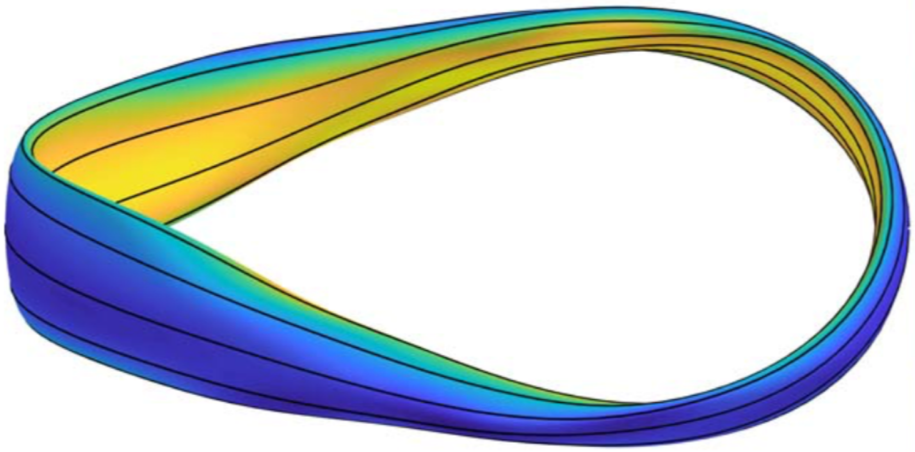
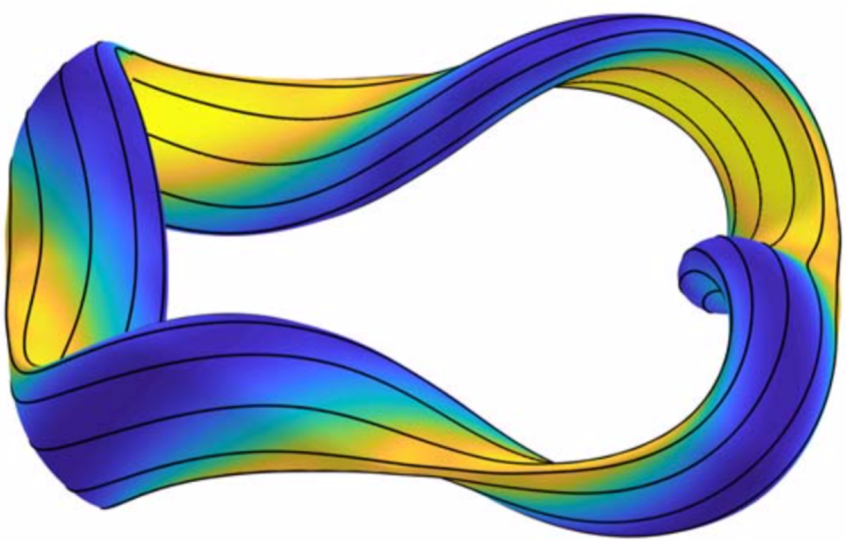
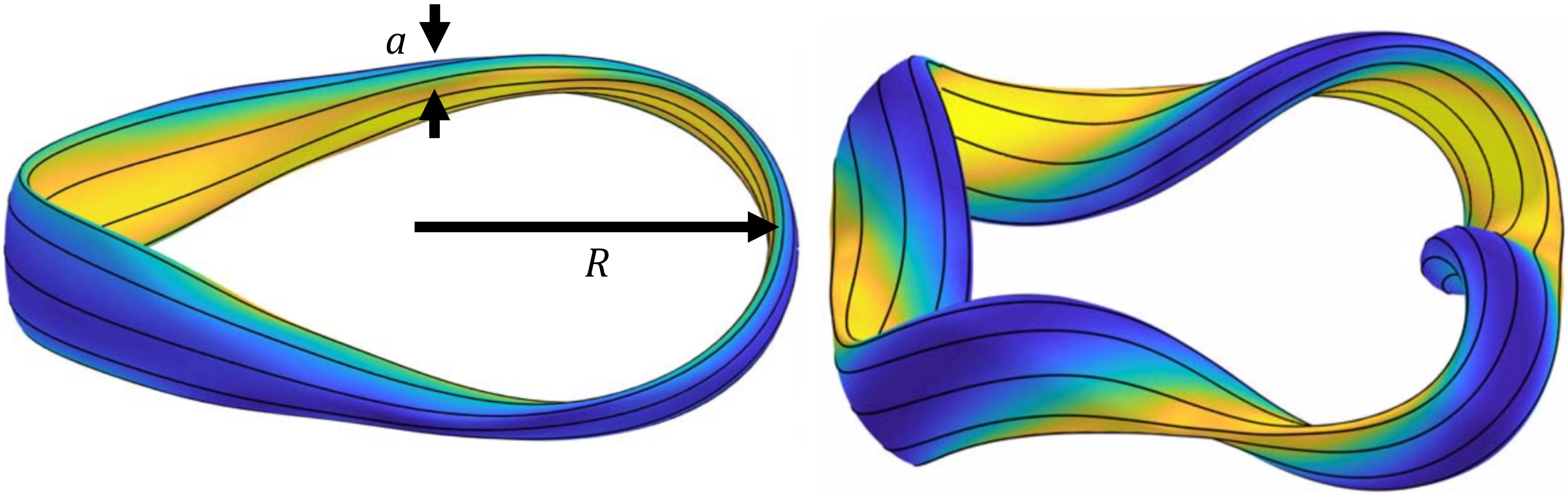
Optimizing O($(a/R)^2$) solutions of the near-axis quasisymmetry equations



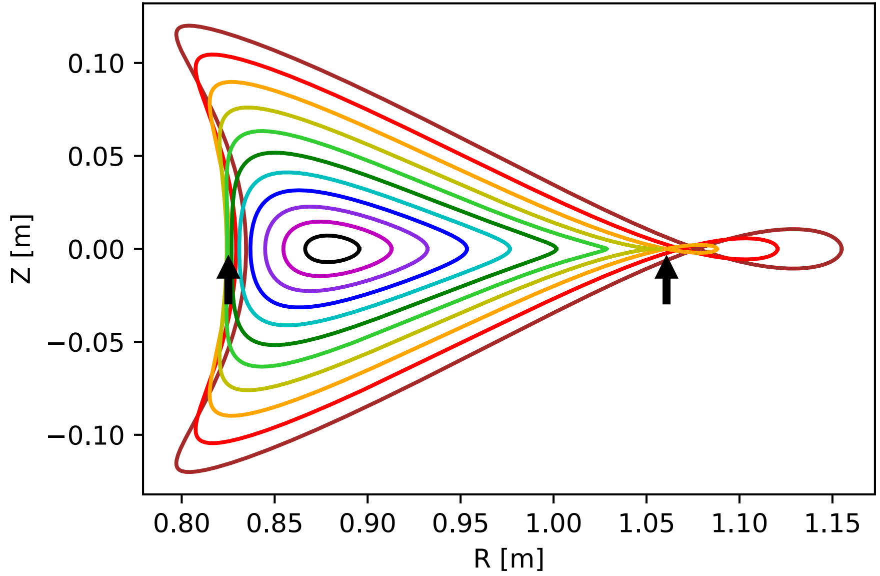


- Expansion about magnetic axis gives huge speed-up for solving MHD equilibrium, reduces dimensionality of parameter space, & can guarantee approximate quasisymmetry.



$$a\,/\,R \ll 1$$

- Expansion about magnetic axis gives huge speed-up for solving MHD equilibrium, reduces dimensionality of parameter space, & guarantees approximate quasisymmetry.
- But, at $O((a/R)^2)$, most solutions are limited to small plasma volume:

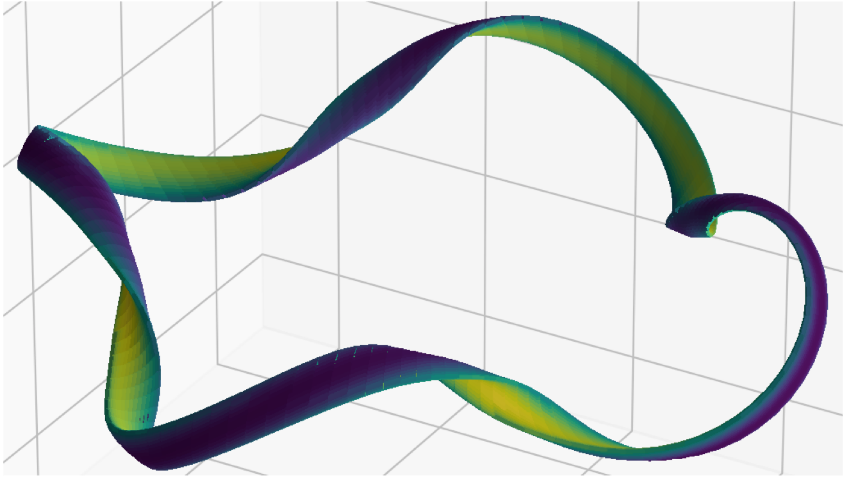


- Also we want other properties: MHD stability, enough rotational transform, etc.

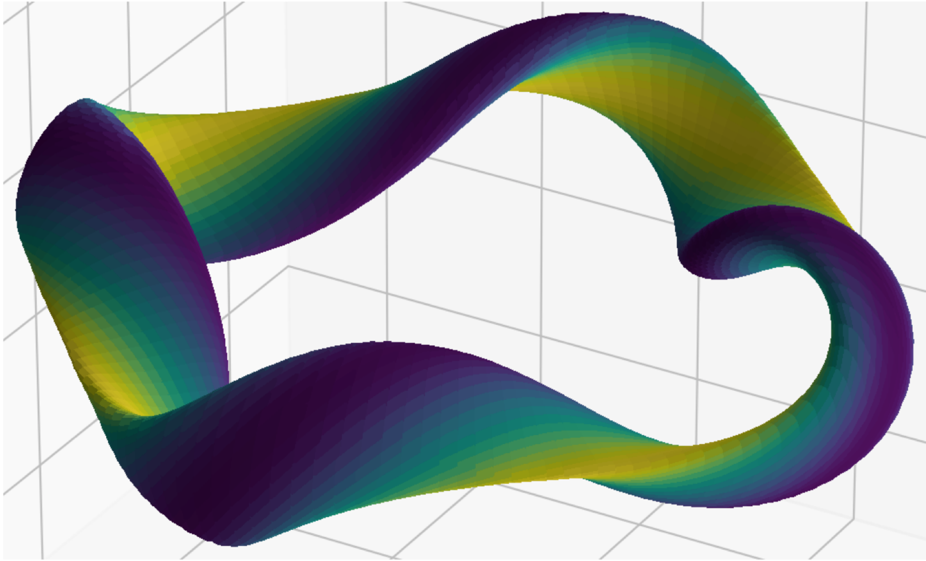
  $\Rightarrow$ Need to optimize the near-axis solutions.

Near-axis stellarator solutions need optimization

Most near-axis solutions look like this:



We want to find the rare solutions like this:



Optimizing near-axis stellarator solutions

- Parameter space is 8D – 12D: Fourier modes of axis shape + 2 scalars.
- Objective function is fast to compute ($\sim$ 1 ms).
- Multiple competing objectives and constraints.
- Optima are extremely narrow.
- Gradients not available now. Finite-difference isn't too bad, since not many dimensions.
- Don't know bounds on parameters. Short-wavelength modes in axis shape are <<1, but not sure how small exactly.

$$
\begin{aligned}
R_0(\phi)\ \ (\mathrm{m}) &= 1 + 0.1700\cos(4\phi) + 0.01804\cos(8\phi) + 0.001409\cos(12\phi) \\
&\quad + 0.00005877\cos(16\phi), \\
z_0(\phi)\ \ (\mathrm{m}) &= 0.1583\sin(4\phi) + 0.01820\sin(8\phi) + 0.001548\sin(12\phi) \\
&\quad + 0.00007772\sin(16\phi),
\end{aligned}
$$

Multiple objectives/constraints

Many could be considered either an objective or inequality constraint.

Large r_c (Minor radius at which singularity occurs)

Iota > 0.4

V'' < 0 ("magnetic well" for MHD stability)

R > 0.3 <R> (Space for coils in the donut hole)

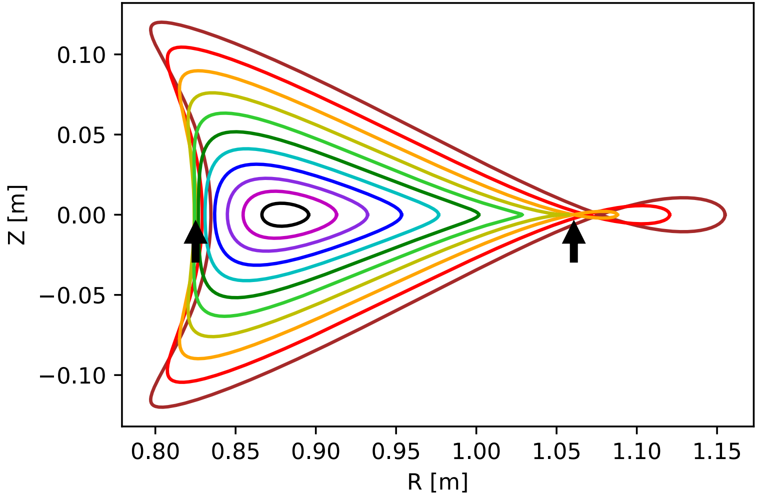
Elongation < 8

Small B_{20} (deviation from quasisymmetry)

Large $L_{\nabla B}$, $L_{\nabla\nabla B}$ (scale lengths in magnetic field, proxy for coil simplicity & distance)

Small X_2, Y_2, X_3, Y_3 (high-order terms in surface shapes)

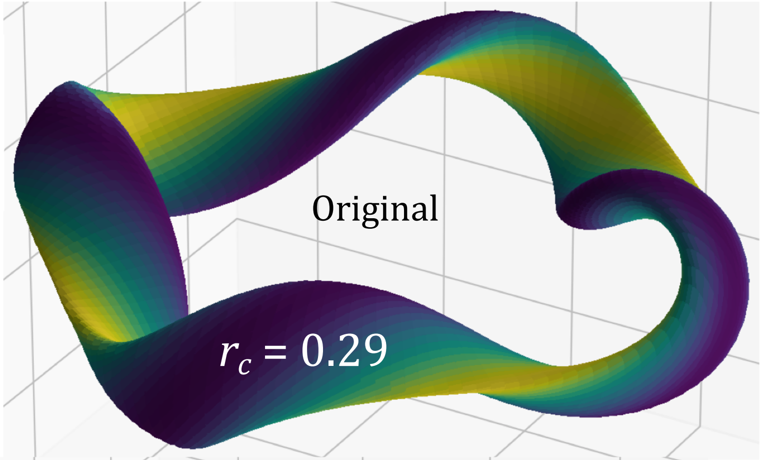
All but iota & V'' are functions of toroidal angle.

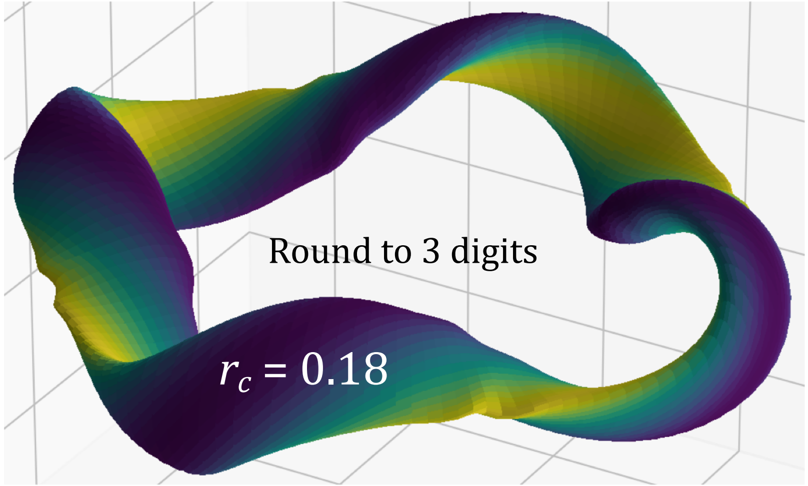


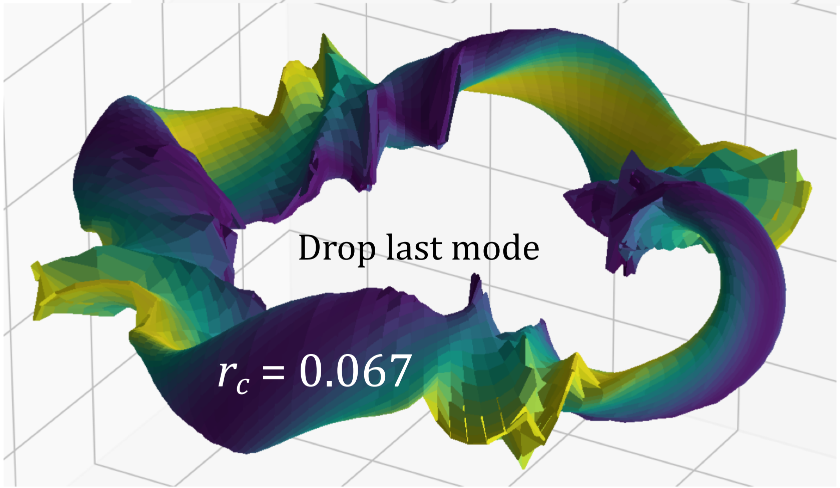
Optima are extremely narrow

$$R_0(\phi)\ \ (\mathrm{m}) = 1 + 0.1700\cos(4\phi) + 0.01804\cos(8\phi) + 0.001409\cos(12\phi) + 0.00005877\cos(16\phi),$$

$$z_0(\phi)\ \ (\mathrm{m}) = 0.1583\sin(4\phi) + 0.01820\sin(8\phi) + 0.001548\sin(12\phi) + 0.00007772\sin(16\phi),$$



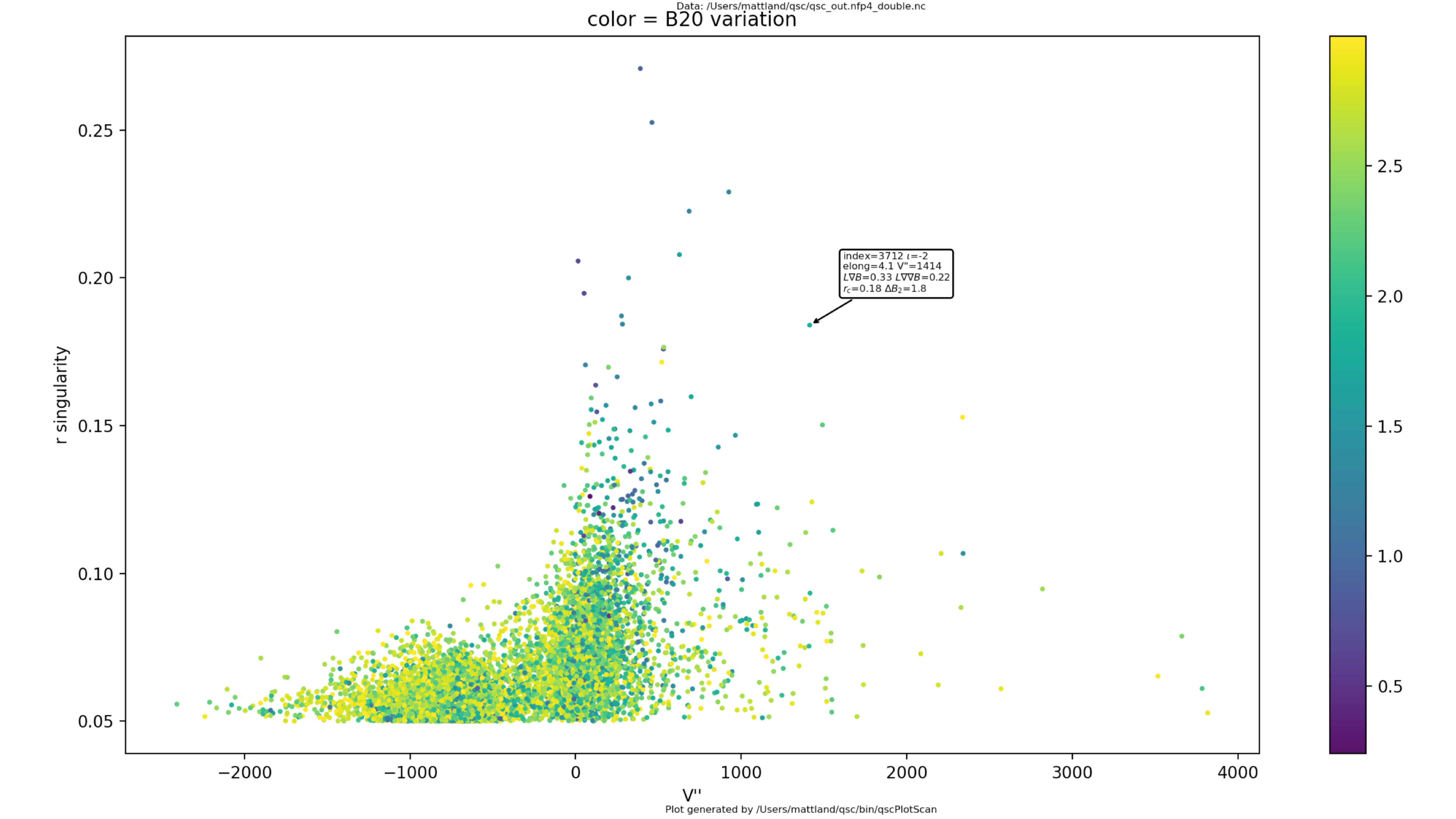




Brute-force searching can generate many starting points for optimization

```
Summary of scan results:                           (fractions in parentheses)
  Configurations attempted:               1587481591
  Rejected due to crude R0 check:                  0 (0)
  Rejected due to min_R0:                          0 (0)
  Rejected due to max curvature:            21579484 (0.01359)
  Rejected due to min iota:                        0 (0)
  Rejected due to max elongation:         1352349439 (0.8519)
  Rejected due to min L_grad_B:             31390372 (0.01977)
  Rejected due to B20 variation:           181991926 (0.1146)
  Rejected due to min L_grad_grad_B:          142016 (8.946e-05)
  Rejected due to d2_volume_d_psi2:                0 (0)
  Rejected due to DMerc:                           0 (0)
  Rejected due to r_singularity:               16018 (1.009e-05)
  Total rejected:                         1587469255 (1)
  Kept:                                        12336 (7.771e-06)
```

384 cores for 30 minutes. Averaged 0.4 ms / configuration evaluated.



Data: /Users/mattland/qsc/qsc_out.nfp4_double.nc
color = B20 variation
index=3712 ι=-2
elong=4.1 V"=1414
L∇B=0.33 L∇∇B=0.22
r_c=0.18 ΔB_2=1.8
r singularity
0.05
0.10
0.15
0.20
0.25
−2000
−1000
0
1000
2000
3000
4000
V''
0.5
1.0
1.5
2.0
2.5
Plot generated by /Users/mattland/qsc/bin/qscPlotScan

Several implementations available

`https://github.com/landreman/qsc`

- C++
- Can do brute-force searches with MPI
- Can do optimization via GSL

`https://github.com/landreman/pyQSC`

- Pure python
- Much slower than C++, but easy to install from PyPI (pip install qsc)

Questions

- What is a good workflow? E.g.
    1. Brute-force search.
    2. Another brute-force search with tighter bounds.
    3. Extract the non-dominated points.
    4. For each such point, now scan over the weights for each objective.
    5. Optimize.
    6. From the resulting set, extract the non-dominated points.

- Recommendations for algorithms or libraries to use?