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.