



Feb 1. Sat.

Fast Equation of Motion

I exploited the following trick in rewriting the equation of motion to speeds up the code significantly.

Recall that the superpotnetial is

$$W = \sum_{a=1}^N e^{\vec{\alpha}_a \cdot \vec{x}} \quad (1)$$

To find the potential part of the equation of motion, we need to take the derivative and its conjugate:

$$\frac{dW}{d\vec{x}} = \sum_{a=1}^N e^{\vec{\alpha}_a \cdot \vec{x}} \vec{\alpha}_a \quad (2)$$

$$\frac{dW^*}{d\vec{x}^*} = \sum_{a=1}^N e^{\vec{\alpha}_a \cdot \vec{x}^*} \vec{\alpha}_a \quad (3)$$

Then the absolute square is just the products of these two

$$\left| \frac{dW}{d\vec{x}} \right|^2 = \sum_{a=1}^N \sum_{b=1}^N e^{\vec{\alpha}_a \cdot \vec{x} + \vec{\alpha}_b \cdot \vec{x}^*} \vec{\alpha}_a \cdot \vec{\alpha}_b \quad (4)$$

Using the identity (found in our last paper):

$$\vec{\alpha}_a \cdot \vec{\alpha}_b = 2\delta^{ab} - \delta^{a,b+1} - \delta^{a+1,b} \quad (5)$$

where $0 \equiv N$, we can eliminate the sum over b :

$$\left| \frac{dW}{d\vec{x}} \right|^2 = \sum_{a=1}^N \left[2e^{\vec{\alpha}_a \cdot \vec{x} + \vec{\alpha}_a \cdot \vec{x}^*} - e^{\vec{\alpha}_a \cdot \vec{x} + \vec{\alpha}_{a-1} \cdot \vec{x}^*} - e^{\vec{\alpha}_a \cdot \vec{x} + \vec{\alpha}_{a+1} \cdot \vec{x}^*} \right] \quad (6)$$

$$= \sum_{a=1}^N e^{\vec{\alpha}_a \cdot \vec{x}} \left[2e^{\vec{\alpha}_a \cdot \vec{x}^*} - e^{\vec{\alpha}_{a-1} \cdot \vec{x}^*} - e^{\vec{\alpha}_{a+1} \cdot \vec{x}^*} \right] \quad (7)$$

The derivative of this appears in the equation of motion, which is the term:

$$\frac{1}{4} \frac{\partial}{\partial (x^b)^*} \left| \frac{dW}{d\vec{x}} \right|^2 = \frac{1}{4} \sum_{a=1}^N e^{\vec{\alpha}_a \cdot \vec{x}} \left[2e^{\vec{\alpha}_a \cdot \vec{x}^*} \alpha_a^b - e^{\vec{\alpha}_{a-1} \cdot \vec{x}^*} \alpha_{a-1}^b - e^{\vec{\alpha}_{a+1} \cdot \vec{x}^*} \alpha_{a+1}^b \right] \quad (8)$$

where α_a^b is the b -th component of the a -th simple root.

This allows us to reduce a double summation to a single summation, which significantly reduces the speed.

I implemented the code for this and tested it. The values my fast equation speeds out for a given point is exactly the same as the old one, but when $N = 10$, the speed of calculating a specific points increases by 25 times!

Next Step: Optimize with Numpy

Now that I have rewrote the equation of motion to significantly boost the speed, I want to further rewrite this equation so that it takes advantage of numpy and computes things at once, instead of point by point.