Simulating Percolation Thresholds

MIT, NMM 2023.864

Scott Kirkpatrick

Objectives of the project

To learn stuff, not change the world

- What does the Pythonic environment offer to simulation?
  - For which problems do GPUs eliminate performance limitations?
- Try these tools on a simple phase transition/disordered system which never got properly resolved when the problem was hot.
- Use current visualization at scale tools to get a feeling for current flow in composites.
- Published studies did their best with M68000, Intel 386, Vax or IBM 168, so now...?

Physics questions to address

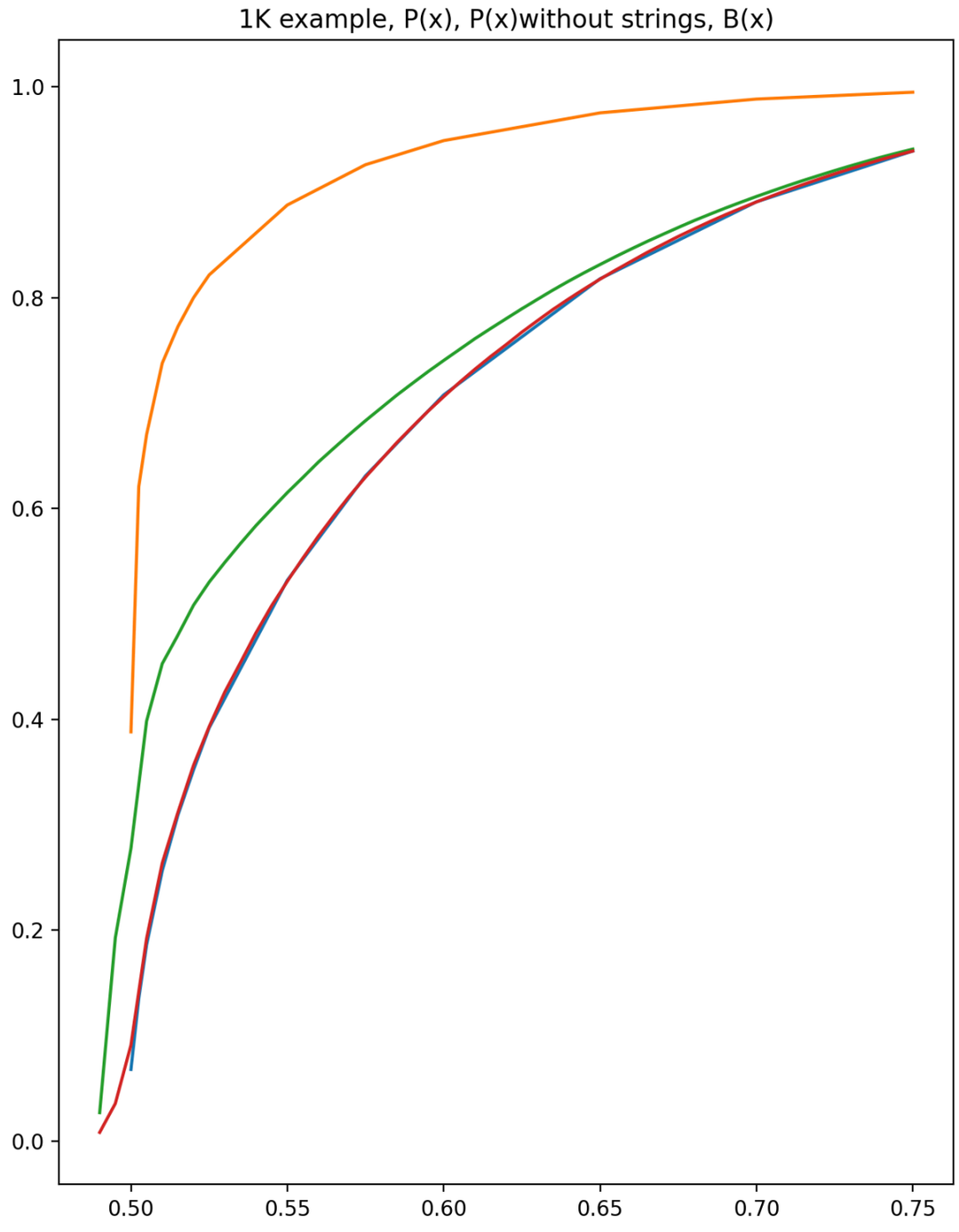
- What makes conduction vanish at a percolation threshold?
  - is it restriction to a smaller volume or torturous paths or both? (There is a literature of adding together exponents for each .)
- The problem is easy to model (punch holes in conducting sheet) but computer simulation of a network of conductors is best.
- 2D bond percolation is the easiest. Threshold at 50% missing bonds, symmetry about this point.
- How much of the ~10 orders of magnitude performance since 1990 can we use/do we need to use.

Where can the current flow?

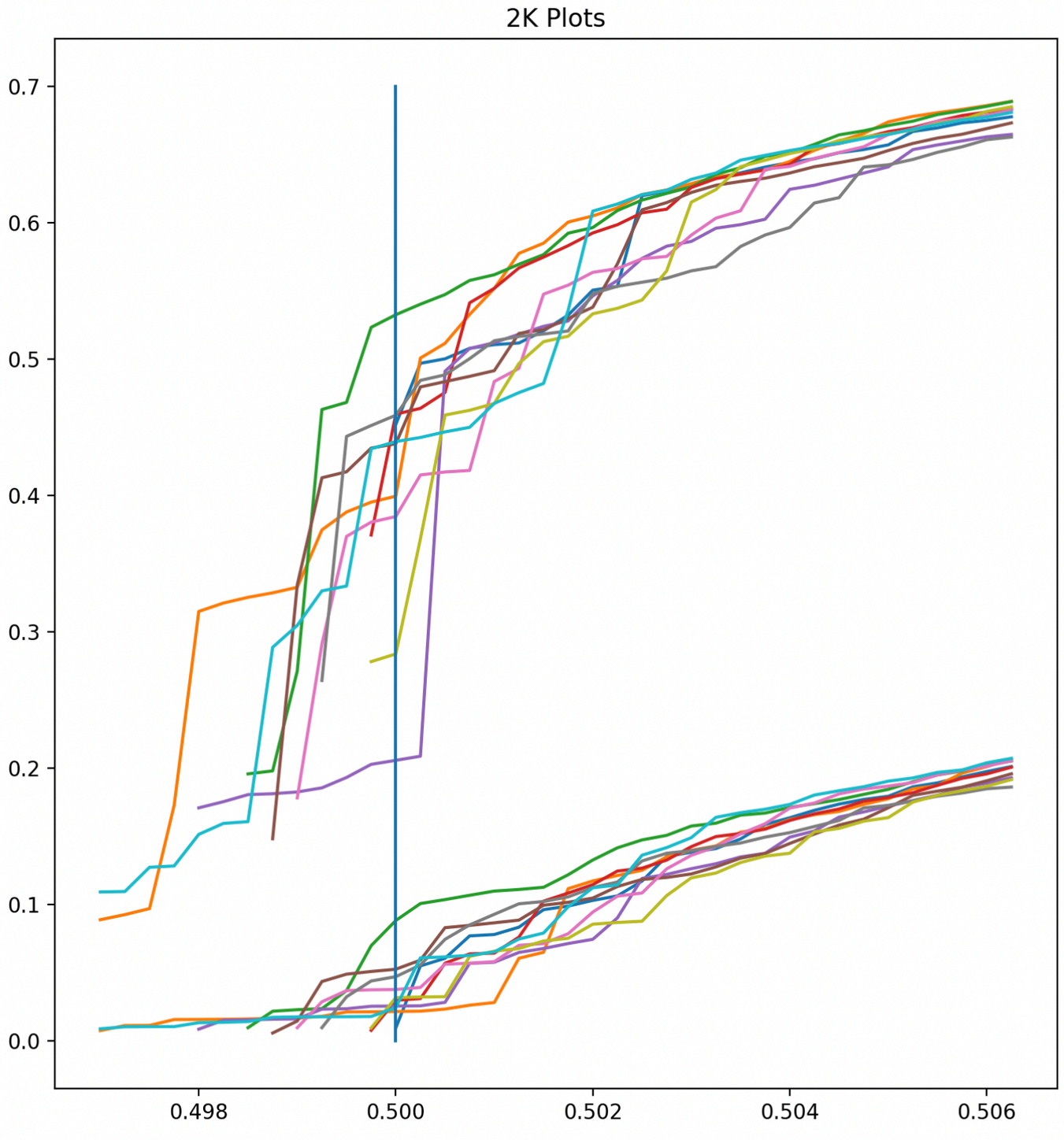
- P = largest connected component
- But 1-d strings don't help. Cut them off (green line)
- B = largest bi-connected component

- In a sufficiently large system, B is where the current flows. But corrections due to edges, contacts, etc. are large.

- NetworkX has good compiled code for identifying P and B. It's very space inefficient, and SLOW for detailed exploration of cluster, such as solving Kirchhoff's Law.
- Networks returns its discoveries as a set, in some random order, so you need to np.fromiter to get back into nparrays.
- Thresholds are very noisy close to a critical point, and searches do not parallelize well.

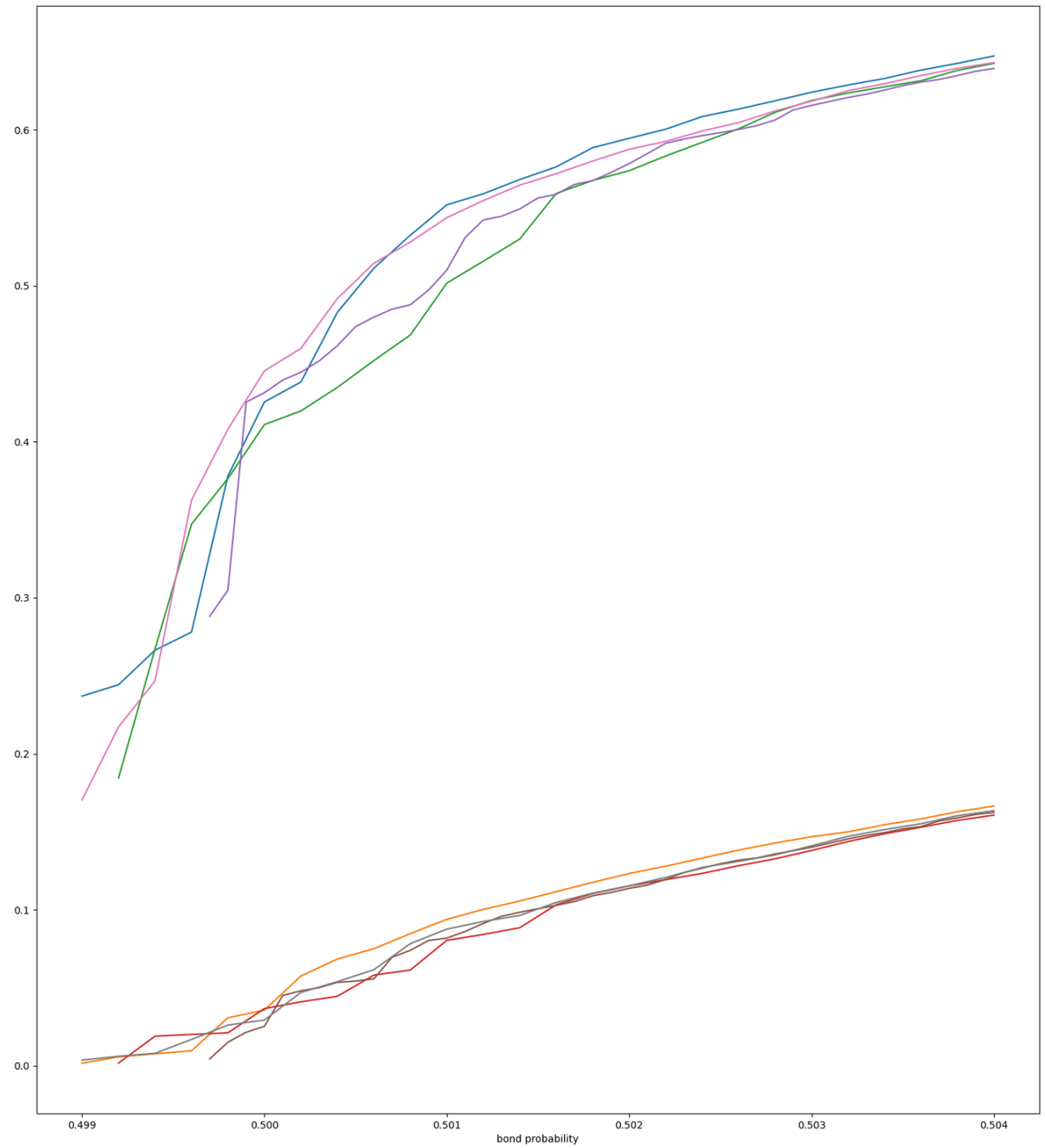


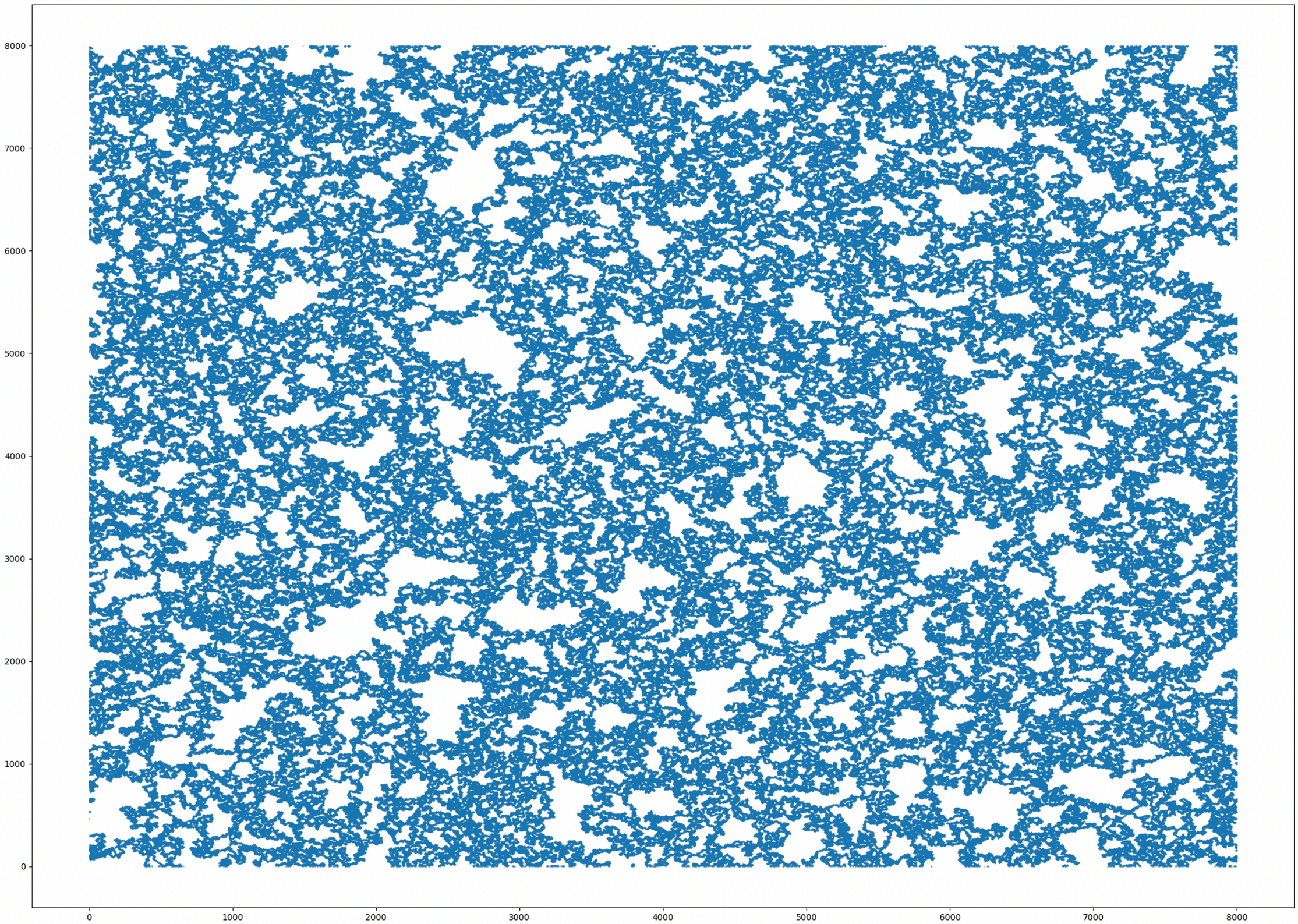
Percolation thresholds are noisy

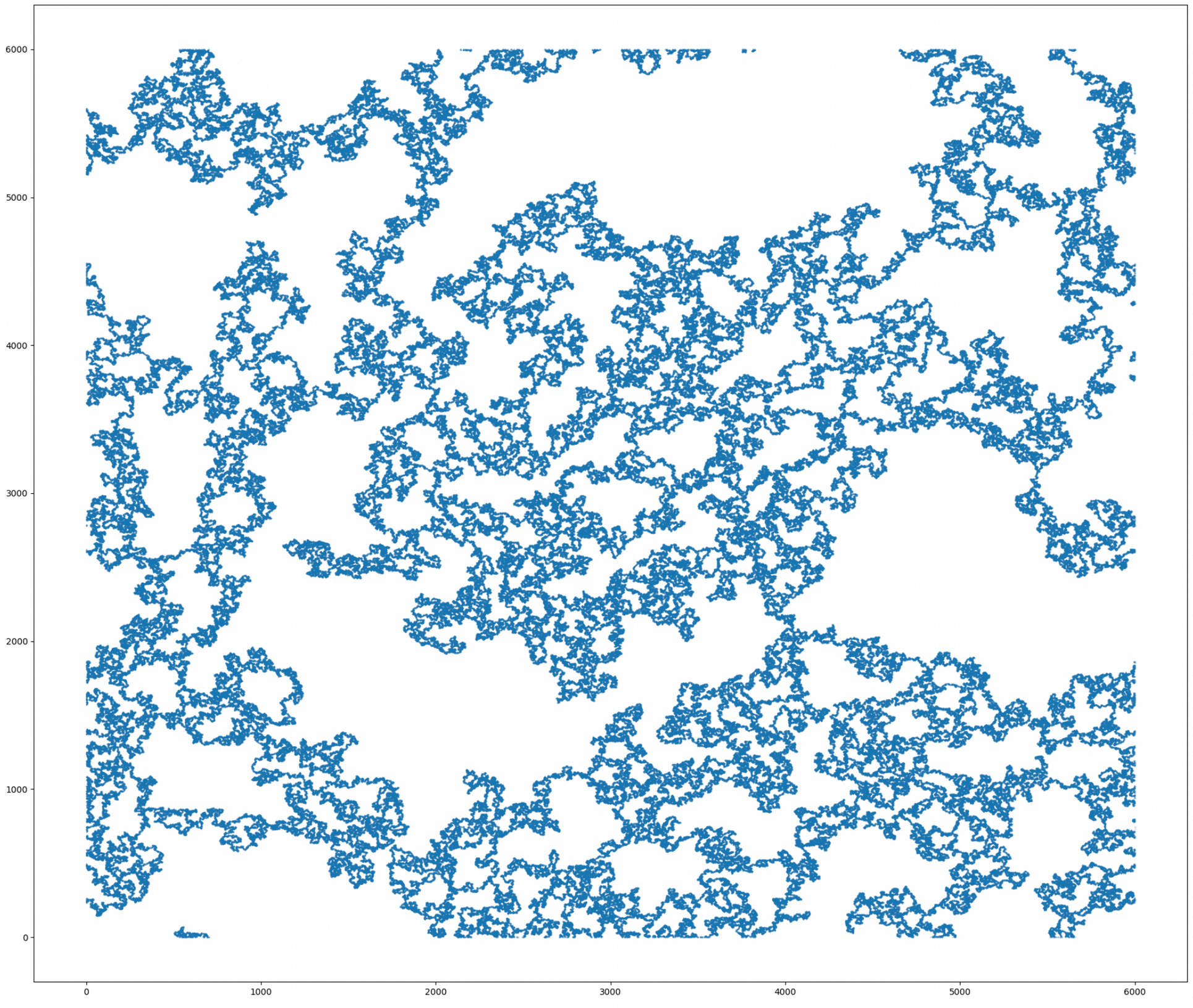


Biggest Sample on a laptop = 10^8 nodes

- P (above), B(below) both extend below the limiting threshold.
- What do these paths or grids actually look like?
- Are they channels or fractal?




8000
7000
6000
5000
4000
3000
2000
1000
0
0
1000
2000
3000
4000
5000
6000
7000
8000


6000
5000
4000
3000
2000
1000
0
0
1000
2000
3000
4000
5000
6000

What's next —calculating conductances

- Gauss-Jacobi, with over-relaxation
  - easy to parallelize, very slow convergence
- Exact solution with LU factors. (will it work on 10^8 or more scale?)
- Real-space renormalization — reduce the whole network to two conductances, one x one y. Has been done on samples with open boundary conditions, and N <= 10^6. Should be parallelizable.
- Averages still needed, since results are anisotropic as well as varying.

Python vs Good Old days

- Just searching a graph — BFS, DFS, DFS with backtracking, Python and NumPy don't seem to offer much. NetworkX code is good, but building a graph takes a long time. Solving equations on a random graph is where CUDA and TaiChi offer advantages.