NPFRGHintsPython

May 4, 2024

RG flows from non-perturbative coarse graining

(Sethna, "Entropy, Order Parameters, and Complexity", ex. XXX)

This exercise is primarily analytical: only those parts with computational components are included in this file. The exercise will be available at https://sethna.lassp.cornell.edu/StatMech/SethnaExercises.pdf.

[ ]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
```

… See exercise for long intro to non-perturbative RG and momentum-space RG…

Sparing ourselves this calculation, we shall quote the LPA approximation for the coarse-graining equations up to quartic terms:

$$\begin{aligned} dF/d\ell &= -k^3/(1+(A/k^2)) \\ dA/d\ell &= 3Gk/(1+(A/k^2))^2 \\ dG/d\ell &= -18(G^2/k)/(1+(A/k^2))^3 \\ dk/d\ell &= -k. \end{aligned} \tag{1}$$

These are not yet the renormalization-group flow equations: we have yet to rescale properly to see the fixed point. But we can use these as if we were the experimentalist, measuring universal power laws in the "laboratory'' provided by $V_\infty(M_\infty, A_\infty, G_\infty, F_\infty)$.

We shall focus on the transition for $G_0 = 0.2$, where the critical point lies at $A_0^c \approx -0.19053$. At the critical point, the macroscopic potential develops a double well, leading to a finite magnetization. Hence we expect $A_\infty = 0$ when A_0 equals A_0^c.

[ ]:
```
def V(M,F,A,G):
    return F + ... + (G/2)*M**4

def coarsegrainingflow(ell,vars):
    """Given vars = [A,G,F], return the derivatives [A',G',F']
    according to the coarse-graining equations."""
    k = np.exp(-ell)
    A, G, F = vars
    dAdell = 3*(G*k)/(1+(A/k**2))**2
    dGdell = -18*(G**2/k)/(1+(A/k**2))**3
    dFdell = ...
    return np.array([dAdell, dGdell, dFdell])
```

(a) Numerically solve the flow equations starting at $\ell = 0$, $k = 1$, $A_0 = A_0^c$, $G_0 = 0.2$, and $F_0 = 0$, evaluating them at $\ell \in [0, 0.25, 0.5, 1, 2, 4]$. Plot the potentials $V_\ell(M)$. Does the potential appear to be at the critical point, where the bulk quadratic term $A_\infty = 0$? Note also that the microscopic energy V_0 has a double well, but the well disappears under coarse-graining. Why should this be expected, for a low enough barrier? (Hint: The barrier height in V_0 is measured in units of the temperature. If it is much less than one, do you expect the system to stay in one well?)

```
[ ]: # Critical temperature for G=0.2

     Ac = -0.19053 # Depends on integrator: method = "DOP853"

     ellInit = 0;
     ellFinal = 4;
     ells = [0,0.25,0.5,...]
     AInit = Ac
     GInit = ...
     FInit = 0.
     Ms = np.arange(-4,4,0.02)

     sol = solve_ivp(coarsegrainingflow, [ellInit, ellFinal], [AInit, ..., FInit],
                     t_eval = ells)
     ATraj, ..., FTraj = sol.y
     for A, G, F, ell in zip(ATraj,GTraj,FTraj,ells):
         plt.plot(Ms, V(Ms, F, A, ...), label=np.around(ell,2))
     plt.legend()
     plt.title("Critical point")
     plt.ylim((-0.4,0.5))
```

Your answer here (or in a separate writeup). Double click to edit. Latex works too ($E = mc^2$).

Since our free energy $V(M)$ is measured in units of the temperature, raising the temperature lowers A, G, and F. As A directly controls the development of magnetization, let us mimic the experimental temperature by varying A_0 through the critical value A_0^c. We shall confine ourselves to the approach to A_0^c from above; more sophisticated NPFRG methods are needed to behave properly below the transition temperature.

Let us measure the macroscopic susceptibility $\chi = \partial \bar{M}/\partial H|_{H=0}$, where $\bar{M}$ minimizes $V_\infty(M) - MH$.

(b) Show that $\chi = 1/(2A_\infty)$ in the paramagnetic single-well phase when $A_0 > A_0^c$. (Hint: Find the equation satisfied by $\bar{M}$, and take its derivative with respect to H.)

Your answer here (or in a separate writeup).

(c) As in part (a), plot the potential V_ℓ as it evolves starting at $A_0^c + 0.05$. Does it converge to a fixed potential, representing the bulk free energy? Measure the bulk quadratic term A_∞ in the free energy for a range of microscopic values A_0 close to A_0^c, and estimate γ. (Go to larger ℓ as your initial temperature gets close to A_0^c. You can do a power-law fit, but also you can just do a log-log plot of $\chi(A_0) \times (A_0 - A_0^c)^{\gamma_{\mathrm{try}}}$ vs. $A_0 - A_0^c$ and vary γ_{try} until it has a flat region.) Compare your value γ_{try} to the value derived from conformal bootstrap, $\gamma_{\mathrm{boot}} \sim 1.237075$.

```
[ ]: # Above critical

     AInit = Ac + 0.05
     GInit = ...
     ...
     Ms = np.arange(-4,4,0.02)

     sol = solve_ivp(...)
     ATraj, ...
     for A, G, ...:
         plt.plot(...)
     plt.title("Above critical")
     plt.legend()
     plt.ylim((-0.4,0.5))
```

```
[ ]: deltaAInits = np.logspace(-5,1,61)
     ellFinal = 10; # Close to critical point convergence is slower
     ells = np.linspace(ellInit, ellFinal, 4)
     Amacro = {}
     chi = {}
     for deltaAInit in deltaAInits:
         # Need more accurate solver for tricky region near critical point.
         sol = solve_ivp(coarsegrainingflow, [ellInit, ellFinal],
                         [Ac+deltaAInit, GInit, FInit], t_eval=ells, method =␣
      ↪"DOP853")
         Amacro[deltaAInit] = sol.y[0][-1]
         chi[deltaAInit] = 1/(2*Amacro[deltaAInit])
     chis = [chi[deltaAInit] for deltaAInit in deltaAInits] # Copy from dictionary␣
      ↪to list
     gammaBootstrap = 1.237075;
     print("gammaBootstrap = ", gammaBootstrap)
     gammaTry = 0;    # Vary until plot is mostly horizontal?
     plt.loglog(deltaAInits, chis*deltaAInits**gammaTry)
```

Your answer here (or in a separate writeup).

Discussion ...

These rescaled equations for $d = 3$ are (finally!) our RG flow equations

$$\begin{aligned} da/d\ell &= 2a + 3g/(1+a)^2 \\ dg/d\ell &= g - 18g^2/(1+a)^3 \\ df/d\ell &= 3f - 1/(1+a). \end{aligned} \tag{2}$$

Part (d) deriving these, part (e) finding the fixed point…

What are a_0^c and g_0, the rescaled parameters corresponding to the experiments we performed in parts~(a) and~(c)? Since ℓ starts at zero with the microscopic Hamiltonian, $k_\ell = 1$, and thus the coarse-graining transition you studied happen at RG-flow initial conditions $g_0 = G_0 = 0.2$ and $a_0^c = A_0^c \approx -0.1905316$.

(f) Launch trajectories near this point, varying a_0 slightly above and below a_0^c, and plot the trajectories in the a_0, g_0 plane. Show that they pass near to the fixed point before veering off to high or low rescaled temperatures.

[ ]:
```python
def rgflow(ell,vars):
    """Given vars = [a,g,f], return the derivatives [a',g',f']
    according to the rescaled RG equations.
    Note that our ODE solver, solve_ivp, allows for a 'time-dependent' ODE,
    so we need to pass in ell even though it's not used."""
    a, g, f = vars
    dadell = 2*a + 3*g/(1+a)**2
    dgdell = g - 18*g**2/(1+a)**3
    dfdell = ...
    return np.array([dadell, dgdell, dfdell])
```

[ ]:
```python
aStar = -1./13.
gStar = 96./2197.
fStar = ...
Star = np.array([aStar, gStar, fStar])

print("Fixed point ", Star)
```

[ ]:
```python
gInit = ...;
ac = Ac

ellInit = 0;
ellFinal = 5;
ells = np.linspace(ellInit, ellFinal, 100)
aInits = np.linspace(ac-0.05,ac+0.05,11)

fInit = fStar;

sols = [solve_ivp(rgflow, [ellInit, ellFinal], [aInit, gInit, fInit],
                  t_eval = ells, method = "DOP853")
        for aInit in aInits]
for aInit, sol in zip(aInits, sols):
    aTraj, gTraj, fTraj = sol.y
    plt.plot(gTraj,aTraj, label=np.around(aInit,2))
plt.scatter([gStar,gInit],[aStar,ac], c="red")
plt.legend()
plt.xlim(0,0.25)
plt.ylim(-0.2,0.05);
```

The next step is to linearize the flows for a and g at the fixed point (a^*, g^*), to find the Jacobian

$$J \equiv \begin{pmatrix} \partial\dot{a}/\partial a & \partial\dot{a}/\partial g \\ \partial\dot{g}/\partial a & \partial\dot{g}/\partial g \end{pmatrix}\Bigg|_{a^*,g^*} = \begin{pmatrix} 5/3 & 169/48 \\ 24/169 & -1 \end{pmatrix} \tag{3}$$

where, e.g., $\dot{a} = \partial a/\partial \ell$.

(g) Calculating all the components of J is straightforward, but a bit tedious. Verify that the lower right element $\partial\dot{g}/\partial g|_{a^*,g^*} = -1$. Then use J to numerically find the eigenvalues and right eigenvectors. Add these to your plot of part (e), and verify that the flows approach the fixed point along the irrelevant eigendirection, and then veer away along the relevant eigendirection.

Your answer here (or in a separate writeup)

```
[ ]: Jacobian = [[5./3., 169./48.],[24./169.,...]]
vals,Tvecs = np.linalg.eig(Jacobian)
# Evil: eigenvectors are columns of matrix, not vecs[0] but vecs^T[0]
vecs = np.transpose(Tvecs)
lambda_t = vals[0]; vec_t = vecs[0];
lambda_g = vals[1]; vec_g = vecs[1];
print("Relevant eig = ", lambda_t, " vec = ", vec_t, " slope = ", vec_t[1]/
  ↪vec_t[0])
print("Irrelevant eig = ", lambda_g, " vec = ", vec_g, " slope = ", vec_g[1]/
  ↪vec_g[0])

# Warning: Eigenvector coordinates are ordered (a,g); must draw them in (g,a)␣
  ↪order
plt.plot([gStar,gStar+vec_t[1]],[aStar,aStar+vec_t[0]], c="k")
plt.plot([gStar,gStar+vec_g[1]],[aStar,aStar+vec_g[0]], c="k")

# Figures from part (d) replotted
for aInit, sol in zip(aInits, sols):
    aTraj, gTraj, fTraj = sol.y
    plt.plot(gTraj,aTraj, label=np.around(aInit,2))
plt.scatter([gStar],[aStar], c="red")
plt.legend()
plt.xlim(0,0.25);
plt.ylim(-0.2,0.05);
```

Calculating predicted critical exponents: parts (h), (i), (j), (k)