Dec 19.

Start of Project

I finally begin my senior thesis project. I technically started last semester, but all of my time was spent on writing and publishing the summer work. My rule for myself: I have to write a diary entry for this project everyday, even if I did nothing. This way, I can stay engaged with the research and can clearly tell if I did not touch research for too long.

Goal of Project

The goal is to compute the confining string tension dependence on N-ality (p) and rank of gauge group (N).

GitHub

I created a public GitHub repository called Confining-String. I will copy some code from the summer project. But the overall structure will be distinct as I want to do things cleaner this time.

Rereading the Paper

As a refresher, my first step is to reread the paper and re-derive the Lagrangian and equation of motion I am trying to solve. I downloaded the paper from JHEP and started rereading section 5.

Source Part of Action

Consider the general case of n quarks, each with charge $\vec{c}_i$, with position $\vec{r}_i = (y_i, z_i)$, where $i = 1, \ldots, n$.

Claim 1. *The source part of the action is given by*

$$S_{source} = -\frac{g^2}{4\pi L} \int dt \int dz \int dy \sum_i (\vec{c}_i)_a \partial_z \delta(z - z_i) \int_{y_i}^{\pm\infty} dy' \delta(y - y') \sigma^a(z, y) \tag{1}$$

Proof. The action of external particle is the spacetime integral of the delta function of its worldline. Since the quarks are stationary, and the energy of a charged particle (just like in classical EM) is the charge times the voltage:

$$S_{source} = -\int dt \sum_i (\vec{c}_i)_a A_0^a(\vec{r}_i)$$

where a labels the $N-1$ fields.

Using the identity (where we assume the potential is zero at infinity),

$$A_0^a(\vec{r_i}) = -\int_{y_i}^{\pm\infty} dy \partial_y A_0^a(z_i, y)$$

and the definition of the field strength tensor,

$$\partial_y A_0 = F_{y0} \implies -\partial_y A_0 = F_{0y}$$

we have

$$A_0^a(\vec{r_i}) = \int_{y_i}^{\pm\infty} dy F_{0y}^a(z_i, y)$$

Next, use the definition of the dual photon

$$\frac{g^2}{4\pi L}\epsilon_{\mu\nu\lambda}\partial^\lambda \sigma^a = F_{\mu\nu}^a$$

This gives

$$F_{0y}^a = \frac{g^2}{4\pi L}\epsilon_{0yz}\partial^z \sigma^a$$

Take the convention that $\epsilon_{0yz} = 1$. Recall that bringing the spatial component down gives a negative sign. We have

$$F_{0y}^a = -\frac{g^2}{4\pi L}\partial_z \sigma^a$$

So the potential can be written as

$$A_0^a(\vec{r_i}) = -\frac{g^2}{4\pi L}\int_{y_i}^{\pm\infty} dy \partial_z \sigma^a$$

Subbing this back into the action,

$$S_{source} = \frac{g^2}{4\pi L}\int dt \sum_i (\vec{c_i})_a \int_{y_i}^{\pm\infty} dy \partial_z \sigma^a(z, y)|_{z=z_i}$$

We can rewrite the fact that $z = z_i$ by introducing an integral and a delta function

$$S_{source} = \frac{g^2}{4\pi L}\int dt \int dz \sum_i (\vec{c_i})_a \int_{y_i}^{\pm\infty} dy \partial_z \sigma^a(z, y)\delta(z - z_i)$$

Change the variable y to y':

$$S_{source} = \frac{g^2}{4\pi L}\int dt \int dz \sum_i (\vec{c_i})_a \int_{y_i}^{\pm\infty} dy' \partial_z \sigma^a(z, y')\delta(z - z_i)$$

Without changing anything, introduce an integral over y as well as a delta function that converts it back to y':

$$S_{source} = \frac{g^2}{4\pi L}\int dt \int dz \int dy \sum_i (\vec{c_i})_a \int_{y_i}^{\pm\infty} dy' \partial_z \sigma^a(z, y')\delta(z - z_i)\delta(y - y')$$

By the property of the delta function, we can drop the prime on the y' in the argument of dual photon

$$S_{source} = \frac{g^2}{4\pi L}\int dt \int dz \int dy \sum_i (\vec{c}_i)_a \int_{y_i}^{\pm\infty} dy' \partial_z \sigma^a(z,y)\delta(z-z_i)\delta(y-y')$$

Integrating by parts for z,

$$S_{source} = -\frac{g^2}{4\pi L}\int dt \int dz \int dy \sum_i (\vec{c}_i)_a \partial_z \delta(z-z_i) \int_{y_i}^{\pm\infty} dy' \delta(y-y')\sigma^a(z,y)$$

We have successfully written the source action in terms of spacetime integral. □

Full Action

Claim 2. *The kinetic and potential part of the action for the complex field is given by*

$$S_{kin,pot} = \frac{g^2}{16\pi^2 L}\int dt \int dz \int dy \left(|\partial_0 x^a|^2 - |\partial_i x^a|^2 - \frac{1}{4}\left|\frac{dW}{dx^a}\right|^2 \right) \tag{2}$$

such that the total action is

$$\begin{aligned} S = \frac{g^2}{16\pi^2 L}\int dt \int dz \int dy \{ &|\partial_0 x^a|^2 - |\partial_i x^a|^2 - \frac{1}{4}\left|\frac{dW}{dx^a}\right|^2 \\ &-4\pi \sum_i (\vec{c}_i)_a \partial_z \delta(z-z_i) \int_{y_i}^{\pm\infty} dy' \delta(y-y') Im(x^a(z,y))\} \end{aligned} \tag{3}$$

Proof. The kintetic part of the action is just the usual absolute square of the complex field, under the usual Lorentz metrics. The overall constant in front of the kinetic and potential action, as well as the potential term, will both be justified later by consistency checks.

When we add the source part and kinetic and potential part of the action, it is clear that the coefficient differs by 4π, which is where the relative factor comes from. Also, in going from the real field to the complex field, if we only want to talk about the dual photon, which is the imaginary part, we have to refer to it in terms of the complex field, $x^a = \phi^a + i\sigma^a$, via

$$Im(x^a) = \frac{x^a - x^{a*}}{2i} = \sigma^a$$

Finally, the last step that remains is to check that the coefficient and the source terms are consistent with known special case. We will do two checks: the action has to agree with the energy of a one dimensional BPS soliton when we ignore one direction and set the charge to zero; if we set $\phi^a = 0$ and set the superpotential $W = 0$, and specialize to two quarks, we should get back the classical field action from Erich's note.

Test 1: If we set the charge to zero and forget the z direction, the action becomes

$$S = \frac{g^2}{16\pi^2 L}\int dt \int dy \left(|\partial_0 x^a|^2 - |\partial_i x^a|^2 - \frac{1}{4}\left|\frac{dW}{dx^a}\right|^2 \right)$$

Recall that action = time integral of (kinetic energy - potential energy). Therefore, we can extract kinetic energy + potential energy by changing the signs in front of terms with no time dependence and discarding the time integral to get

$$E = \frac{g^2}{16\pi^2 L}\int dy \left(|\partial_0 x^a|^2 + |\partial_i x^a|^2 + \frac{1}{4}\left|\frac{dW}{dx^a}\right|^2 \right)$$

Taking the static solution (soliton), the energy is

$$E = \frac{g^2}{16\pi^2 L}\int_{-\infty}^{\infty} dy \left(\left|\frac{dx^a}{dy}\right|^2 + \frac{1}{4}\left|\frac{dW}{dx^a}\right|^2 \right)$$

Up to an overall factor, this is exactly the expression of the energy of a BPS soliton in equation (18.3) in "Mirror Symmetry" book.

Test 2: Set $W = 0$ and $\phi^a = 0$.

$$S = \frac{g^2}{16\pi^2 L}\int dt \int dz \int dy \left((\partial_0\sigma^a)^2 - (\partial_i\sigma^a)^2 - 4\pi \sum_i (\vec{c}_i)_a \partial_z \delta(z - z_i) \int_{y_i}^{\pm\infty} dy' \delta(y - y')\sigma^a(z, y) \right)$$

Specialize to the case of two quarks (quark-antiquark pair) lying on the y-axis and a distance R apart, with opposite charges $\pm\vec{c}$.

$$S = \frac{g^2}{16\pi^2 L}\int dt \left(\int dz \int dy \left[(\partial_0\sigma^a)^2 - (\partial_i\sigma^a)^2 \right] + 4\pi \int dz \int dy \sum_i (\vec{c}_i)_a \delta(z - z_i) \int_{y_i}^{\pm\infty} dy' \delta(y - y')\partial_z\sigma^a(z, y) \right)$$

where we revert the integration by part of z. Since $z_1 = z_2 = 0$ for both quarks are on the y-axis,

$$S = \frac{g^2}{16\pi^2 L}\int dt \left(\int dz \int dy \left[(\partial_0\sigma^a)^2 - (\partial_i\sigma^a)^2 \right] + 4\pi \int dy \sum_i (\vec{c}_i)_a \int_{y_i}^{\pm\infty} dy' \delta(y - y')\partial_z\sigma^a(z, y)|_{z=0} \right)$$

Equating y and y' and taking out the delta function.

$$S = \frac{g^2}{16\pi^2 L}\int dt \left(\int dz \int dy \left[(\partial_0\sigma^a)^2 - (\partial_i\sigma^a)^2 \right] + 4\pi \sum_i (\vec{c}_i)_a \int_{y_i}^{\pm\infty} dy \partial_z\sigma^a(z, y)|_{z=0} \right)$$

Now let the negative quark be at the origin $y_1 = 0$, and the positive quark be at $y_2 = R$.

$$S = \frac{g^2}{16\pi^2 L}\int dt \left(\int dz \int dy \left[(\partial_0\sigma^a)^2 - (\partial_i\sigma^a)^2 \right] - 4\pi(\vec{c})_a \int_0^{\infty} dy \partial_z\sigma^a(z, y)|_{z=0} + 4\pi(\vec{c})_a \int_R^{\infty} dy \partial_z\sigma^a(z, y)|_{z=0} \right)$$

Interchanging lower and upper limit and combining integrals,

$$S = \frac{g^2}{16\pi^2 L}\int dt \left(\int dz \int dy \left[(\partial_0\sigma^a)^2 - (\partial_i\sigma^a)^2 \right] - 4\pi(\vec{c})_a \int_0^{\infty} dy \partial_z\sigma^a(z, y)|_{z=0} - 4\pi(\vec{c})_a \int_{\infty}^{R} dy \partial_z\sigma^a(z, y)|_{z=0} \right)$$

$$S = \frac{g^2}{16\pi^2 L}\int dt \left[\int dz \int dy \left[(\partial_0\sigma^a)^2 - (\partial_i\sigma^a)^2 \right] - 4\pi(\vec{c})_a \int_0^{R} dy \partial_z\sigma^a(z, y)|_{z=0} \right] \tag{4}$$

But this is exactly the same as the formula for action of quark anti-quark on page 18 of Erich's note. □

Dec 20.

Rereading Paper

I continue to read section 5 of the paper. I am completely convinced now that the formula of the action is correct.

Equation of Motion

I now verify the equation of motion, which is the equation I will solve for the rest of the project.

Claim 1. *The static equation of motion is*

$$\nabla^2 x^a = \frac{1}{4}\frac{\partial}{\partial (x^a)^*}\left|\frac{dW}{d\vec{x}}\right|^2 + 2\pi i \sum_j (\vec{c}_j)_a \partial_z \delta(z-z_j)\int_{y_j}^{\pm\infty} dy' \delta(y-y') \tag{1}$$

Proof. The action is

$$S = \frac{g^2}{16\pi^2 L}\int dt \int dz \int dy \left(|\partial_0 x^a|^2 - |\partial_i x^a|^2 - \frac{1}{4}\left|\frac{dW}{dx^a}\right|^2 - 4\pi \sum_j (\vec{c}_j)_a \partial_z \delta(z-z_j)\int_{y_j}^{\pm\infty} dy' \delta(y-y') Im(x^a(z,y)) \right) \tag{2}$$

from which we can extract the Lagrangian density (we can discard the overall coefficient since it cancels out in the EL equation),

$$\mathscr{L} = |\partial_0 x^a|^2 - |\partial_i x^a|^2 - \frac{1}{4}\left|\frac{dW}{dx^a}\right|^2 - 4\pi \sum_j (\vec{c}_j)_a \partial_z \delta(z-z_j)\int_{y_j}^{\pm\infty} dy' \delta(y-y') Im(x^a(z,y))$$

For static equation of motion,

$$\frac{\partial \mathscr{L}}{\partial(\partial_i x^{a*})} = \frac{\partial}{\partial(\partial_i x^{a*})}\left(-\partial_i x^b \partial_i x^{b*}\right) = -\partial_i x^b \delta^b_a = -\partial_i x^a$$

Hence,

$$\partial_i \frac{\partial \mathscr{L}}{\partial(\partial_i x^{a*})} = -\partial_i \partial_i x^a = -\nabla^2 x^a$$

On the other hand,

$$\frac{\partial \mathscr{L}}{\partial x^{a*}} = -\frac{1}{4}\frac{\partial}{\partial (x^a)^*}\left|\frac{dW}{dx^a}\right|^2 - 4\pi \sum_j (\vec{c}_j)_a \partial_z \delta(z-z_j)\int_{y_j}^{\pm\infty} dy' \delta(y-y') \frac{\partial}{\partial (x^a)^*}\frac{x^a - x^{a*}}{2i}$$

$$\frac{\partial \mathscr{L}}{\partial x^{a*}} = -\frac{1}{4}\frac{\partial}{\partial (x^a)^*}\left|\frac{dW}{dx^a}\right|^2 - \frac{2\pi}{i} \sum_j (\vec{c}_j)_a \partial_z \delta(z-z_j)\int_{y_j}^{\pm\infty} dy' \delta(y-y')(-1)$$

$$\frac{\partial \mathscr{L}}{\partial x^{a*}} = -\frac{1}{4}\frac{\partial}{\partial (x^a)^*}\left|\frac{dW}{dx^a}\right|^2 - 2\pi i \sum_j (\vec{c}_j)_a \partial_z \delta(z - z_j) \int_{y_j}^{\pm\infty} dy' \delta(y - y')$$

By The Euler-Lagrange equation,

$$\partial_i \frac{\partial \mathscr{L}}{\partial(\partial_i x^{a*})} = \frac{\partial \mathscr{L}}{\partial x^{a*}}$$

Hence, we have,

$$-\nabla^2 x^a = -\frac{1}{4}\frac{\partial}{\partial (x^a)^*}\left|\frac{dW}{dx^a}\right|^2 - 2\pi i \sum_j (\vec{c}_j)_a \partial_z \delta(z - z_j) \int_{y_j}^{\pm\infty} dy' \delta(y - y')$$

$$\nabla^2 x^a = \frac{1}{4}\frac{\partial}{\partial (x^a)^*}\left|\frac{dW}{d\vec{x}}\right|^2 + 2\pi i \sum_j (\vec{c}_j)_a \partial_z \delta(z - z_j) \int_{y_j}^{\pm\infty} dy' \delta(y - y')$$

□

Derivative of Dirac Delta

Reading the next part of the paper, which was written by Andrew, there is a clever argument for the numerical implementation of derivative of 1 dimensional Dirac Delta, but it is exactly the same equation as what one would get by naive analogy with positive/negative infinite slope.

Let $g(x) = \partial_x \delta(x - x_i)$.

$$g(x_k) = \begin{cases} \frac{1}{h^2}, & k = i-1 \\ -\frac{1}{h^2}, & k = i \end{cases} = \frac{1}{h^2}(\delta_{k,i-1} - \delta_{k,i}) \tag{3}$$

Dec 21.

Equation of Motion for Quark-Antiquark Pair

I now stop reading my old paper. It was a good refresher to remember the basic problem and good warm-up to go through the derivation of action and equation of motion. Now I specialize to the equation of motion for a quark-antiquark pair.

Claim 1. *The equation of motion for a quark-antiquark pair of charge $\pm\vec{c}$ is*

$$\nabla^2 x^a(z,y) = \frac{1}{4}\frac{\partial}{\partial(x^a)^*}\left|\frac{dW}{d\vec{x}}\right|^2 + i2\pi C_a \partial_y \delta(y)\int_{-\frac{R}{2}}^{\frac{R}{2}} dz'\delta(z-z') \tag{1}$$

where the positive quark is at position $(z=-\frac{R}{2}, y=0)$ and negative quark is at position $(z=\frac{R}{2}, y=0)$, as shown in the Figure 1.

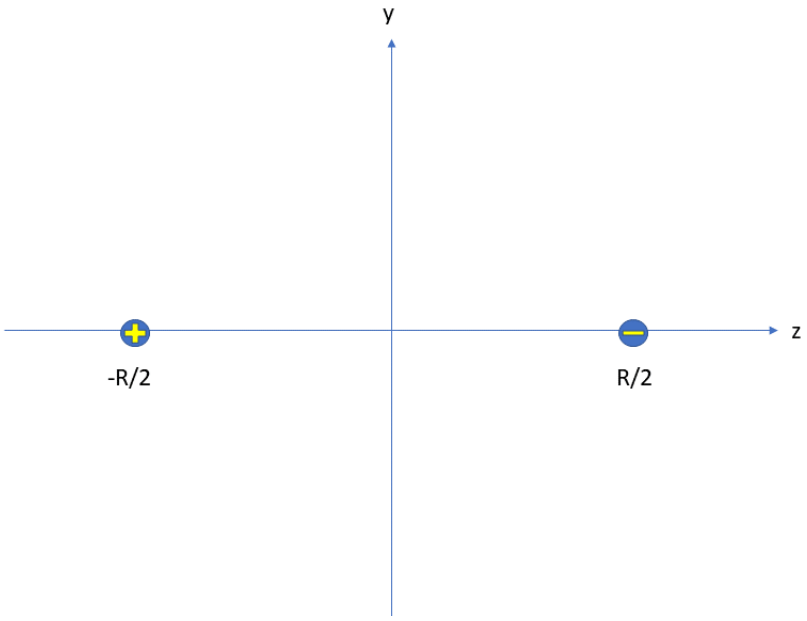

Figure 1: Schematic of Quark-Antiquark Pair

Proof. Taking the equation of motion derived from yesterday, with the exception that we make the change of variable $z \longleftrightarrow y$, to match the usual convention of y being vertical, our generic (for arbitrary number of quarks) equation of motion is

$$\nabla^2 x^a = \frac{1}{4}\frac{\partial}{\partial(x^a)^*}\left|\frac{dW}{d\vec{x}}\right|^2 + i2\pi\sum_j (\vec{c}_j)_a \partial_y \delta(y-y_j)\int_{z_j}^{\pm\infty} dz'\delta(z-z') \tag{2}$$

Since $y_1 = y_2 = 0$, $z_1 = -\frac{R}{2}$, $z_2 = \frac{R}{2}$, we have

$$\nabla^2 x^a = \frac{1}{4}\frac{\partial}{\partial (x^a)^*}\left|\frac{dW}{d\vec{x}}\right|^2 + i2\pi C_a \partial_y \delta(y)\int_{-\frac{R}{2}}^{\infty} dz' \delta(z-z') - i2\pi C_a \partial_y \delta(y)\int_{\frac{R}{2}}^{\infty} dz' \delta(z-z')$$

Swapping upper and lower integration limit,

$$\nabla^2 x^a = \frac{1}{4}\frac{\partial}{\partial (x^a)^*}\left|\frac{dW}{d\vec{x}}\right|^2 + i2\pi C_a \partial_y \delta(y)\int_{-\frac{R}{2}}^{\infty} dz' \delta(z-z') + i2\pi C_a \partial_y \delta(y)\int_{\infty}^{\frac{R}{2}} dz' \delta(z-z')$$

Combining the two integral,

$$\nabla^2 x^a = \frac{1}{4}\frac{\partial}{\partial (x^a)^*}\left|\frac{dW}{d\vec{x}}\right|^2 + i2\pi C_a \partial_y \delta(y)\int_{-\frac{R}{2}}^{\frac{R}{2}} dz' \delta(z-z')$$

□

This will be the equation I will be solving for the next few months. Enough derivation for now! Tomorrow, my goal is to put together all of these formulas I have derived in the past few days, formulate a formal, succinct problem statement, and start coding.

Dec 22.

Energy

The energy of a field configuration is given by

$$E = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} dzdy \left(\left|\frac{\partial x^a}{\partial z}\right|^2 + \left|\frac{\partial x^a}{\partial y}\right|^2 + \frac{1}{4}\left|\frac{\partial W}{\partial x^a}\right|^2 \right)$$

This is a generalization of the one dimensional BPS energy we derived earlier. It also makes sense since in classical EM, the energy is $|E|^2$, and the $E = \nabla\Phi$. Since the dual photon has the same unit as the electric potential, the energy should also contains the gradient square of x^a.

Problem Statement

The action of a static quark-antiquark pair of charge $\pm\vec{C}$, separated by a distance R, with the positive quark at position $(z = -\frac{R}{2}, y = 0)$ and negative quark at position $(z = \frac{R}{2}, y = 0)$, as shown in Figure 1, is given by

$$S = \frac{g^2}{16\pi^2 L}\int dt \int dz \int dy \left(|\partial_0 x^a|^2 - |\partial_i x^a|^2 - \frac{1}{4}\left|\frac{dW}{dx^a}\right|^2 - 4\pi C_a \partial_y \delta(y) \int_{-\frac{R}{2}}^{\frac{R}{2}} dz' \delta(z - z') Im(x^a(z, y)) \right) \tag{1}$$

where $W(\vec{x})$ is the holomorhpic superpotential

$$W(\vec{x}) = \sum_{a=1}^{N} e^{\vec{\alpha}_a \cdot \vec{x}} \tag{2}$$

and $\vec{\alpha}_1, \ldots, \vec{\alpha}_{N-1}$ are the simple roots, and $\vec{\alpha}_N = -\sum_{a=1}^{N-1} \vec{\alpha}_a$.

The static equation of motion is

$$\nabla^2 x^a(z, y) = \frac{1}{4}\frac{\partial}{\partial (x^a)^*}\left|\frac{dW}{d\vec{x}}\right|^2 + i2\pi C_a \partial_y \delta(y) \int_{-\frac{R}{2}}^{\frac{R}{2}} dz' \delta(z - z') \tag{3}$$

Upon solving for the field, the energy of the configuration is given by

$$E = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} dzdy \left(\left|\frac{\partial x^a}{\partial z}\right|^2 + \left|\frac{\partial x^a}{\partial y}\right|^2 + \frac{1}{4}\left|\frac{dW}{d\vec{x}}\right|^2 \right) \tag{4}$$

It is also known that for large R, the energy depends linearly on the distance, and that the coefficient of proportionality, f, also called string tension, depends only on the number of fundamental color indices (N-ality p), and the number of color, i.e. the rank of gauge group (N):

$$E = f(p, N)R \tag{5}$$

The goal is to find the unit-less ratio

$$F(p, N) = \frac{f(p, N)}{f(1, N)} \tag{6}$$

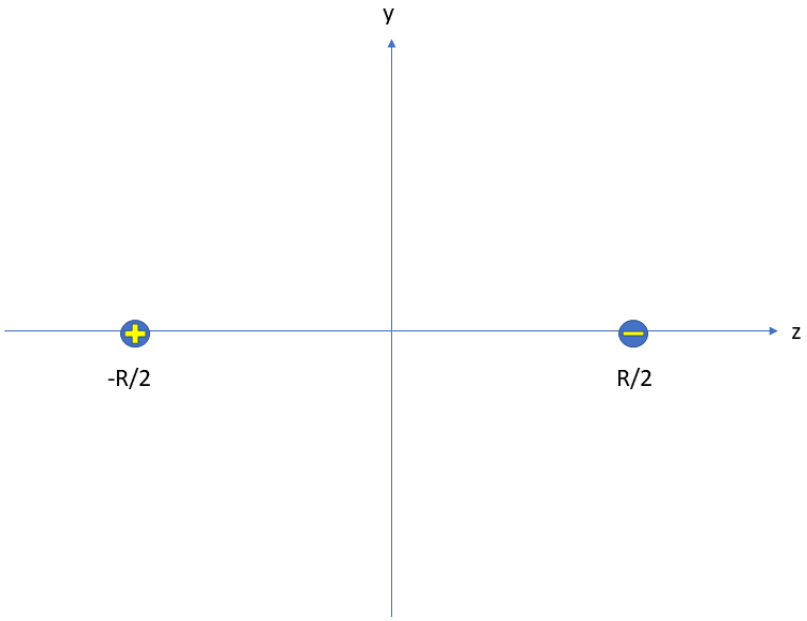

Figure 1: Schematic of Quark-Antiquark Pair

Grid

In starting to write code, the first step is to review the code from the summer project and move them to the new repository. The reason to carefully review the code is that the old version has become way too cluttered, with many different versions of similar class written in the same file. For the new project, I am only simulating confining pair, so I can remove all the unnecessary code with Baryon, deconfinement, etc. Also, I have this constant worry that there might be some small mistake, so I want to check over everything again. In addition, I want to do certain things differently this time around. For example, I want to implement the half grid calculation to speed things up.

The first thing is to review the Grid class and its related classes stored in the Grid module. I read over the general Grid class (except for the plot empty grid, which is exclusively used for testing). Everything is very basic common sense and looks good. There is no need to change anything.

Secondly, I reviewed the children class, Grid_Dipole. I added the important comment that this only works if the input parameters has the 0 y-axis right at the middle of the grid. In fact, this is so important that I am adding a function that enforces this.

I made some small modifications to the Standard_Dipole class, as well as some tests, but overall, this module looks good. I copied it to the new repository. Eventually, I want to add a half-grid class, but I will do that later and it seems the current code is perfectly compatible with it.

Dec 23.

Did not do research today.

Dec 24.

Did not do research today.

Dec 25.

Did not do research today.

Dec 26.

Math

I now review the Math module, which contains the important classes: SU class, which computes and stores important constants associated with the SU(N) group; and the Superpotential class, which represents the superpotential function, W, and computes its derivatives. I decided that this module is way too complex to rewrite. However, my fear that there is some mistake uncaught before (very small chance) could very well happen here, just due to its complexity. So I am now taking on the hard task of reviewing all the calculation here.

SU(N) Class

Reviewed the definition of weights of the fundamental representation $(\nu^A)_a$.

Reviewed the definition of the simple roots, $\vec{\alpha}_a$.

Reviewed the definition of the fundamental weights, $\vec{w}_a$.

Reviewed the definition of the Weyl's vector, $\vec{\rho} = \vec{w}_1 + \cdots + \vec{w}_{N-1}$.

As a side note, I also added these definitions to my Anki deck, so that I can memorize them.

Next, I migrated the old test_Math module to the new repository, and I ran the old tests on SU class. Since I am so confident that this part is correct from before, and I just read through the source code, I trust that the tests are correct without reading through them in detail.

Superpotential Class

Reviewed the "__call__" function, which is the most important part that everyting else depends on.

The key to solve the equation of motion is the potential term (showing the b-th component, i.e. potential term of $\nabla^2 x^b$):

$$\frac{1}{4}\frac{\partial}{\partial x^{b*}}\left|\frac{dW}{d\vec{x}}\right|^2 = \frac{1}{4}\frac{\partial W}{\partial x^a}\frac{\partial^2 W^*}{\partial x^{b*}\partial x^{a*}} \tag{1}$$

I reviewed the very complicated code whose complexity primarily lies in dealing with the shape of the array when evaluating on a 3D array (vector field on a grid). But the core mathematical formulae used are

$$\frac{dW}{d\vec{x}} = \sum_{i=1}^{N} e^{\vec{\alpha}_i\cdot\vec{x}}\vec{\alpha}_i \tag{2}$$

and

$$\frac{\partial^2 W}{\partial x^b \partial x^a} = \sum_{i=1}^{N} e^{\vec{\alpha}_i\cdot\vec{x}}(\vec{\alpha}_i)^a(\vec{\alpha}_i)^b \tag{3}$$

It took hours to analyze all the code. Now, that I believe it is correct, I will just run the test (without analyzing them because I trust the old tests written and what I just read enough. And the result is that the code passed all tests. So I conclude that the Math module is good to go for the new repository.

Final note: I am making this new repository highly organized. Inside the code folder, test belong to a subfolder, while all the "tool" modules, such as Grid and Math, that are going to be called by the main code (to be added), will be inside a "Tool" subfolder. The final main code will then be very simple and I won't need to worry at all about all the dirty math and array shape behind the scene.

Dec 27.

Today's goal is to write the relaxation and laplacian code. The two used to be separate classes in the old code. But I think it becomes too messy and logically incoherent when we try to use the half grid method as well. The laplacian is fundamentally tied to the method we choose. A half grid has a corresponding half laplacian, and the relaxation for half grid is also different. So these two things clearly belong to the same unit, and different units correspond to different methods, such as full grid vs half grid. My goal today is to complete this full unit by combining and modifying old code and solve the first confining string equation of motion of this project.

Relaxation Class

I first migrated the old relaxation code to the new repo. I took out all the options related to Neumann boundary; that will be useful when we eventually switch to half grid calculation, but that's for another code. Since we are doing relaxation with the boundary being a uniform value, I made the default initial grid to be just uniformly the external vacuum value.

I then reviewed the relaxation algorithm. Everything is straight forward and looks right.

I modified the relaxation code such that it includes the full grid equation of motion as its default. I decided that the half grid method will be a child class that modifies its equation of motion and update method. Again, that's for another day.

Charge Class

I created a Charge class in the "tools" folder. This class handles the name and the vector of a charge, which is a linear combination of fundamental weights. I also tested it and it is doing very well. This is going to make the human interaction much simpler, as I can just type in words like "w1 + w2", and the computer will know what charge to compute.

Solver_Full_Grid and First Result

I wrote the first draft the main function, which solves the equation of motion by calling on other classes. I also tried to run it and it seems promising. At the moment, I only have this simple function. A lot more is lacking:

1. Energy function: I need a function that calculates the energy as well as generate the energy density function plot
2. σ-space class: I want to generalize the Charge class to a σ-space class that allows me to deal with both charge and the vacuum.
3. Proper storage system: I need to add in capabilities to store results and images

4. Test solver: I need to test whether what I just wrote is correct. I ran it on SU(2) and it seems to give promising result, with field that looks like the confining double string! However, I need to check that the middle is indeed a separate vacuum.

5. Test solver by differentiating: I can take the Laplacian of the solution and see if it reproduces the Poisson equation. This should be doable, but tricky with all the monodromies.

6. Magnetless Confining String: This can be its own project. I have thought about this before but very pleased that it turns out to be true. The magnetless solitons I discovered in the last project is only for one dimensional wall. I suspected the corresponding two dimensional confining strings will also be magnetless. However, this is not so obvious close to the charges. But it seems to be true in the SU(2) case that I just tested. I suspect this is true in general, that magnetless solitons correspond to magnetless confining strings, and maybe, just maybe, we can also find analytic solutions for these magnetless confining strings.

I intend to accomplish the first 4 points tomorrow. Today is a very productive day!

Dec 28.

Start to address the problems from yesterday.

Magnetless Solution

I first noticed that for SU(2), the confining string solution has identically zero ϕ. This is a stronger result than the fact that the corresponding one dimensional soliton is magnetless. I might think it is obvious that a one dimensional magnetless soliton shall have a corresponding magnetless 2D solutions, but I don't think it is that obvious. When the charge is introduced, it is even less obvious.

I had the idea that maybe there could be a analytic solution for a magnetless 2D solution as well. After working it out for SU(2), I realized it is basically impossible. For the record, I worked out that for SU(2), the ϕ component of the equation of motion is consistent if we set $\phi = 0$, and the imaginary component is

$$\nabla^2\sigma = \sqrt{2}\sin(2\sqrt{2}\sigma) + 2\pi\frac{1}{\sqrt{2}}\partial_y\delta(y)\int_{-\frac{R}{2}}^{\frac{R}{2}} dz'\delta(z-z')$$

It is nice that I verified it is consistent to set the real component to zero, but it is not at all clear how one could write down an analytic solution for this. For a start, if there is not Superpotential, then we only have the second term, and we know that the analytic solution is the difference of two (weirdly-defined) arctan. And if we restrict it to one dimension, the result must somehow reproduce my magenetless analytic solution.

I later found out that Andrew basically did the same thing in our paper. So there is nothing new here and I don't think this is worth exploring more, at least for now.

Simga Space Critical Points Class

I generalized the Charge class from before and modified it such that it can now handle arbitrary linear combinations of fundamental weights and minimum vectors. These constitute all the critical points (or vaccua) of the superpotential in the sigma lattice. This is a powerful code that not only will make human interaction much simpler (can just type "w1" or "x1 +w1" to specify charge and vacua), it will also make the eventual automation of the mass computation much easier (can easily loop through the charge and the vacua, and save the result in strings that can be easily understood).

Solution Viewer

I think I finally came up with a way to store the solution properly. I knew that the old way of doing it is way too idealistic and impractical. Before, I created a field_solution class that stores all the results and has lots of properties and functions that allow me to play with the result. This is all very nice and neat except that I constantly find new things to add and change to the field solution class. But once this is

done, I cannot apply the same code to the old results I generated. This caused a lot of troubles for me back in the summer because it takes a lot of computational time to get these relaxation solutions. So I knew I needed to come up with a better solution storage and viewing system this time around.

Then as I am looking through my old field solution code, it occurs to me that all of these code is valuable and useful still, and I only need to make one twist:

Decouple the storage from the viewing!

I just need to come up with a storage system that permanently stores the bare minimum information of each solution. These are the basic things we never need to change: the actual solution field, the grid parameters, N, N-ality, the vacuum, the charge, a list of error, the tolerance used, maximum loop. That is all! I don't think I will ever need to modify this list in any significant way. So I just need to save a file that contains all of these information, forever.

On the other hand, I will have a class called solution viewer. This object allows me to plot the result, its energy density, computes its energy, checks its validity, and so forth. This is decoupled from the storage. This way, I can add new features all I want, without ever having to modify the actual stored results.

The default solver will take a bunch of parameters, check if it has been solved in the storage, and compute only if it has never been computed before. If it has, it checks whether various property has been computed and plotted, such as energy. It will then display the old plots or regenerate using the viewer object.

The is a plan. I have not actually implement it yet. I will start tomorrow, as today's work was interrupted by an end of year party.

Dec 29.

No research done today (spent the day moving back to Toronto).

Dec 30.

I spent the first day back in Toronto working at office, McDonald, and cafe. I got some good work done. First, I designed and implemented a storage system.

Storage System

When the solver function is called, a solution folder path ("title") is generated, which is just a string that describes the path to a folder whose name records all the parameters of the specific solution. All of these solutions are in a "Results" parent folder.

Once this is done, the code will check whether such folder path already exists. If it does, this means this solution has already been generated before. Then the code creates a black-box object that allows the user to "view" the solution, by which I mean view all property as well as compute and display energy density etc. This black-box object is called a "Solution Viewer", and it just takes the file address of the solution folder. At the moment, I leave it as a black-box and will implement it later.

If the folder path does not exists, then the solver code will go through the normal relaxation protocol, generate a solution, and store all the core data in a giant dictionary. This dictionary includes the field solution, the parameters, the error array, the number of loops, etc. It will not include any properties that need to be computed after the solution is found. I call this the "core dictionary" and it will not be altered once a computation is done.

Then the solver code will create an instance of the above mentioned "Solution Viewer", and this time, it will take the solution, computes all of its derived properties, such as plots, energy density, energy, laplacian check, and stores all the results and images in the same folder as the dictionary.

Solution Viewer - Beginning

Then I started writing the solution viewer. This is going to be a slower but not hard process. The reason is that it is almost completely adapted from the old "solution field" code. But since this code is so long, it will take a while for me to digest and tweek it.

Interruption by Axion Side Project (All side project titles will be red)

At 2 pm, just as I was about to get into solution viewer code, I was interrupted by an email from the boss himself. He is now working on a new paper about axion, and wants me to check something about domain wall in this new but similar theory using my code from the summer. This is a small but tedious task. I have to take the old code, understand it, rewrite something to make the new domain wall solution. Erich believes it will cost one afternoon, but I think it will be a full day of work.

I got quite interested and started right away. I spent hours reading up the detailed precursor note he sent me, which explains the context of the physics and some equations leading to Lagrangian and energy of

the system. I followed and filled in detail derivation for most of it (which I will not reproduce here, at least not today).

Then I started to code up this problem. I spent about 5 hours of the evening on this in total today. I got to the point where I coded up the equation of motion, but realized there is a discrepancy between my new way of storing vector fields in the confining string project (points are columns, vector components are rows) and the way the old "domain wall" code (points are row, components are columns). So while writing the equation of motion code, I did it wrong. I have to change it all tomorrow. This is so tedious!

Dec 31.

Finished First Draft of Axion Code

After some debugging, I finished all the problems with array shapes, and adapted the old plotting code (which compares numerical and theoretical derivative) to the present problem (which compares numerical and theoretical second derivative). Surprisingly, the numerical second derivative can be cheaply computed using "derivative_sample" on itself twice (not a very accurate algorithm); it works really well.

All of this were stored in an independent "DYM" sub-folder. I got to the point where I just needed to run the code for a few different cases. Unfortunately, I kept getting extraneous solutions that solve the equation of motion but are clearly not domain wall solutions. I am very sure that Erich should be right that these domain walls must exist, and it is my fault that the equation of motion that I derived is wrong. Nevertheless, I sent an email to Erich, explaining to him the extraneous results I got, and my suspicion of a wrong equation of motion. I sent him the equation I used.

Erich replied very quickly. He is glad that I am responding to this project so fast. And indeed, he claims that my equation of motion is missing a negative sign. However, I do not understand his explanation of my mistake. He speaks in a language of varying the energy, while I believe I should be varying the Lagrangian. I have to go through the math again.

But Erich reminded me that I don't have to reply today. It's new year eve afterall! So I stopped and went out to celebrate for the evening.

Jan 1.

Understood Why Vary Energy

First day of the year, I spent most of my time applying to the last university (Columbia) and enjoying the holiday. But I took some free time to read up on the Lagrangian and Hamiltonian section of the beginning of the Schroeder book. I realized that the two differ by a negative sign if not for a time gradient term. For our soliton case, since we set time dependence to zero, Lagrangian and Hamiltonian only differ by an overall negative sign. So varying energy is the same as varying the Lagrangian. I finally understood Erich's comment from yesterday's email.

But superficially, now that I understand the right way, I did some mental calculation and disagree with the sign mistake that Erich told me about my equation of motion. I needed to work it out properly, but that is not for a holiday like today (when I was already very productive with applying to the last school). Tomorrow shall be the day I finish this side work and resume my own thesis project.

Thu, Jan 2.

Finished Axion Code

Today was a very productive day for the Axion side-project. I worked most of the day at McDonald and Chatime. I found those environments quite productive and made a mental note to work at these places more.

I went back and re-derive the equation of motion. The original one I had was wrong basically because I forgot the Lorentzian metrics give the spatial derivative a negative sign in the Lagrangian. Once I take that into account, everything else I did was correct.

One trouble I had was figuring out how Erich re-scaled the units and derived the Hamiltonian (with very simple unit) from the Lagrangian (with complicated pre-factors and units). But since I now understand why it is equivalent to vary the energy when there is no time dependence, I just apply the Euler-Lagrange equation on the expression of the energy given by Erich; the resulting equation of motion must be in the simplified unit where the mass of dual-photon is set to unity. In short, I understood everything except for the unit, but it didn't prevent me from getting the right equation of motion, which is all I needed.

Then I made the appropriate sign change in my code and everything just went smoothly. I ran the soliton solution for all different values of the scale α. Nothing that when α gets large, the axion's profile becomes so large that it does not approach a flat line at the boundary of the grid, signalling the grid is too small. I made sure the grid is large enough for the lowest α, but didn't bother to increase them further for the larger one since I now know that the solution is essentially the same, and increasing the grid significantly seems to be too time-expensive.

In addition, I noticed some interesting "bumps" in the dual photons solutions. I summarized all of these results and send all the images to Erich in the evening.

Final Run

After receiving that email, Erich requested that I run the code on a larger grid to fix that grid size problem. Both of us expected this to be costly, even at size ± 50. But then I found that ± 50 is not enough even and ran it on ± 100. Surprsingly even ± 100 is not that costly, only about 10 minutes. Erich is satisfied and this basically concluded my part of the axion project.

Fri, Jan 3.

Missed Email

This was the weekend when I went back home one more time before school starts. I basically tuned out of emails and relaxed for the most part. Apparently, there were further things Erich wanted me to do, but I didn't see the email until Monday.

There was an email requesting me to numerically solve for other DW walls for the axion project. Next, Andrew did his version of the axion simulation and sent it to us. However, his result disagrees with mine in that his axion field did not spread out more as its scale increases. From the energy function, it is obvious that the profile over which there is non-zero gradient should be larger as the axion scale gets large. So I have no doubt that I am correct and Andrew will agree with me after further inspection.

Started Solution Viewer

Going back to my confining string project now, I started copying the old code called "field solution" class and started adapted it to my solution viewer class. In principle, everything is the same except that the old code stores the field solution and the new code read the solution from the core dictionary. But it was quite a lot of work to go through the code and adapted everything. I believe it will be worth the effort because this mode of storage is much more resilient to future changes.

Today, I only got to the point where I can plot field and error.

Sat, Jan 4.

Finished Solution Viewer

I added the capability to compute and plot energy for the solution viewer. This is non trivial because I need to adjust the monodromy, whose location is different than before (for deconfinement, it is 2 vertical lines; for confining string, it is one vertical line). I then merged the branch of solution viewer. I also ran some solution. The solution looks right qualitatively for SU(2), but I haven't check it in detail.

2 Readings

Another email I missed from Erich that I read only after coming back. The first one is a paper by Misha on the current status of high energy physics. It was a good read: the paper points out the problem with the lack of evidence for theorists to work with, but that as any field matures, this is what is expected and progress will be slow and the theorists should get used to it; on the other hand, the author hopes that the new generation will focus more on the unsolved, observable problems such as dark matter.

The second paper is a draft of the axion paper with my plot. I haven't gotten around to reading it yet.

Sun, Jan 5.

Official End to Axion Project

I got an email from Andrew that he redid his calculation and now his result agrees with me, which is expected by me. Erich also told us that there is no need for the new axion domain wall simulation now. I am officially done with it.

An "Eureka" Idea: Direct Test of Edge Effect

I didn't actually get any concrete work done today because I spent half the day travelling back to Toronto. However, something quite remarkable happened today. In the morning, while thinking about the overall direction of the research, I had a "Eureka" moment in which I realized a new way to solve the confining string problem!

In summary, we already have all the technology to solve the confining string (at least Andrew does; I do in principle, but my something about my SU(3) solution looks strange. It is probably some bug or some convergence issue; neverthesless, I have everything I need to solve in principle). The last few days, I was just reviewing and rewriting the old code to make sure they are right and optimizing them for mass computation.

However, the main obstacle is the edge effect. During the summer, we got stuck because we decided to study the grid size dependence (edge effect) to the simulation. By running the simulation on a grid, we assumed that the edge of the grid is far enough from the center (the location of interest) that it can be approximated to infinity. But when we consider the classical electrodynamics, we found that by solving the grounded box - image charge problem, the edge effect is very strong! In fact, the expected behaviour of logarithmic dependence of the energy gets destroyed by the edge effect at about 20% or 30%. This is a huge computational cost.

It occured to me, while waking up and still in bed, that we didn't need to do any of this. There is a much better method to study the grid effect, both in terms of time cost and accuracy. The idea is super simple. We run the same simulation twice: once on a large grid, once a small grid. Presumable, the large grid is a few times larger than the small grid. The ratio of the size is such that the small grid total area covers a region in the center of the large grid such that it is so far away from the edge that we are confident there is no edge effect in that region. Then we run a range of quark distance R in both grid and compare their slope, to see how much does the edge effect change anything.

This is much better in terms of accuracy because there is zero doubt in whether the two simulation are equivalent: they are literally the same simulation, just different grid. The previous method has to make the assumption that the grid effect in a theory without superpotential somehow translates to a theory with one.

Furthermore, this method is quite time efficient because we only have to run the large grid once, for each

N at least. Presumably, the grid effect does not depend on the charge, and I suspect not even N, which is the number of grid component. This idea is a huge game changer to the project! Baring some compeltely unforseen problems, I believe the project will finish successfully just by running large scale simulation.

Mon, Jan 6.

Today is the first day of school.

Good SU(2) Tension Result but Bad SU(3) Result

I ran a few things last night. First I solved SU(2) on a 30 by 30 grid, and solved it for R ranging from 1 to 29. Then I did the exact same thing but only on a 10 by 10 grid. The result is very beautiful and the edge effect is surprisingly small, which seems to be too good to be true.

Basically, for both the small and the large grid run (for sU(2)), I got a **strictly** linear relation. This is just way too nice! The slope is very easy to extract in this case. Further more, all the energy almost completely agrees for the small and large grid! This appears to support the claim that there is no edge effect! This is also too good to be true. See the plot below.

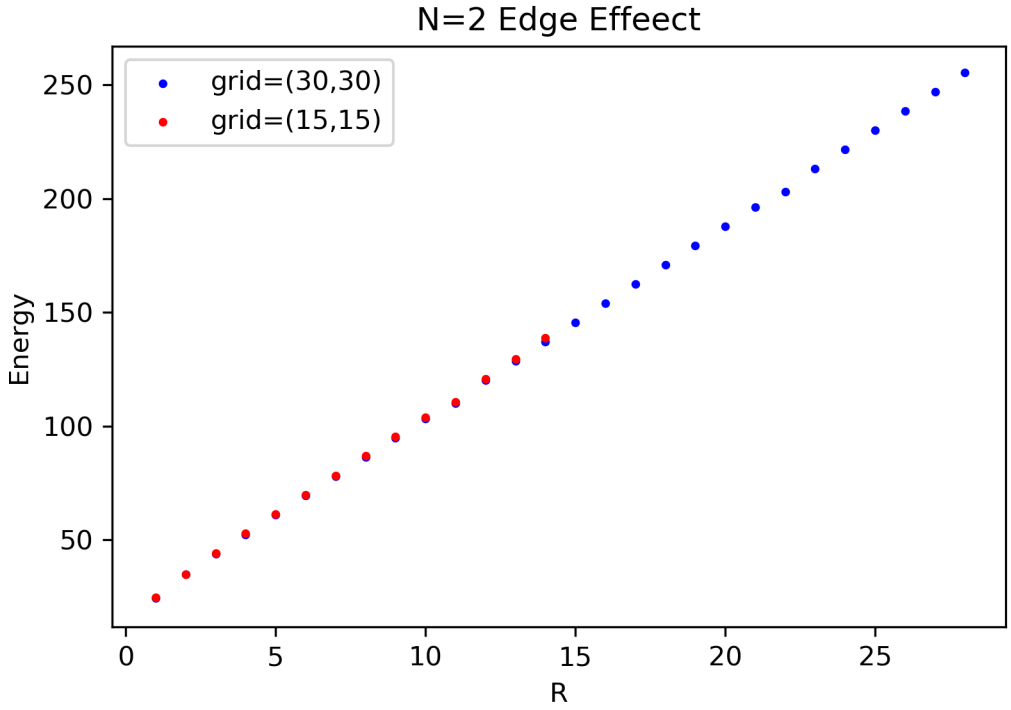

Figure 1: SU(2) Edge Effect Study

However, when I tried to solve the simulation for SU(3), the result just doesn't look right. The middle does not reproduce a vacuum and so the energy does not go to zero in the center. It could be that the convergence is just not finished, or that there is some problem with the equation of motion. I chatted with Andrew and I decided that I will run it for longer tonight and if it still doesn't work, I will do a check of the equation of motion with him tomorrow.

I highly doubt that my code is wrong, because my earlier work used the same basic code and I had the

same deconfinement and baryon result as Andrew. But I am confused by this problem and so I do not trust the too-good-to-be-true SU(2) energy result at the moment (although the SU(2) solution seems right). I wish this can be fixed soon.

Reflection: Christmas List

On December 27, I listed a few important things to do. It was an important list and it occured around Chirstmas, so I will call it the Christmas List. At this point, I think every point there was addressed:

1. All the energy and energy density were taken care of in the Solution Viewer class.
2. The Sigma Critical points class is very sucessful in converting phrase like "x1" and "w1" to numpy array.
3. A proper storage system is implemented, with the key results stored in a core dictionary, and the information and plots are extracted using the solution viewer object whenever desired.
4. I examined the SU(2) solution and they all seem correct. The double string picture emerges, and the fields look similar to Andrew's earlier results.
5. I implemented an Laplacian check, in which after every field configuration is solved, I compare its numerical Laplacian and theoretical Laplacian (by plugging into equation of motion) and compare them side by side. This works very well.
6. After an hour of work, it is clear that it is basically impossible to write down a 2D solution of magnetless solution.

I declare that the Christmas List is resolved.

Jan 7.

Check Validity of Equation of Motion with Andrew

After being very confused about why my SU(3) result doesn't look right, I checked with Andrew today to make sure my equation is correct. We input a few numbers into our potential term of the equation of motion and check result match. Also, we took a special case of SU(2), whose simplified analytic form is given in our paper, and compare. For a good hour, mine was correct, but Andrew's numbers were off. He was so worried! But later, it turned out he was just not calling his simple roots vector correctly. After using his own code properly, we decided that both of our equation of motions are completely correct. I want to stress, to my future self reading this, that I shall not doubt my equation's correctness again.

I did learn something new: the equation of motion I am currently using is not optimal at all. I am currently computing a sum for every point. This can be simplified using a delta function relation between the simple roots. I will eventually optimize the code and fix this.

BPS Initial Grid

After making sure the equation of motion I used is correct, I decided that it must the fault must lie on the initial grid. It turns out that I have been using the boundary value as a constant initial configuration, while Andrew did something much smarter. He guessed the vacua that ought to be inside, solve the corresponding BPS equation, and make the initial grid to be (repeated) BPS solutions. This will guide the computer to the right solution.

I hope that my SU(3) solution will work after adapting to the BPS initial grid. I spent the entire evening writing this code. I started a "BPS Initial Grid" new branch, copied the old "Domain Wall" repository code for solving BPS, and started modifying it into a small package that be called upon in the confining string repository.

New-School-Year List

Just like the Christmas list, I will create a new to-do list, named after beginning of the new school year. The goal is to finish this by end of January.

1. Finish BPS initial grid.
2. Successfully solve SU(3) confining string.
3. Do a edge effect study of SU(3), hopefully confirming the SU(2) result that there is basically no edge effect.
4. Optimize the code by using a shortcut equation of motion.

5. Optimize the code by running it on half grid.

Jan 8.

No research was done today.

Jan 9.

No research was done today. Got offered an interview by UPenn and spent most of the day preparing for it.

Jan 10.

No research was done today. Did the UPenn interview and pick up mom from airport.

Jan 11.

No research was done today. Spent a little time hand solving the SU(2) analytic BPS solution again, in preparation of the talk. But not much real progress.

Jan 12.

While preparing for the talk next week, I learned two things related to this research, the first thing should probably go into the thesis.

Precise Relation of $\vec{\sigma}$ and $\vec{\phi}$ to 4D

Recall from our paper the definition

$$\frac{g^2}{4\pi L}\partial_\mu \phi^a = F^a_{\mu 3} \tag{1}$$

$$\frac{g^2}{4\pi L}\epsilon_{\mu\nu\lambda}\partial^\lambda \sigma^a = F^a_{\mu\nu} \tag{2}$$

where $\mu, \nu = 0, 1, 2$, and x^3 is the direction of $\mathbb{S}^1$.

Suppressing the constant factor, if $\mu = 0$ and $\nu = i$, then in the second equation, we have

$$E^a_i = F^a_{0i} = \epsilon_{0i\lambda}\partial^\lambda \sigma^a$$

which implies

$$E^a_1 = F^a_{01} = \epsilon_{012}\partial^2\sigma^a = \partial^2\sigma^a \tag{3}$$

$$E^a_2 = F^a_{02} = \epsilon_{021}\partial^1\sigma^a = -\partial^1\sigma^a \tag{4}$$

So the spatial derivative of the dual photon is the electric field rotated by 90 degrees.

If $\mu, \nu = 1, 2$, then

$$B^a_3 = F^a_{12} = \epsilon_{120}\partial^0\sigma^a = \partial^0\sigma^a \tag{5}$$

So the time derivative of the dual photon is the magnetic field in the $\mathbb{S}^1$ direction.

In the first equation, if $\mu = 0$,

$$E^a_3 = F^a_{03} = \partial_0\phi^a \tag{6}$$

which means the time derivative of $\vec{\phi}$ is the electric field in the $\mathbb{S}^1$ direction.

If $\mu = 1, 2$, then

$$B^a_2 = -F^a_{13} = -\partial_1\phi^a \tag{7}$$

$$B^a_1 = F^a_{23} = \partial_2\phi^a \tag{8}$$

So the spatial derivative of $\vec{\phi}$ is just the magnetic field rotated by 90 degrees.

In summary, the spatial derivative of $\vec{\sigma}$ is the rotated electric field; the spatial derivative of $\vec{\phi}$ is the rotated magnetic field; the time derivative of $\vec{\sigma}$ is the perpendicular magnetic field; and the time derivative of $\vec{\phi}$ is the perpendicular electric field. This is depicted in figure 1.

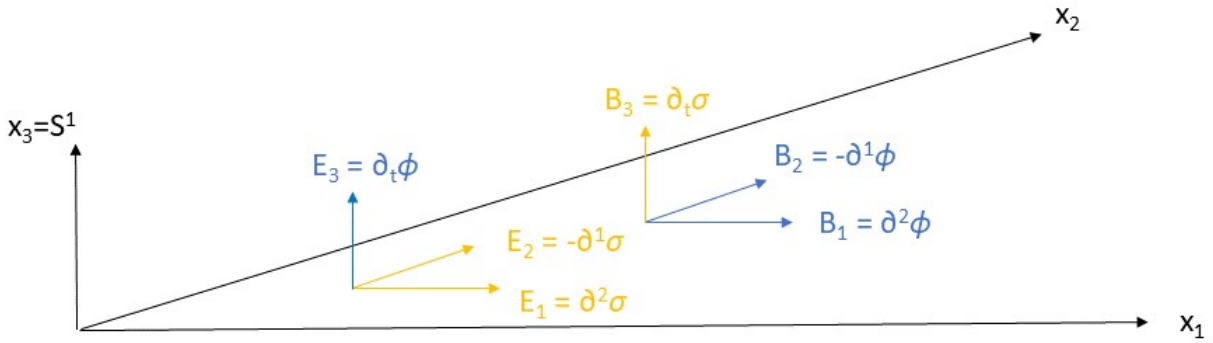

Figure 1: Relation of $\vec{\sigma}$ and $\vec{\phi}$ to 4D

Special Case of My Magnetless Soliton Formula

Recall my magnetless finding from the summer: if N is even, there are always at least 2 magnetless soltions. Their boundaries are given by

$$i2\pi \left(\underbrace{\vec{w}_1 + \vec{w}_3 + \cdots + \vec{w}_{N-1}}_{\text{all odd terms}} \right) \to \vec{x}_{N/2} \tag{9}$$

$$i2\pi \left(\underbrace{\vec{w}_2 + \vec{w}_4 + \cdots + \vec{w}_{N-2}}_{\text{all even terms}} \right) \to \vec{x}_{N/2} \tag{10}$$

I proved these in general. Now let's look at the case for $N = 2$. In $SU(2)$, there are $\vec{w}_1$ only, and there is only $k = 1$ wall. So a soliton can either go from 0 to $\vec{x}_1$, or from $\vec{w}_1$ to $\vec{x}_1$. The second one is described by the first formula as all odd terms. But what about the soliton starting at the origin? It seems not to be in any of the formula for magnetless boundaries!

Is this a mistake? No! The answer is that $\vec{w}_N$ should be identified as 0. In that case, the only time the all even terms boundaries will be 0 is for $SU(2)$. Then this case is taken care of. In the paper, we also gave a consistency check of this formula by showing that these two fluxes form a size 2 orbit under the center symmetry. But recall that in the formalism we developed for the center symmetry, we must identify $\vec{w}_N$ with 0, exactly what we have here. Hence,this formula describes $SU(2)$ as well. It is remarkable that I proved this formula in all generality, without noticing the pit fall with $SU(2)$, but it turns out $\vec{w}_N = 0$ is built into the math.

Jan 13.

No research was done today. Prepared a lot for tomorrow's UCSB interview.

Jan 14.

No research was done today. The interview with UCSB went really well.

Jan 15.

No research was done today. Delivered the THEP seminar, which was very successful.

Jan 16.

No research was done today.

Jan 17.

No research was done today.

Jan 18.

No research was done today.

Jan 19.

No research was done today.

Jan 20.

No research was done today.

Jan 21.

No research was done today.

Jan 22.

No research was done today.

Jan 23.

No research was done today.

Jan 24.

No research was done today.

Today I woke up to the happiest email I have ever gotten: I was accepted by University of Pennsylvania!! I was notified by my interviewer, Prof Heckman. The whole day felt like a dream. Went out to celebrate with Arthur at night.

Jan 25.

No research was done today.

I received a welcome email from Prof Vijay Balasubramanian from UPenn in the morning. I spent most of the day working on General Relativity problem set. At night, I celebrated Chinese New Year with my family.

Jan 26.

No research was done today.

I read the email from Prof Vijay Balasubramania in detail. I got very excited about his work and the prospect of joining his group. I spent the entire day reading about the "It from Qubit" collaboration, which he is part of. I had one of the greatest day of my life, as I realized I was so close to something I have dreamed for a long time: going to a top university, doing research in quantum gravity, especially the exciting new direction of spacetime emergence via entanglement.

Jan 27. Mon.

After a good 2 weeks of being completely swallowed by grad school interviews, giving the THEP Seminar, and QFT problem set, I finally went back to research. I plan to devote an hour or two on research everyday from now on , or else I fear I won't finish by the end of the semester.

Continue Implementing BPS Initial Grid

I went back to the code and reviewed the BPS package. It works quite well in solving a BPS equation and plotting it (with derivative check plots), without an outside user caring about the inside work. In other words, this code is well compartmentalized and packaged.

Next, I started adding a BPS initial grid option to the main solver. The idea is as follows:

1. I guess what the inner vacuua have to be, which is actually two different vacua separated the horizontal axis, with monodromies in a way that give the charge of the quarks.
2. Then I solve the two BPS equations, corresponding to the boundary vacua to the "top" inner vacua, and another BPS soliton corresponding to the "lower" vacua to the outside boundary again.
3. I "stich" together these two BPS solitons, which is a guess of what a slice of the final 2D solution will look like far from the charge.
4. I repeat this double BPS solitons as many times as needed to fill up the space between the two charges, while the fields to the far-left and far-right of either sides of the charges remain at the boundary values.

This initial configuration then guides the code to relax to a double string configuration with vacuua that support the monodromy needed.

Jan 28. Tue.

Continue Implementing BPS Initial Grid

I continued to implement the BPS initial grid, particularly making the solver save BPS result into core dictionary as well as the BPS soliton image saved to the same folder with all the other results. This was actually not easy to do due to my compartmentalizing the BPS solution, and now I need to call a module inside that folder that plots BPS externally. This is unfortunate since I originally envisioned a complete self contained BPS package, but it is not easy to do when I need to control when and where to save the plots.

This actually happens many times during this research. It would be almost impossible to wrote a code base this large, without completely separating out the different modules, class, and functions, so that it is easy to debug and code up one part at a time. But the complete compartmentalization often backfires, when future complication arises or new flexibility of the code to answer or explore some research is required. My code is in a constant tug of war between good code that is well-compartmentalized and solves an existing problem and the need to be flexible.

Good Progress, but Still Some Trouble with SU(3)

When I solve SU(3) with BPS initial grid, the situation is much better than before. The double string result initially looks right, and the field and energy matches that of Andrew's at first. But when I let is run longer, it is apparent that the center vacuum becomes less pronounce, and energy starts sipping into the center. I figure that it might just be because my grid isn't large enough, and with larger distance between the charges, the field has space to settle in the middle.

Tomorrow I shall run SU(3) on larger grid and charge separation with BPS initial grid. Based on today's initial result, the prospect seems good. Hopefully, I will soon finally solve the problem of simulating *one* double string picture, because my project requires me to simulate *many* double string pictures to extract the string tension. My goal is to at least get over this hurdle before Februrary.

Jan 29. Wed.

Today is an incredibly productive day!

Finish BPS Initial Grid

I added the capability to take the vertical middle slice of the final solution, and plot it over the initial double BPS solutions. Although a correct solution needs not to overlap with the initial guess (the peak can shift in position), the height of the peaks of the $\vec{\phi}$ and $\vec{\sigma}$ ought to match those of BPS.

In fact, this actually help me figure when a grid is too small for a solution to have a vacuum in the middle. If the BPS solutions look like the end do not have enough space to properly flatten, then this corresponds to a 2D solution with energy in the middle.

Thus, plotting the BPS actually let me tell beforehand whether my separation (and grid) is large enough for the 2D simulation!

Successfully Solved SU(3)

I figured out a general vacua configuration that supports the monodromy of any charge of the form $\vec{w}_a$, and confirm it by solving SU(3). In brief, I make the top half of the inner vacua to be zero, while the bottom half to be $\vec{w}_p$, where $\vec{w}_p$ is the charge of the quark and p is the N-ality. The outside vacua is $\vec{x}_1$. The two domain walls are therefore 1-walls, which are the lowest energy configurations. The solution of this configuration, for two charges separated by a distance of 10 dual photon mass, is shown in figure 1. The potential energy, of the double string picture, is shown in figure 2.

I now declare that SU(3) double string is finally solved! Although I have yet to try it, I have no doubt exactly the same method and the same monodromy supports the result for all N.

What remains to be done now is an edge effect study, before we can start optimizing and mass compute!

Implemented Continue-Solver

This has been on my mind for a long time and I finally implemented it. The old code requires solving from fresh whenever we want to repeat a calculation but with more loops. Now I made an addition to the code such that when this need happens, it takes the previous final result as the starting configuration and add more loops to the simulation.

This is actually a huge time saver for the rest of the project!

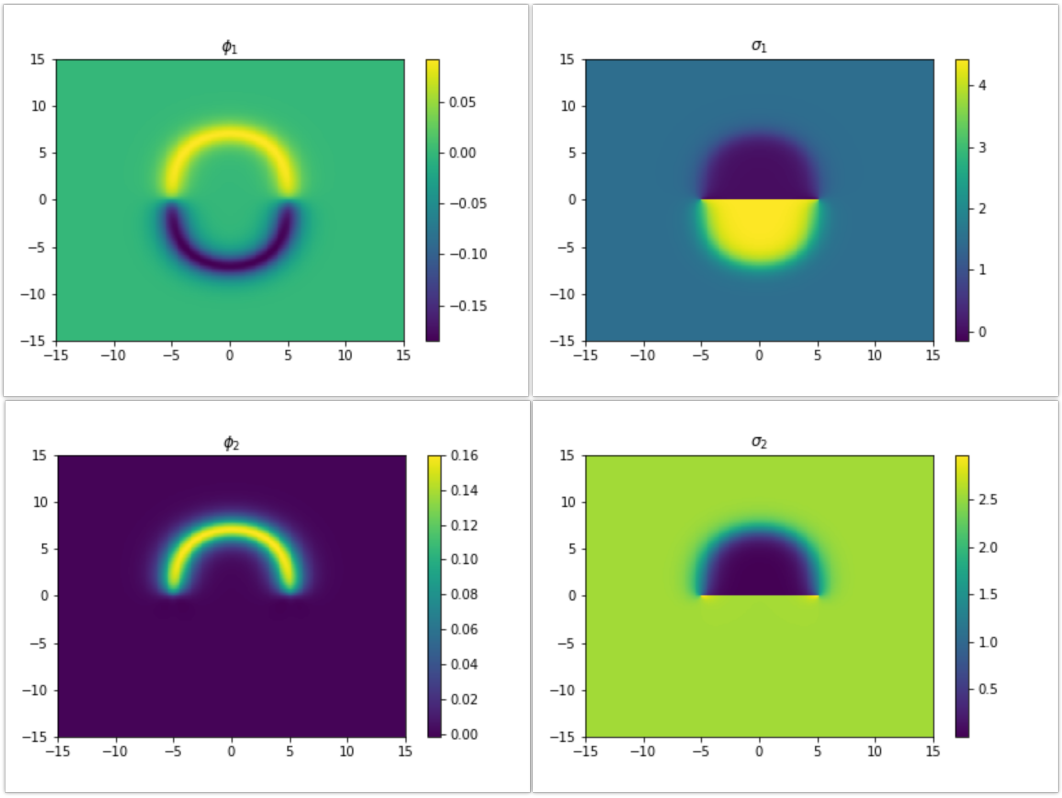

Figure 1: SU(3) Field R=10

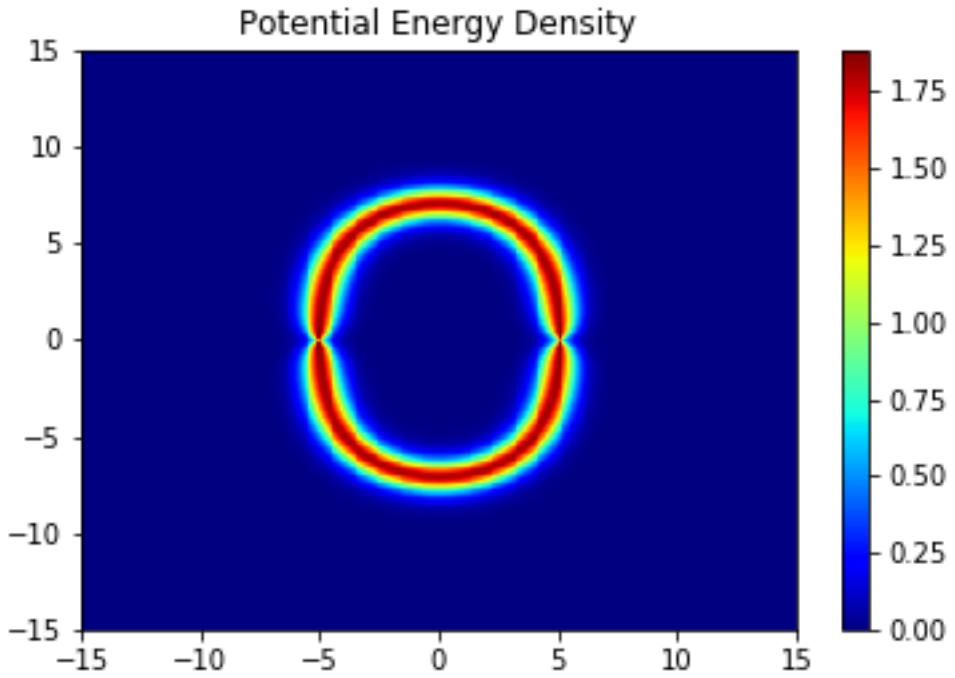

Figure 2: SU(3) Potential Energy Density

Jan 30. Thu.

SU(3) Edge Effect Study

After figuring out how to solve SU(3) confining string yesterday, I immediately set up a edge effect study. To do this, I simulated the confining string picture on a grid of 10 by 10, for a charge separation of $R = 2, 3, \ldots 9$. Then I repeated the calculation on a grid of 20 by 20. (Note that all of these are computed to 400 loops.)

The result is shown below in figure 1.

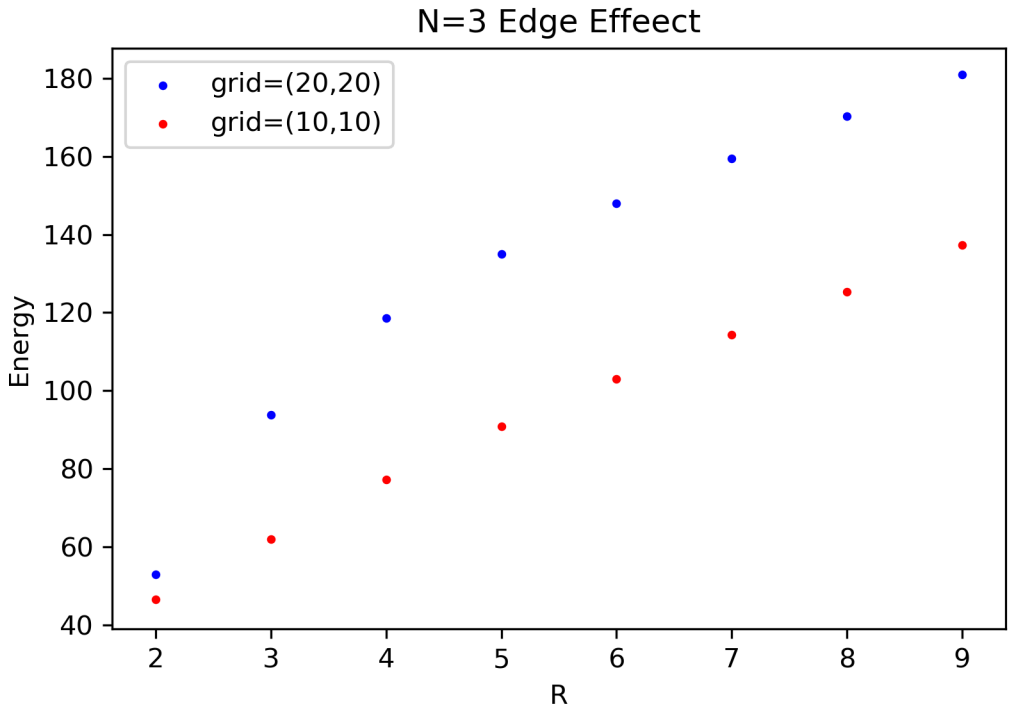

Figure 1: SU(3) Edge Effect

It is clear that the larger grid has an overall larger energy, due to the energy being an integral over all space, and there are just more space to be integrated for the larger grid. However, beside this energy difference, there is very little edge effect. Adding a constant to make the difference clear, see figure 2, it is clear that even close to the boundary, the energy basically agree. In particular, the slope of the energy as a function of separation completely agrees, as is clear from the diagrams.

Surprisingly, the only major difference between the two results are the low-separation energy. The "true" energy has a drop-off from the linear relation, while the small-grid result doesn't. However, this should

not change our result, as the slope of the graph are unaltered in any case.

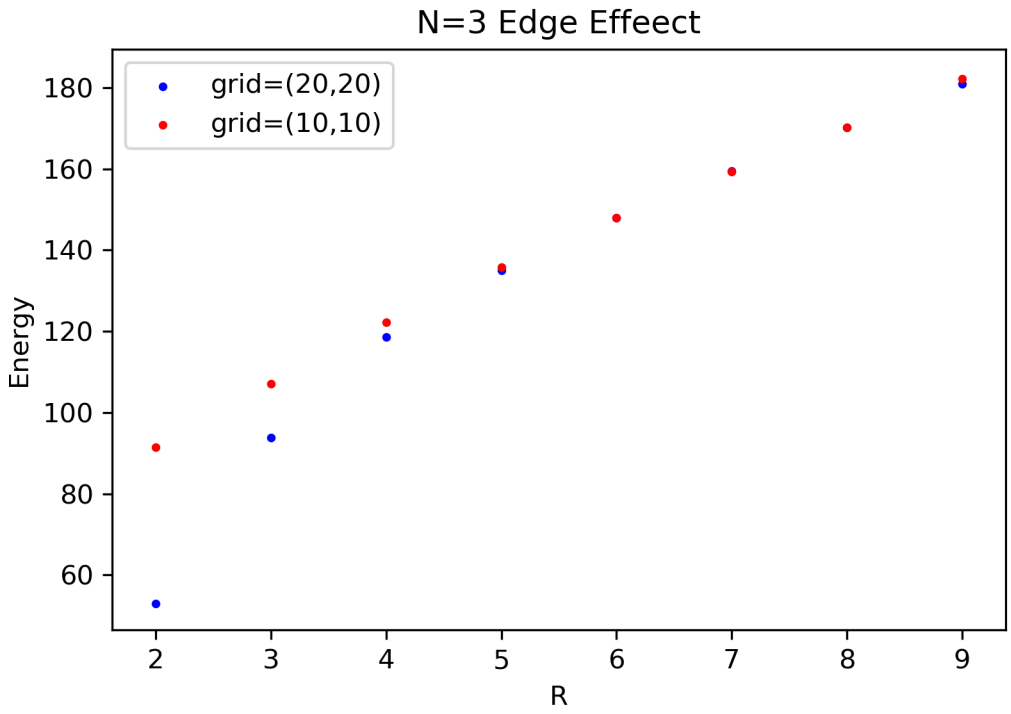

Figure 2: SU(3) Edge Effect shifted

This is overall a very good news for my project. First, it means that the prediction of the linearly rising potential is reproduced by my code. Second, in practice, the agreement implies that there is no need for us to consider edge effect. We just take the points that maitain linear relation and find the slope.

First Meeting with Erich & Explanation of No Edge Effect

After getting this nice edge effect result, I had my first official meeting with Erich on this project. I first spent most of the meeting explaining to him how I solved SU(3). In particular, I explained the monodromy (with 0, $\vec{w}_k$, and k=1 wall) I used to support the charges. We agreed that such configuration is always possible, for N-ality p, where we simply use $\vec{w}_p$, since the two upper and lower BPS walls are related by the $\mathbb{Z}_N$ symmetry and have the same energy.

I then went over the initial configuration using BPS. Erich agrees that this makes sense, but wants me to explore both $k = 2$ wall and starting with initial configuration that is not "guiding" the computer, i.e. try starting with constant initial conditions. I tried this before, and the result doesn't give double string, but in all cases, the time it took for convergence was just too long and I gave up. Perhaps I will study this later when my code gets efficient. Besides, exploring these would be a side-project.

Finally, Erich agreed that my result is consistent with an linearly rising potential. We also realized that in hindsight, it makes sense that a confining potential has no edge effect. After all, at far distance, a confining potential forces the flux out of the vacuum and confine it into flux tubes. So at the boundary,

there is very few flux left. In contrast, a non-confining theory, as we did over the summer by setting $W = 0$, experiences strong edge effect.

Next Steps: February List

I finished most of the points on the New-School-Year list, except for the optimization part. Over the next month, the goal is to optimize the code (carry over from last list), do one two more sanity checks, and start using a supercomputer to get overall results.

1. Optimize the code by using a shortcut equation of motion.
2. Optimize the code by exploiting Numpy to its fullest.
3. Optimize the code by running it on half grid.
4. Sanity check: run the same simulation for a range of separation for SU(3) with N-ality $p = 2$ to check that the slope is the same as the previous case.
5. Start using Supercomputer to solve the rest.
6. Sanity check: run a SU(4) edge effect study (using Supercomputer).

My goal is to finish the first 3 points (optimization) by February 9.

Finish point 4 and get started with the Supercomputer (hopefully with SU(4) result) by the end of Feb 16.

By March 1, I hope that I will have most of the results down.

Jan 31. Fri.

Today I spent some time writing up the journal of the past few days as well as started to think about implementing a fast equation of motion implementation. Not much concrete work was done.

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

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

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

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

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

Then the absolute square is just the products of these two

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

Using the identity (found in our last paper):

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

We can eliminate the sum over b:

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

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

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

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

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 siginificantly 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.

Feb 2. Sunday.

No research was done today.

Feb 3. Mon.

No research was done today. Was accepted to NYU today.