# Laser Calibration Software Challenge

## Context

Here at IonQ, we make money by shooting lasers at ions suspended in an electric field. The more accurate the laser is, the more money we make (more or less). Because of various mechanical/optical drifts over time, we have to recalibrate the alignment of our lasers fairly often.

One such programmatically controlled routine for doing this involves rotating a mirror with a software controlled mirror mount over a certain range (“a scan” is the local jargon). One high level algorithm that accomplishes the procedure is:

```
ion_responses = {}

for pos in range(start, stop, step):
  move_mirror_to_position(pos)
  ion_response = measure_ion_response()
  ion_responses[pos] = ion_response

return pick_best_position(ion_responses)
```

`move_mirror_to_position`

This function would send a device command over a serial interface, and would rotate the mirror mount to the new angle, changing the path of the beam slightly and ultimately moving it across the ion chain.

`measure_ion_response`

Measuring the ion’s response is a fairly involved process, but essentially involves

1. shoot a laser pulse at the ion that’s expected to make the ion “bright”
2. shoot another laser pulse at the ion from a different, already calibrated laser that will make the ion give off a photon if in the “bright” state
3. repeat about 100 times
4. the percentage of the time the ion gives off a photon is roughly how “good” the ion response is

## Requirements

Using mirror movement and ion measurement as stubs (as in the pseudocode above), write up an algorithm that finds the best mirror mount position given a certain ion response.

1. mirror mount positions go from 0 to 1 (or any arbitrary continuous range really)

2. ion responses go from 0 to 100 photons per measurement round (or any arbitrary , small, integer range)

You don’t necessarily have to follow the high level algorithm above. You can move the mirror to any position you want whenever you want, and measure whenever you feel like. If you do take the algorithm above, the routine for picking the best position would need to be able to choose a position that potentially hadn’t actually been visited (i.e. between two steps of the scan) through some kind of interpolation. This interpolation would be most accurate if it assumed some kind of distribution in the ion response as a function of laser profile.

Feel free to use any curve fitting/regression/optimization library you can find, but leave a comment explaining what the regression/fitting routine does and why you used it.

# Extra Credit

If you’re feeling sassy, or you want to boost your resume, implement the above as a scan over multiple mirror mounts, all composed in series. One mirror mount would serve as the “x” axis, and another would serve as the “y” axis. A kind of multidimensional optimization, some would say.