Conformal invariance

(To be included in the next edition of Sethna, "Entropy, Order Parameters, and Complexity")

The Ising model on a square lattice has an emergent rotation invariance as well as an emergent scale invariance. The complex patterns of up and down spins look the same on long length scales also when rotated by an angle. Indeed, making use of the symmetries under changes of length scale, position, and angle (plus one spatially nonuniform transformation), systems at their critical points have a conformal symmetry group.

In two dimensions, the conformal symmetry group becomes huge. Roughly speaking, any complex analytic function $f(z) = u(x + iy) + iv(x + iy)$ takes a snapshot of an Ising model $M(x, y)$ and warps it into a new magnetization pattern at (u, v) that 'looks the same'. (Here u, v, x, and y are all real.)

You may remember that most ordinary functions (like z^2, $\sqrt{z}$, $\sin(z)$, $\log(z)$, and $\exp(iz)$) are analytic, and all of them yield cool transformations of the Ising model -- weird and warped when magnified until you see the pixels, but recognizably Ising-like on long scales.

(a) What analytic function shrinks a region uniformly by a factor b, holding $z = 0$ fixed? What analytic function translates the lattice by a vector $r_0 = (u_0, v_0)$? What analytic function rotates a region by an angle θ? Expanding $f(z + \delta) = f(z) + \delta \partial f / \partial z$, show that an analytic function f to linear order in δ is a rotation and dilation about z followed by a translation. What complex number gives the net translation?

In the renormalization group, we first coarse grain the system (shrinking by a factor b) and then rescale the magnetization (by some power b^{y_M}) in order to return to statistically the same critical state: $\widehat{M} = b^{y_M} M$. This rescaling turns the larger pixels more gray; a mostly up-spin 'white' region with tiny pixels is mimicked by a large single pixel with the statistically averaged gray color.

We can discover the correct power for M by examining the rescaling of the correlation function $C(r) = \langle M(x)M(x + r)\rangle$. In the Ising model at its critical point the correlation function $C(r) \sim r^{-(2-d+\eta)}$. In dimension $d = 2$, $\eta = 1/4$. We expect that the correlation function for the conformally transformed magnetization will be the same as the original correlation function.

(b) If we coarse-grain by a uniform factor b, what power of b must we multiply M by to make $C(r) = \langle \widehat{M}(f(z))\widehat{M}(f(z) + r)\rangle$?

When our conformal transformation takes a pixel at $M(z)$ to a warped pixel of area A at $f(z)$, it rescales the magnetization by $\widehat{M}(f(z)) = |A|^{-1/16} M(z)$. The pixel area for a locally uniform compression by b changes by $|df/dz|^2 = 1/b^2$. You may use this to check your answer to part~(b).

Now we proceed to some computations. Load up the packages.

```
In [ ]: %pylab
        from scipy import *
        from scipy.misc import imread
        from matplotlib.collections import PolyCollection
```

Now load a snapshot of the Ising model, replacing the directory shown with your local directory and the filename with your image file. Transform the image into an array

```
In [ ]: # Load image
        Simage = imread("Snapshots/Tc.png", flatten=True)
        # Image is stored as gray scale (0,255). Ising uses down=0->-1, up=255->1.
        S = 2*Simage/255 - 1
        figure()
        imshow(S, interpolation='nearest', cmap='gray',origin='lower')
```

'Transform' selects an L x L region from S, generates a list of spins and square polygons covering the unit square in the complex plane, transforms the squares into quadrilaterals under the map f(z), rescales the magnetization S according to the quadrilateral area by $A^{-1/16}$, converts the quadrilaterals back into real coordinates, and returns the warped polygons and rescaled magnetization.

In [ ]:
```
def Transform(S,f,L=None,**fkwargs):
    """Returns polygons and rescaled spins, given original spin lattice S,
 distorted
    according to function f(z)"""

    # L is length of corner of lattice to be analyzed
    # Use small L to debug your code and also to explore by zooming,
    # before settling on final plots
    if L is None:
#       L = len(S)
        L = 512
    # Unpack L x L lower-right-hand corner of 2D spin array into 1D
    spins = S[:L,-L:].flatten()

    # Map lattice to complex plane (0,1] x (0,1]j.
    latticeCenters = (1./L)*array([(m+0.5)+1.j*(n+0.5) for n in range(L) fo
r m in range(L)])

    # One pixel in the original complex lattice
    pixelExtent = (1./L)*array([-0.5-0.5j,-0.5+0.5j,0.5+0.5j,0.5-0.5j])

    # All the squares
    undeformedSquares = array([center+pixelExtent for center in latticeCent
ers])

    # Distort quadrilaterals under f(z)
    # fkwargs passes optional constants like "b=0.25" to your function f
    dQ = f(undeformedSquares,**fkwargs)

    # Find areas of quadrilaterals using the 'shoelace formula' for the are
a of a polygon.
    areas = -0.5*sum(dQ*np.roll(dQ.conj(),1,axis=1),axis=1).imag

    # Rescale spins by corresponding areas
    rescaledSpins = areas**(-1/16.)*spins

    # Drop spins whose quadrilaterals go to infinity, or which have negativ
e areas
    # (which have a point in the interior at infinity, and should be 'fille
d' on the outside)
    dQ = dQ[isfinite(areas)&(areas>0.)]
    rescaledSpins = rescaledSpins[isfinite(areas)&(areas>0.)]

    # Return to real coordinates
    quads = transpose(array([dQ.real,dQ.imag]),axes=(1,2,0))

    return quads, rescaledSpins
```

GraphQuads plots the resulting magnetization pattern, given the function $f(z)$ and the length. Note that $\log(z)$, for example, is singular at the grid point (0,0); GraphQuads and Transform will send RuntimeWarning messages about such events. (We drop those infinite-area polygons before we graph.)

In [ ]:
```
def GraphQuads(S,f,L=None,**fkwargs):
    quads, rescaledSpins = Transform(S,f,L,**fkwargs)
    fig, ax = plt.subplots()
    coll = PolyCollection(quads, array=rescaledSpins, cmap='gray', \
                          edgecolors='none',antialiased=False)
    ax.add_collection(coll)
    ax.axis('scaled')
    plt.show()
```

For example, try $f(z) = \log(z)$ with a small L, to explore.

In [ ]:
```
GraphQuads(S,log,64)
```

Define your own function, like

```
def fexp(z): return exp(2.j * pi * z)
```

or

```
def fTwoProteins(z):
    u=-0.02
    w = exp(2*pi*z)
    return (w+u)/(u*w+1)
```

Zoom in to a square region. Find an interesting region; one with squares that vary in orientation, and in length by at least a factor of two. (The region x=(-0.5,0.),y=(-0,0.5) for log above is a bit too small.)

Run now for large L, and save the whole plot and the zoomed plot to print and hand in.

(c) Discuss your zoomed plot critically - does it appear that the Ising correlations and visual structures have been faithfully retained by your analytic transformation?

(d) Load an Ising model equilibrated at T=100, and at T=3, and a coarsened Ising model quenched to low temperatures with a coarsening length of 4-5 pixels. Distort and zoom with each. Are the correlations and visual structures retained away from the critical point? (T100.png, T3.png, and Tcoarsening.png are available.)

(e) Invent a non-analytic function, and use it to distort your Ising model. (The author tried two methods: inventing functions $u(x, y)$ and $v(x, y)$, and using the real part of $f(z)$ and the imaginary part of $g(z)$.) Find an example that makes for an interesting picture. As above, examine the latter critically - does it appear that the Ising correlations and visual structures have been faithfully retained by your analytic transformation? Describe the distortions you see. (Are the pixels still approximately square?)

In [ ]:
```
```