Code written for "Feedback Induced Phase Transitions in Active Heterogeneous Conductors"

Samuel A. Ocko[1] and L. Mahadevan[2]

[1]*Department of Physics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*
[2]*School of Engineering and Applied Sciences, Department of Physics, Harvard University, Cambridge, Massachusetts 02138, USA*

There are three parts of the program:

1. The core part of the program is written in C++, which contains the simulations of the system and the linear equation solving required to compute currents.

2. Another part of the program is a wrapper for the first part, written in Objective-C, and is used for visualizing the system, as well as playing around with different parameter values.

3. The last part of the program is written in MATLAB and is used to analyze the data generated by the first and second parts.

Note that in the programs, evapWindTax is used interchangably with the removal coefficient α_{R} ("Evaporation" of particles), while condWindTax is used interchangeably with the addition coefficient α_{A}("Condensation" of particles).

I. C++ CODE

A. MyCircuit

Translates a list of cell resistances and driving voltages into a set of linear equations, solves it, and then translates the solution back into a list of currents between cells. Starting with our basic equations

$$I_{ij} = \frac{1}{\Omega_{ij}}[V_i - V_j + F_{ij}], \qquad \mathrm{DIV}_i = \sum_j I_{ij} = 0.$$

Where, in this case, $F_{ij} = [\vec{r}_j - \vec{r}_i] \cdot \hat{z}$. This gives us a linear equation to solve:

$$\kappa\vec{V} + \vec{\mathrm{B}} = \mathrm{D}\vec{\mathrm{I}}\mathrm{V} = 0$$

Where

$$\mathrm{B}_i = \sum_j \frac{F_{ij}}{\Omega_{ij}}$$

And κ is a sparse, positive-semidefinite symmetric matrix. Therefore, this set of linear equations which may be solved using the Eigen::SimplicialLDLT⟨SpMat⟩ object [1]. Once we solve for the relative values of $\vec{V}$, we can reconstruct $\vec{I}$.

It is also possible to have cells connected to a fixed voltage source through a variable conductance or to have a fixed, nonzero divergence, but these features are not used for the simulations done in the paper.

B. MyAPMSystem

Maintains a state of the system, as well as simulating the dynamics of the system. The decision of which sites to remove particles from and add particles to is done through rejection sampling. MyAPMSystem contains an instance of MyCircuit in order to solve for the current through the network.

C. MyAutomatorMode

Used to perform automated runs through different parameter values. The high-level idea is:

1. Load data from a txt file into MyAutomatorMode, which determines which parameters will be iterated through.
2. Modify the MyAutomatorMode object to determine whether it will perform its own simulations or simply read the results of another simulation.
   (a) SimulateAndSnapshotizeMode will make the automator perform simulations and output a snapshot image of the final state
   (b) linuxizeMode(mode) will make the automator perform its own simulations, but not output any images. It additionally changes the path to be the local path. (Called linuxize mode because the simulations in the paper were done on linux clusters).
   (c) Snapshotize(mode) will make the automator read the data and render a snapshot of the final state of the system.
   (d) videoizeMode(mode) will make the automator read the data and render many images describing the evolution of the system.
3. Use MyAutomatorMode to fill a list of SamsSingleRunParams, each which describes the parameters for a single run.
4. When iterate() is called, iterate through another item on the list, and run a single simulation.

Five files are outputted:

- *SingleStatePlot ....txt*: Describes the final state
- *EnsemblePlot....txt*: A list of system states after equilibration
- *EvolutionPlot....txt*: A list of system states from the beginning to the end.
- *Conductivity ....txt*: A single number with mean conductivity of the system.
- *Snapshot ....tiff*: A single number with mean conductivity of the system.
- *Frames .../ Frame ....tiff*: A single number with mean conductivity of the system.

  Mean field data MyAutomator also has a method called generateMFTFiles(), which generates lists of conductivity as well as samples of v_i/κ for both uniform densities as well as densities varying over short length scales. These files are then used by the MATLAB code to predict phase separation.

II. OBJECTIVE-C CODE

A. MyView

Visualizes the state of MyAPMSystem, and can also save pictures to a tiff file.

B. SamsChannelizationSolvingAppDelegate

Interfaces the Objective-C User interface with the C++ Core code. Mostly just data plumbing.

III. MATLAB CODE

Note; Permeability is used interchangeably with conductivity in the MATLAB code.

A. doEverything

The highest-level method, calls all the other methods.

B. MFTEinsteinCriteria

Takes the mean evaporation and condensation rates to predict the amplitude of different fluctuations.

C. UniformPhaseSeparationPlot

Uses the MFT evaporation and condensation rates to predict a phase separation.

D. CrystalPhaseSeparationPlot

Uses the MFT evaporation and condensation rates for the thinnest possible structures to predict a phase separation.

E. isBimodal

Decides whether a histogram with error bars is bimodal to a statistically significant degree.
If anything isn't clear or you have questions about the code, please do not hesitate to contact samocko@gmail.com.

---

[1] The entire Eigen package is contained within a subdirectory of the code.