DC solution calculation notes

Saturday, October 29, 2022 15:53

Tweaking Sandia solution

The first solution set attempted is to simply correct the Sandia solution for the short electrode we have, i.e. by setting the voltage on the shorted electrodes to be the voltage on the electrode closer to the ion position. This should definitely be fine for the short between the quantum and loading area but it's questionable if it'll work for the short within loading. It'll also not work for the open and grounded electrodes.

We end up not using this potential since the actual calculation of the potential was way easier than what I was originally told.

Loading the voltage data

Sandia provided potentials generated by each electrodes (including DC and RF) in a *.va file (some trap seems to use .bin extension instead) with 1um resolution around the RF NULL of the trap. The data is stored in the file simply as a huge 4D array (3 spatial dimension and 1 dimension for all the electrodes included) with some metadata stored in the front. The format of the metadata header will not be listed here but can be easily understood either from the mathematica code or from my julia version.

Data fitting

The mathematica notebook from Sandia seems to be fitting all the data based on an interpolation function generated from the data. This seems to be relying mainly on the local (few um) data for characterizing the potential around a certain point. For these reasons I decided to use a fitting method that's more direct and explicitly sensitive to data across a wider range.

Instead of doing differentiation on the interpolation function to determine the property of the potential, I decided to simply fit the data directly. The simplest bases, and the one that matches what we use in the experiment well, is polynomial and the fit is basically a least squares fitting of the potential generated by an electrode to an arbitrary polynomial up to N*M*L order. The fitting range must be at least as large as the fitting orders to avoid over fitting and the range can otherwise be selected to match the desired region of interest (particularly in the X direction). Although the fitting still won't be perfect (since the Lorentzian-like shape of the potential generated by each electrode is expected to have really bad Taylor expansion convergence behavior) this at least allow me to make explicit trade-offs between short range and long range shape imperfections.

Implementation-wise, this fitting is done using matrix division (in julia). The dividend vector in this case is a flattened version of the data to be fitted. Each column of the divisor matrix represents a different term in the polynomial (NML of them in total). The 3D function is flattened in the same way as the vector for each column in the matrix.
I also initially used the formula x^ny^mz^l to generate each column of the matrix for fitting, where x, y, z are the coordinates from the fitting center in unit of um (i.e. grid size). However, it turns out that the matrix generated this way does not work very well during factorization/matrix division and the reason appears to be related to the large value of some of the high order terms (it seems that the code misbehaves roughly when the max term gets close to about 6*10^14). Changing the formula to (x/X)^n(y/Y)^m(z/Z)^l where X, Y, Z are the half size in each direction (i.e. "normalizing the values") fixes this issue.

Find trap RF center/null

Sandia's demo code for trap computation starts with locating the trap center/RF NULL. We can do this by doing a 2D quadratic fit of the RF field along each slice to find where the Y and Z linear terms are zero. The secular frequency at these RF centers can then be determined from the quadratic terms of the RF field.

General optimization procedure

For constructing a solution, we need to make sure the total voltage produced follows the desired shape. We'll generally have more electrodes than the numbers of constraints and we'll allow the extra degrees of freedoms to vary (which corresponds to generating potential shapes that does not matter as much for us) in order to optimize for our objective (most likely minimum voltage / voltage changes). In order to reduce the complexity/memory consumption of the optimization, instead of using the spacial distribution of the potential, I use the numbers from the polynomial fit to describe both the field generated by the electrode and the desired potential shapes.

There are three different methods I've used to do this, each optimizing for slightly different objectives.

1. For a single static potential, optimize for the minimum square sum voltage in order to achieve a certain fixed shape. This is the simplest to do and can be done just with matrix division. In this case the dividend is the representation of the desired potential using the polynomial coefficient (a vector of numbers), and the divisor is the matrix with each column being the polynomial coefficient generated by each participating electrode.
   This method cannot take into account anything but a fixed objective function, e.g. no relaxation of the potential within a certain range. It isn't really compatible with more complex objective either.
2. To make the process more flexible, we could use an actual optimization routine instead. An obvious way to express the constraints on the desired potential is to, well, directly express it as a constraints (i.e. use the voltage on each electrode as the variable and requires linear combinations of them to match a set of numbers). However, since the constraint is linear, it's easier to simply solve the constraint up front (using the first method). This will both give us a set of voltage that satisfy the constraint (i.e. the result of the first method) and a set of vectors (free terms) that can be added to it without any effect on the constraint. We can now treat this as an unconstrained problem where the linear coefficient on the free terms can be any real numbers.

   Moreover, with this method, we could also allow each terms to vary a little, by adding a free term that changes the potential with a limit on the range we can apply that term.

   This method is quite flexible and using the Ipopt solver can be done reasonably fast based on the result of the first method as well.
3. So far, the method we use only optimize for a single voltage condition (i.e. single

frame). For shuttling, we would also like to optimize for certain properties (e.g. smoothness) globally. To do that, we simply combine multiple optimization problems using the second method together with the new objective function being the sum of the previous ones. We can then add other constraints like minimizing the voltage differences between neighboring points.

Based on the experience so far, the following terms combination works the best for the optimization

1. Sum of the max voltage for each frame
2. Sum of the max voltage over a range of frames
3. Sum of the max voltage change between each pair of neighboring frames
4. Sum of the max voltage change over a range of frames

Since the voltage requires changes semi-periodically every 70um due to the electrode structure, for the 2nd and 4th one, it gives the best result when using 70um as the frame range size. Using a range less than the full shuttling range also allow effect of defects (e.g. short electrodes) to not mess up the optimization near the range we actually care about.

Generate compensation term for a single point

This one is very easy to do, and is simply follow the method 1 or 2 without any tweaks needed. I generated two sets of solutions, one for the loading area and one for the quantum. We confirmed that the loading solution works by trapping ion in it and moving the ion left and right by changing the DX term. The quantum one cannot be tested yet without a way to move the ion there.

Generating compensation solution series/shuttling solution

After generating the single solution, the next step is to generate the solutions to transfer the ion to the quantum area. I started this by generating a series of full compensation solutions that are centered at different points along the trap axis. During this process, I noticed that the solution requires very high voltage near the transition area/when the quantum slot opens up.

After checking which constraint needs to be relaxed, I discovered that the blowing up behavior in that region can be completely fixed just by not requiring the ZX term to be zero. It seems that the potential generated by the electrode has some ZX bias in this area which is somewhat consistent with the fact that the RF NULL height is also changing in this area. Using the series of compensation solution, we were able to move the ion up to >200um towards the quantum area from the loading area by switching between different set of solutions while keeping the term coefficient constant (e.g. keeping an X2 = 0.5, ZX = 0.5 while changing the center of the solution by a few um). The distance moved is larger than the loading area size, which confirms that moving the ion out of the two shorted loading electrode is possible. (The shorted electrodes make it harder to push the ion out.)
Based on the series of compensation solution, we can also construct the full shuttling solution that allows moving the ion position in real time. When looking at the solution series, however, it is clear that the solution changes (flickers) a lot. In principle the low pass filter on the electrode is global and linear, meaning the effect of the low pass filter is equivalent to applying the same filter on the potential in space. When this is the case, the fact that some electrode flickers causing the potential away from the ion to change doesn't really matter for the ion. In practice, however, I expect the inaccuracy in the potential data / calculation, difference in filtering function on different electrodes and potentially other effect may cause a shuttling solution with larger changes of electrode voltages between frames to behave worse/deviate more from design, compared to a solution that varies slowly between frames. To do this, I switched to the method 3 and added global smoothing by minimizing the voltage changes between frames.

Using the resulting solution, we were able to shuttle the ion from loading to quantum and back.