Conformal invariance

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

Notice that Mathematica's notebook has trouble with big files. Use Python. Or, if you prefer -- save often, try not using a second display (a known problem), and try not opening other programs in the background (especially second Mathematica files). Debug everything with small systems, and then directly save your graphs of larger systems to a file (no bigger than 512x512, probably smaller) and look at them outside Mathematica.

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 $f(z) = u(x+I\,y) + I\,v(x+I\,y)$ takes a snapshot of the 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], Exp[z], ...) are analytic, and all of them yield cool transformations of the Ising model -- weird and warped when you look at the deformed pixels, but recognizably Ising-like on long scales.

(a) What analytic function dilates 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: $\hat{M} = b^{y_M} M$. 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 factor 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 $C(r) = \langle \hat{M}(f(z))\,\hat{M}(f(z)+r)\rangle$?

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

$$M(f(z)) = |A|^{-1/16} M(z).$$

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).

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 $S_{mn} = \pm 1$.

```
SetDirectory["/Users/sethna/Teaching/6562/Exams/Mathematica/Snapshots"];
Simage = Import["Tc.png"];
S01 = ImageData[ColorSeparate[Simage][[1]]];
S = 2 * S01 - 1;
```

Given the vertices in the complex plane, this gives the area. It is derived from the 'shoelace formula' for the area of the polygon (which uses real-valued 2D vertices).

```
area[complexQuadrilateral_] :=
 (1/2) Im[Dot[RotateRight[complexQuadrilateral], Conjugate[complexQuadrilateral]]]
```

'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.

```
Transform[S_, f_, L_] := Module[{spins, latticeCenters, pixelExtent},
  (* Unpack lower right corner of 2D spin array to 1D list *)
  spins = Flatten[Table[S[[m, n]], {m, 1, L}, {n, Length[S] - L + 1, Length[S]}]];
  (* Spins at lattice centers;
  build array of squares, filling [0,1] + I [0,1] *)
  latticeCenters = Flatten[(1/L) Table[(m - 0.5) + I (n - 0.5), {m, 1, L}, {n, 1, L}]];
  pixelExtent = (1/L) {-0.5 - 0.5 I, -0.5 + 0.5 I, 0.5 + 0.5 I, 0.5 - 0.5 I};
  undeformedSquares = Table[center + pixelExtent, {center, latticeCenters}];
  (* Warp squares into quadrilaterals under f[z] *)
  deformedComplexQuadrilaterals = f[undeformedSquares];
  (* Find areas *)
  areas = Map[area, deformedComplexQuadrilaterals];
  (* Drop spins whose quadrilaterals go to infinity,
  or which have negative areas (which have a point in the interior at infinity,
    and should be'filled' on the outside) *)
  combo = Transpose[{areas, deformedComplexQuadrilaterals, spins}];
  comboPruned = Select[combo, NumberQ[#[[1]]] && #[[1]] > 0 &];
  {areas, deformedComplexQuadrilaterals, spins} = Transpose[comboPruned];
  (* Rescale spins based on area of final polygon *)
  rescaledSpins = areas^(-1/16.) spins;
  (* Convert range of spins to (-1,1) for gray scale *)
  rescaledSpins /= Max[Max[rescaledSpins], -Min[rescaledSpins]];
  (* Return to real coordinates *)
  deformedQuadrilaterals =
   Map[{Re[#], Im[#]} &, deformedComplexQuadrilaterals, {2}];
  (* Return quadrilaterals, rescaled spins *)
  {deformedQuadrilaterals, rescaledSpins}]
```

GraphQuads plots the resulting magnetization pattern, given the function f[z] and the length. Zoom with the optional fourth argument; save the file (rather than showing it in the notebook) with the fifth argu-

ment.

Warning: My machine has trouble with notebook graphics with L>128. (It may have to do with a second monitor.) I recommend exploring smaller systems first with L=64. (What function f[z]? How much to zoom?) Then output larger L as a file, and view with a separate program.

```
GraphQuads[S_, f_, L_, plotRange_: {All, All}, fileName_: " ", axes_: True] :=
 Block[{quads, rescaledSpins, g}, {quads, rescaledSpins} = Transform[S, f, L];
  g = Graphics[Table[{GrayLevel[(rescaledSpins[[n]] + 1) / 2], Polygon[quads[[n]]]},
     {n, Length[quads]}], PlotRange → plotRange, Axes → axes];
  If[fileName ≠ " ", Export[fileName, g], Show[g]]]
```

For example, try f[z] = Log[z] with a small L.

```
GraphQuads[S, Log, 64]
```

Define your own function, like ***fexp[z_] := Exp[2πI z]*** also written in shorthand as ***Exp[2π I #]&*** . Another example is

```
fTwoProteins[z_] := Module[{u = -0.02, w}, w = Exp[2 π z]; (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 I chose for Log is not really interesting enough, even for L=512, but takes less computer space to store for big L.)

```
GraphQuads[S, Log, 64, {{-0.5, 0}, {0, 0.5}}]
```

Save your result as a file for large L, to print and hand in. For example, let's save the whole figure as a png (to save space), and the zoomed-in figure as a pdf (to allow you to zoom in with your pdf browser).

```
GraphQuads[S, Log, 512, {All, All}, "Log512.png"]
```

```
"Log512.png"
```

```
GraphQuads[S, Log, 512, {{-0.5, 0}, {0, 0.5}}, "Log512Zoom.pdf"]
```

(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?

(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?)