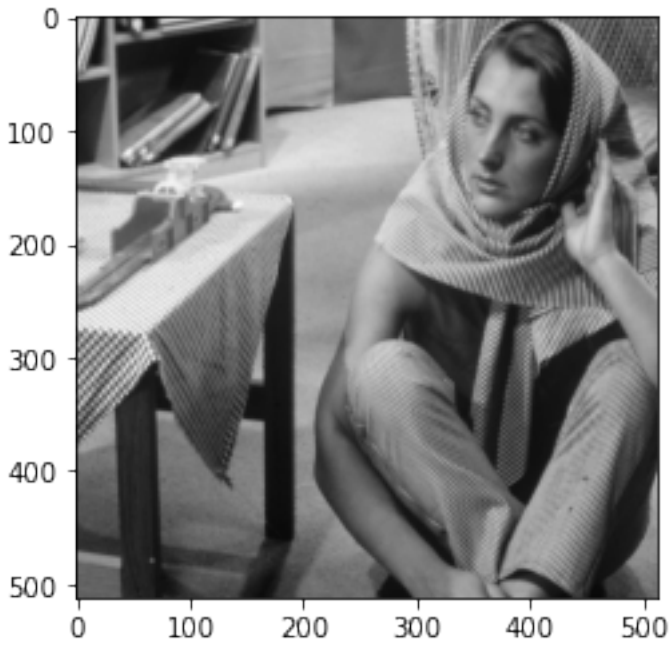
1.3.1 Unrestricted swapping motion

Swapping preserves the intensity distribution.

```
def sienergy(img, init, i, j):
    """Inversion-symmetric image energy"""
    eq = takewrap(img, i, j) == takewrap(init, i, j)
    return -np.abs(np.sum(2*eq - 1))

def ienergy(img, init, i, j):
    """Image energy based on 3x3 block deviation"""
    return np.abs(init[i, j] - img[i, j])

def swisingstep(β, img, edges):
    w, h = np.shape(img)
    i0 = np.random.randint(w)
    i1 = np.random.randint(w)
    j0 = np.random.randint(h)
    j1 = np.random.randint(h)
    E0 = ienergy(img, edges, i0, j0) + ienergy(img, edges, i1, j1)
    img[i0, j0], img[i1, j1] = img[i1, j1], img[i0, j0]
    E1 = ienergy(img, edges, i0, j0) + ienergy(img, edges, i1, j1)
```

```
    P = np.exp(-β*(E1 - E0)) if E1 > E0 else 1
    if np.random.rand() > P: # Restore old
        img[i0, j0], img[i1, j1] = img[i1, j1], img[i0, j0]
    return img

def nnisingstep(β, img, edges):
    w, h = np.shape(img)
    i0 = np.random.randint(w)
    i1 = int((i0 + np.sign(np.random.rand() - 1/2)) % w)
    j0 = np.random.randint(h)
    j1 = int((j0 + np.sign(np.random.rand() - 1/2)) % h)
    E0 = ienergy(img, edges, i0, j0) + ienergy(img, edges, i1, j1)
    img[i0, j0], img[i1, j1] = img[i1, j1], img[i0, j0]
    E1 = ienergy(img, edges, i0, j0) + ienergy(img, edges, i1, j1)
    P = np.exp(-β*(E1 - E0)) if E1 > E0 else 1
    if np.random.rand() > P: # Restore old
        img[i0, j0], img[i1, j1] = img[i1, j1], img[i0, j0]
    return img
```

```
img = Image.open("barbara.png")
eimg = -1 + 2 * (np.array(img) / 255)
initimg = eimg.copy()
plt.imshow(initimg);
```

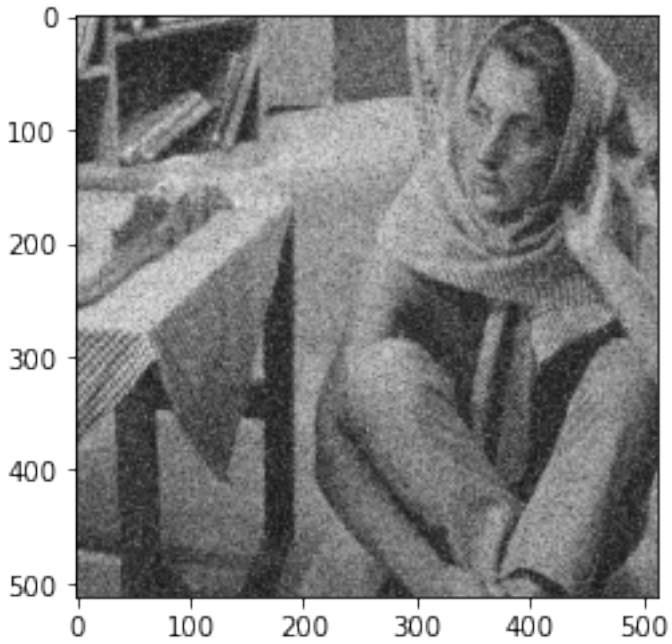


`swmovie.gif`: Image metric Ising (arbitrary swaps with `ienergy`).

```
n = 2000000
f = IntProgress(min=0, max=(1 + (n-1) // 1000)) # instantiate the bar
display(f)
with imageio.get_writer('swmovie.gif', mode='I') as writer:
```

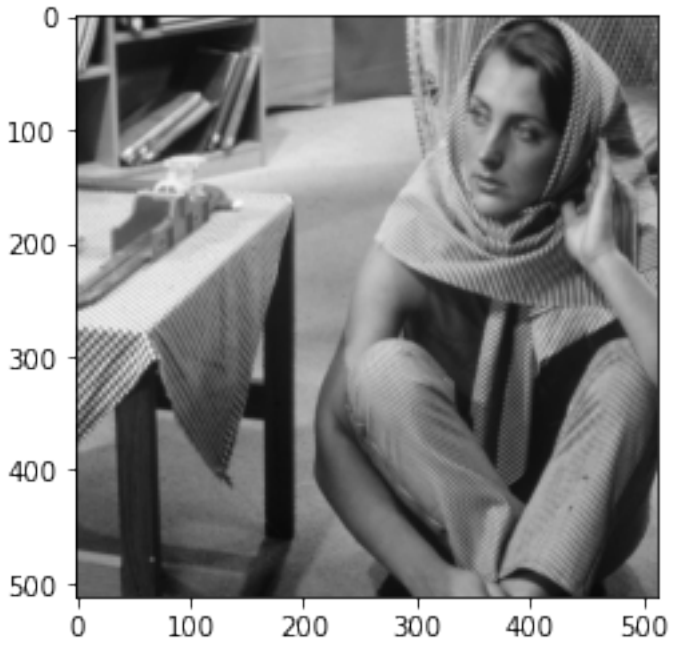
```
    frame(writer, eimg)
    for i in range(n):
        k = i/n
        swisingstep(3, eimg, initimg)
        if i % 1000 == 0:
            f.value += 1
            frame(writer, eimg)
#     for i in range(n):
#         k = i/n
#         swisingstep(4*(1 - k) + 1e-3*k, eimg, initimg)
#         if i % 1000 == 0:
#             f.value += 1
#             frame(writer, eimg)
#     for i in range(n):
#         k = i/n
#         swisingstep(1e-3*(1 - k) + 4*k, eimg, initimg)
#         if i % 1000 == 0:
#             f.value += 1
#             frame(writer, eimg)

plt.imshow(eimg);
```

```
IntProgress(value=0, max=2000)
```



1.3.2 Nearest-neighbor swapping motion

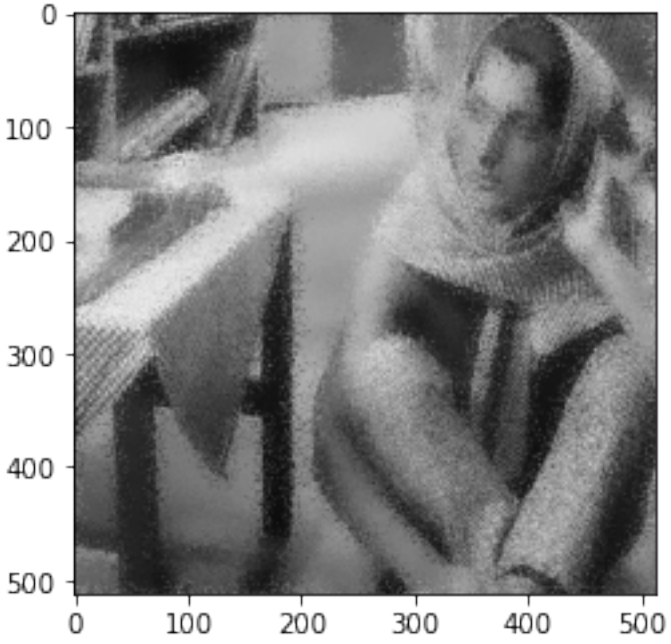
```
img = Image.open("barbara.png")
eimg = -1 + 2 * (np.array(img) / 255)
initimg = eimg.copy()
plt.imshow(initimg);
```



nnmovie.gif: Image metric Ising (neighborly swaps with ienergy).

```
n = 2000000
f = IntProgress(min=0, max=3*(1 + (n-1) // 1000)) # instantiate the bar
display(f)
with imageio.get_writer('nnmovie.gif', mode='I') as writer:
    frame(writer, eimg)
    for i in range(n):
        k = i/n
        nnisingstep(5*(1 - k) + 1e-4*k, eimg, initimg)
        if i % 1000 == 0:
            f.value += 1
            frame(writer, eimg)
    for i in range(n):
        nnisingstep(1e-4, eimg, initimg)
        if i % 1000 == 0:
            f.value += 1
            frame(writer, eimg)
    for i in range(n):
        k = i/n
        nnisingstep(1e-4*(1 - k) + 5*k, eimg, initimg)
        if i % 1000 == 0:
            f.value += 1
            frame(writer, eimg)

plt.imshow(eimg);
```

```
IntProgress(value=0, max=6000)
```


0
100
200
300
400
500
0
100
200
300
400
500