Homework 6

Biological Physics

Due Wednesday, May 11, 2016

The Vertex Model

For this homework, you will simulate a network of epithelial cells using the vertex model.

Additional Files

1. `random_network.csv`: A 48×2 array of initial positions for the network of 24 cells with 48 vertices, generated by taking a hexagonal lattice and randomly shifting the vertices. This contains the positions of all 48 vertices, but does not tell us which vertices are attached to which other vertices.

2. `random_network_cells.csv`: A 24×6 array of the vertices of each cell. Each row is a cell with 6 vertices, with the 6 numbers corresponding to the 6 vertices of each cell, counter-clockwise around the cell. This file tells us which vertices are attached to which other vertices. You may want to read this in with the following code in order to get the cell array we will need later on in this assignment:

```
cell_arr = csvread('random_network_cells.csv');
cells = num2cell(cell_arr, 2);
```

3. `transitionT1.m`, `transitionT2.m`, and `transition.m`: code that will take a network of cells, and attempt to perform T1 and T2 transitions on the network. You will only need to call the `transition(...)` function in `transition.m`, but `transitionT1.m` and `transitionT2.m` need to be present in order for the code in `transition.m` to call those functions. This code will test all 3-vertex cells for T2 transitions, and all edges with a length less than the `lowdist` parameter for T1 transitions, and when a transition would be favorable (decrease total energy), the transition is applied and the function returns. See `help transition` for more details on how to use this function.

4. `drawpolys.m`: code for drawing polygons in a manner similar to Farhadifar *et al.*.

The two functions provided (`transition.m` and `drawpolys.m`) have some documentation you can access with e.g. `help transition` or `help drawpolys`.

The initial network in `random_network.csv` was created with periodic boundary conditions, with width $L_x = 9\sqrt{\frac{2}{3\sqrt{3}}} \approx 5.58$ and height $L_y = 4\sqrt{\frac{2}{\sqrt{3}}} \approx 4.3$, the width and height of a box of 4×6 hexagonal cells with area $A_0 = 1$.

Energy and Forces in the Vertex Model

The energy function in Farhadifar *et al.* (2007) is given by the sum of three terms:

$$\begin{aligned} E &= E^{\text{elasticity}} + E^{\text{tension}} + E^{\text{contractility}} \\ &= \sum_{\alpha} \frac{K}{2}\left(A_\alpha - A_0\right)^2 + \sum_{\langle i,j\rangle} \Lambda \ell_{ij} + \sum_{\alpha} \frac{\Gamma}{2} L_\alpha^2, \end{aligned}$$

where α is the cell number, i and j are vertices, $\langle i, j\rangle$ means all distinct pairs of vertices that are bonded, and L_α is the perimeter of cell α.

To simulate this model, we will need to calculate the forces on each vertex, and move the vertices in the direction of the force. To do this, we will need to take the derivative of the energy E with respect to the x and y coordinates of each vertex i in the simulation:

$$\begin{aligned} \vec{F}_i &= -\vec{\nabla}_i E \\ &= \begin{pmatrix} \frac{dE}{dx_i} \\ \frac{dE}{dy_i} \end{pmatrix} \end{aligned}$$

Elasticity In class, we derived the force on a single vertex due to elasticity:

$$\begin{aligned} \vec{F}_i^{\text{elasticity}} &= -\vec{\nabla}_i E^{\text{elasticity}} \\ &= \sum_{\alpha} K\left(A_\alpha - A_0\right) \vec{\nabla}_i A_\alpha, \end{aligned}$$

where we need the further derivative

$$\vec{\nabla}_i A_\alpha = \begin{cases} -\frac{1}{2}\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\left(\vec{r}_{\alpha,i+} - \vec{r}_{\alpha,i-}\right) & \text{vertex } i \text{ is part of cell } \alpha \\ 0 & \text{otherwise} \end{cases}$$

where $\vec{r}_{\alpha,i+}$ and $\vec{r}_{\alpha,i-}$ are the locations of the two vertices counter-clockwise and clockwise around cell α relative to vertex i, as in Figure 1, and $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\left(\vec{r}_{\alpha,i+} - \vec{r}_{\alpha,i-}\right)$ represents matrix multiplication (`A * B` in Matlab).

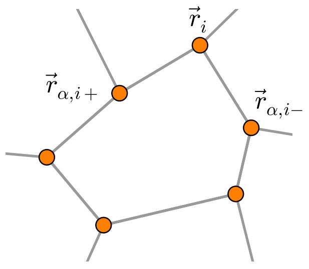

Fig. 1: An example cell, with three vertices labeled.

Tension Similarly, we derived the tension force:

$$\begin{aligned}\vec{F}_i^{\text{tension}} &= -\vec{\nabla}_i E^{\text{tension}} \\ &= -\sum_{\langle i,j\rangle} \Lambda \vec{\nabla}_i \ell_{ij} \\ &= -\sum_{\langle i,j\rangle} \Lambda \hat{r}_{ij},\end{aligned}$$

where $\hat{r}_{ij} = \frac{\vec{r}_i - \vec{r}_j}{|\vec{r}_i - \vec{r}_j|}$ is the *unit* vector in the same direction as the vector separating vertex i from vertex j.

Contractility For the contractility term, we have

$$E^{\text{contractility}} = \sum_\alpha \frac{\Gamma}{2} L_\alpha^2$$

L_α is the perimeter of cell α:

$$L_\alpha = \sum_{i \in \alpha} \ell_{\alpha,i,i+},$$

where $\ell_{\alpha,i,i+}$ is the distance from vertex i to the neighbor of i counter-clockwise from i around the cell. Below, you will calculate the force on vertex i from the contractility term.

Minimizing the Vertex Model

1. **[Output]** Load the attached `csv` files, and plot the locations of all vertices, as well as the connections between them. See Figure 2 for an example.

2. Load the attached `csv` files, and using the formulas above, calculate the energy for the given initial conditions, to verify the following values:

$\frac{E^{\text{elasticity}}}{KA_0^2}$	$\frac{E^{\text{tension}}}{KA_0^2}$	$\frac{E^{\text{contractility}}}{KA_0^2}$
0.549544	$50.2424 \frac{\Lambda}{KA_0^{\frac{3}{2}}}$	$212.21 \frac{\Gamma}{KA_0}$

   Note that our simulation units assume that $A_0 = 1$, so if you use $K = \Lambda = \Gamma = 1$, you should get the numbers as above. These values are also significantly higher than those in the paper, as this initial network is not at a local minimum.

3. Calculate the forces on the network, $\vec{F}_i = -\vec{\nabla}_i E = \begin{pmatrix} \frac{d}{dx_i} E \\ \frac{d}{dy_i} E \end{pmatrix}$, for each vertex i, producing a 48×2 area. This will require finishing the derivation of $\vec{F}_i^{\text{contractility}} = -\vec{\nabla} E^{\text{contractility}}$ analytically. It may also help to note for this part that there are two equivalent ways of writing the tension term:

$$E^{\text{tension}} = \sum_{\langle i,j\rangle} \Lambda \ell_{ij} = \sum_\alpha \frac{\Lambda}{2} L_\alpha$$

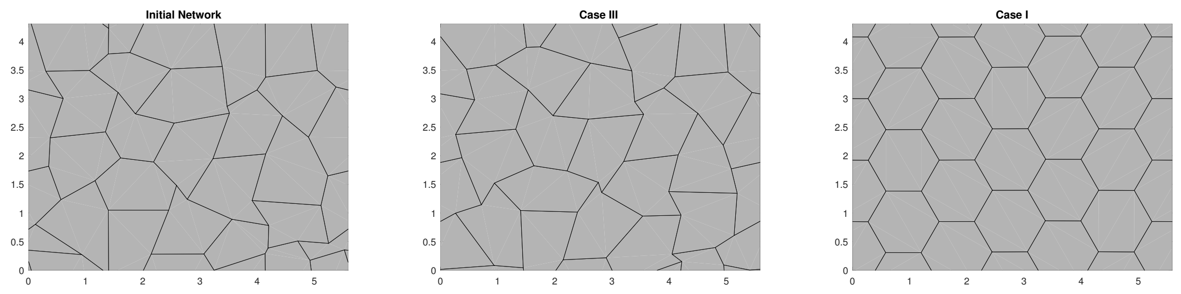

Fig. 2: Plots of the initial (left), soft (center), and hexagonal (right) networks. The soft and hexagonal networks correspond to Case III and Case I in Figure 1 of Farhadifar *et al.*.

4. Move the vertices a distance $\Delta\vec{r}_i = \varepsilon\vec{F}_i$, and calculate the energy again. Remember from calculus that $f(x+\Delta x) \approx f(x) + \frac{df}{dx}\Delta x$, and similarly,

$$\begin{aligned} E\left(\{\vec{r}_i + \Delta\vec{r}_i\}\right) &\approx E\left(\{\vec{r}_i\}\right) + \sum_i \left(\vec{\nabla}_i E_i\right)\cdot\left(\Delta\vec{r}_i\right) \\ &\approx E\left(\{\vec{r}_i\}\right) - \sum_i \vec{F}_i \cdot \left(\varepsilon\vec{F}_i\right) \\ &\approx E\left(\{\vec{r}_i\}\right) - \varepsilon\sum_i \left|\vec{F}_i\right|^2 \end{aligned}$$

Thus, if the positions are in array `pos`, the forces in array `fs`, and `E0` is the energy you calculated above, then you can do the following calculation:

```
pos1 = pos + epsilon .* fs;
E1 = [energy at pos1];
dE = (E0 - E1) ./ (epsilon .* sum(sum(fs .^2));
```

[Output] Print out the value you get for `dE` and the value you used for ε.

Then for a small epsilon (e.g. 10^{-6}), you should have `dE` $\approx 1 \pm \varepsilon$. If this is true, it tells you that the forces that you are calculating are actually the derivative of the energy you are calculating (at least, with the vertices in the given positions). If you do not get a value close to 1, it may help to alternate setting $\Lambda = 0$, $K = 0$, and/or $\Gamma = 0$, and use that to narrow down which part of the force and/or energy calculation is incorrect.

Note. You may want to put this minimization code in its own function, as we will be doing this frequently later. If you do this, make sure you have thought through what the input and output arguments should be, as this function will need access to the cell network, particle positions, boundary conditions, energy coefficients (A_0, K, Λ, and Γ), and a value for ε.

5. Using $\Lambda = -0.85KA_0^{\frac{3}{2}}$ and $\Gamma = 0.1KA_0$ as in the Case III from Farhadifar *et al.*, repeatedly use steepest descent to minimize the energy:

$$\vec{r}_i(t+\Delta t) = \vec{r}_i(t) + \varepsilon\vec{F}_i(t)$$

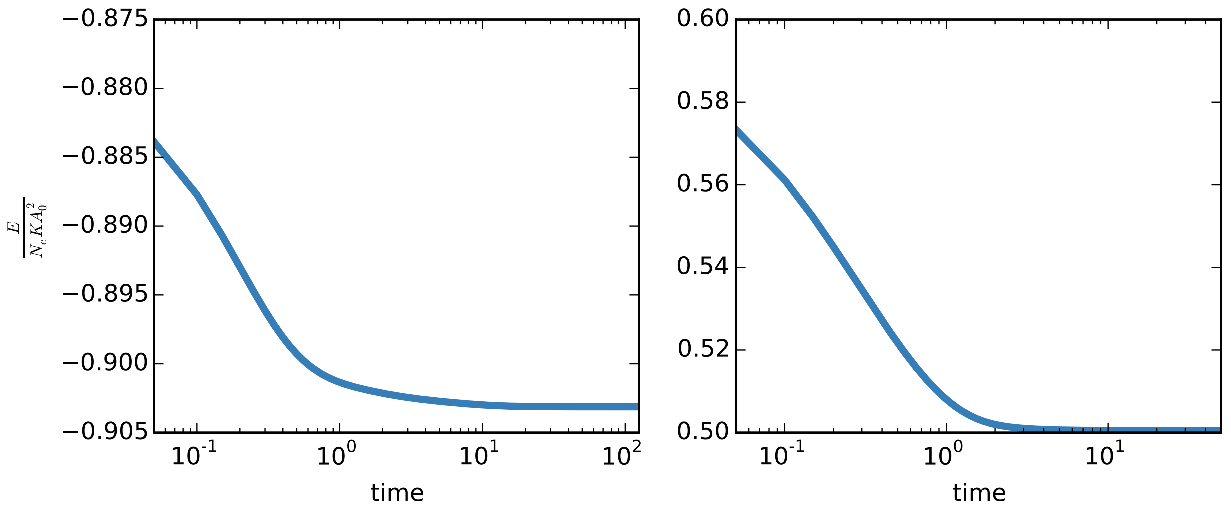

Fig. 3: Plots of energy per cell vs. time for a steepest descent calculation of random initial positions to a "soft" network (left) with parameters as in Case III, and for a transition from a "soft" network to a hexagonal network (right) as in Case I in Figure 1 of Farhadifar *et al.* 2007. Note that these values are significantly larger than those in the paper: while we have minimized the energy with respect to the vertex positions, we have not relaxed the box size.

The Δt here notes only that $\vec{r}_i\,(t+\Delta t)$ comes after $\vec{r}_i\,(t)$, as there is only a vague analogy with time with this method. You can use $\varepsilon = 0.05$; this value is calibrated to be small enough that $\vec{r}_i\,(t+\Delta t)$ is always "downhill" from $\vec{r}_i\,(t)$, but large enough that it shouldn't take more than a couple minutes to run the simulation, and hopefully only a few seconds. You should stop when forces are approximately balanced, $\sqrt{\sum_i \vec{F}_i^2} \leq 10^{-6}$, where that cutoff of 10^{-6} should give us plenty of precision for any calculations we are interested in. This should produce a "soft" network, as described in the paper. Note that at every step, the energy should decrease!

(a) **[Output]** Make a plot of the positions of the vertices and their connections, as in Figure 2.

(b) **[Output]** Plot $\frac{E(t)}{N_c K A_0^2}$ as a function of "time", where each "time" is a step down steepest descent.

6. Starting with the "soft" network above, use $\Lambda = 0.12 K A_0^{\frac{3}{2}}$ and $\Gamma = 0.04 K A_0$ as in Case I of Farhadifar *et al.*. This should produce a hexagonal network.

(a) **[Output]** Check that your network is hexagonal, and print out the energy you find per cell. Note that for hexagons of area A_0, the side length of each hexagon is $\ell = \frac{12^{\frac{1}{4}}}{3}\sqrt{A_0}$, and the perimeter $P = 6\ell = 2\left(12^{\frac{1}{4}}\right)\sqrt{A_0}$. Therefore, we should expect this hexagonal network to have elastic energy

$$\frac{E^{\text{elasticity}}}{N_c K A_0^2} = \frac{1}{N_c K A_0^2} \sum_\alpha \frac{K}{2} \left(A_\alpha - A_0\right)^2 = 0$$

- For the tension energy, we note that $\sum_{\langle i,j \rangle} \ell_{ij} = \frac{1}{2} \sum_\alpha L_\alpha$, so then

$$\frac{E^{\text{tension}}}{N_c K A_0^2} = \sum_{\langle i,j \rangle} \Lambda \ell_{ij} = \Lambda \sum_\alpha L_\alpha = \frac{\Lambda}{K A_0^{\frac{3}{2}}} 12^{\frac{1}{4}} \approx 1.86121 \frac{\Lambda}{K A_0^{\frac{3}{2}}}$$

- And then

$$\frac{E^{\text{contractility}}}{N_c K A_0^2} = \frac{1}{N_c K A_0^2} \sum_\alpha \frac{\Gamma}{2} L_\alpha^2 = \frac{\Gamma}{K A_0} 4\sqrt{3} \approx 6.9282 \frac{\Gamma}{K A_0}$$

If we then use the parameters $\Lambda = 0.12 K A_0^{\frac{3}{2}}, \Gamma = 0.04 K A_0$ as above, we get

$$\frac{E}{N_c K A_0^2} \approx 0.5004$$

(a) **[Output]** Make a plot of the cells, e.g. with `drawpolys.m`.

(b) **[Output]** Plot $\frac{E(t)}{N_c K A_0^2}$ as above.

7. Add a barostat to your code.

- To do this, we will now rewrite E as a function of both particle positions $\vec{r}_i$ and h, a scaling coefficient:

$$\begin{aligned} E &= E^{\text{elasticity}} + E^{\text{tension}} + E^{\text{contractility}} \\ &= \sum_\alpha \frac{K_\alpha}{2} \left(h^2 A_\alpha - A_0\right)^2 + h \sum_{\langle i,j \rangle} \Lambda_{ij} \ell_{ij} + h^2 \sum_\alpha \frac{\Gamma_\alpha}{2} L_\alpha^2 \end{aligned}$$

- Next, we will minimize this with respect to both particle positions and h (note that ε and ε_L do not need to be equal):

$$\begin{aligned} \vec{r}_i \left(t + \Delta t\right) &= \vec{r}_i \left(t\right) - \varepsilon \vec{\nabla}_i E \left(t\right) = \vec{r}_i \left(t\right) + \varepsilon \vec{F}_i \left(t\right) \\ h \left(t + \Delta t\right) &= h \left(t\right) - \varepsilon_L \frac{\partial E \left(t\right)}{\partial h} \end{aligned}$$

Now instead of changing h at every step, we can instead modify $\vec{r}_i$ and the boundary conditions L by modifying our positions and the box size at each timestep to

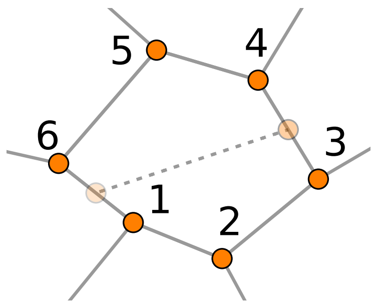

Fig. 4: An example cell, with all vertices labeled and a cell division shown. The division here would be with `split1=3` and `split2=6`.

keep up with the changes in h. This will then give us the equations[1]

$$\boxed{\begin{aligned} L\left(t+\Delta t\right) &= \left(1-\epsilon_L \left.\frac{\partial E}{\partial h}\right|_{h=1}\right) L\left(t\right) \\ \vec{r}_i\left(t+\Delta t\right) &= \left(1-\epsilon_L \left.\frac{\partial E}{\partial h}\right|_{h=1}\right) \left(\vec{r}_i\left(t\right)+\epsilon \vec{F}_i\right) \end{aligned}}$$

- If we then take that derivative, we get

$$\boxed{\left.\frac{\partial E}{\partial h}\right|_{h=1} = 2\sum_\alpha K_\alpha\left(A_\alpha^2 - A_\alpha A_0\right) + E^{\text{tension}} + 2E^{\text{contractility}}}$$

- You can then verify this as before:

$$E\left(t+\Delta t\right) \approx E\left(t\right) - \varepsilon\sum_i \left|\vec{F}_i\right|^2 - \varepsilon_L\left(\left.\frac{\partial E}{\partial h}\right|_{h=1}\right)^2$$

- You should now also be able to get hexagonal packings for Case I and Case II with energy values that match those given in Farhadifar *et al.*, with $\frac{E}{N_c K A_0^2} \approx 0.419$ for a hexagonal packing in Case I, and $\frac{E}{N_c K A_0^2} \approx 0.4528$ for a hexagonal packing in Case II (as shown in Fig. S2A of Farhadifar *et al.* Supplemental Material). Case III will not yield a hexagonal packing when minimized.

8. Implement cell division using the method discussed in class:

[1] Mathematically, we can do this more rigorously by writing E as a function of $\vec{s}_i$ and L, where $\vec{r}_i = L\vec{s}_i$. We then do steepest descent with $L\left(t+\Delta t\right) = L\left(t\right) + \varepsilon_L \frac{\partial E\left(\{\vec{s}_i\},L\right)}{\partial L}$ and $\vec{s}_i\left(t+\Delta t\right) = \vec{s}_i\left(t\right) + \varepsilon\vec{\nabla}_{\vec{s}_i} E\left(\{\vec{s}_i\},L\right)$. Algorithmically, we then calculate and store $L\left(t\right)$ and $\vec{r}_i\left(t\right)$, remembering that the energy E is a minimum of $L\left(t\right)$ and $\vec{s}_i\left(t\right) = \frac{\vec{r}_i(t)}{L(t)}$. This more rigorous framing of the problem is equivalent to the equations given above.

- I suggest (but do not require) that you write a function that takes the network and adds a new division to the network:

  ```
  function [newcells, newlocs] =
      celldiv(cells, locs, L, cellix, split1, split2)
  ```

  - `cells`, `locs`, and `L` are the cell network, vertex positions, and boundary conditions
  - `cellix` is the index in cells of the cell to split
  - `split1` and `split2` are the indices of the vertices in `cells{cellix}` *after* which the division should be placed.
- Next you will need to find the indices of all four relevant vertices, e.g. vertices 3, 4, 6, and 1 for the division shown in Fig. 4.
- Then you will need to find the midpoints of these cell edges. Don't forget boundary conditions! The easiest way to do this while respecting the boundary conditions is $\vec{m} = \vec{r}_1 + \frac{\vec{r}_{12}}{2}$, where $\vec{r}_{12}$ is the difference vector from point $\vec{r}_1$ to $\vec{r}_2$: `r12 = mod(r2 - r1 + L/2, L) - L/2`.
- Finally, you will need to update the `locs` array to include the two new points. You can just add them to the end of the array.
- Update the `cells` network to include the new vertices and new cells. Note that two new cells have been created, 1 has been erased (or replaced), and 2 additional neighboring cells (at the lower left and upper right of Fig. 4) have had a vertex added.

9. Run the full model for Cases I, II, and III.

- To run the full model, you will need to follow the following algorithm:
  - Randomly divide a cell in two using your cell division code.
  - Repeat:
    - ∗ Run steepest descent until either $\frac{1}{N_c}\sum_i \left|\vec{F}_i\right|^2 + \left(\frac{\partial E}{\partial h}\Big|_{h=1}\right)^2 < 10^{-4}$ or taking a step causes the energy to increase.
    - ∗ Use `transition.m` to take care of any transitions that may need to be permitted.
    - ∗ If the steepest descent has reached its threshold and no transitions can occur, return to cell division.
- Using $\varepsilon = 0.05$ and $\varepsilon_L = 0.001$, and a minimum distance $d = 0.05$ for the `lowdist` parameter for `transition.m` should give reasonable results. 168 divisions (3 generations) is acceptable for the purposes of this assignment, but may not yield results quite like those in Farhadifar *et al.*; if you have the time, run it for 360 divisions (4 generations), and you should be able to recapitulate their results. Note that for 360 divisions, you may need to lower ε_L to $\varepsilon_L = 0.0005$.
- I suggest putting the steepest descent and cell division code each in their own functions, to make it easier to write a "master" script that runs the above loop.

- see Table 1 for the coefficients.

10. **[Output]** Make a plot of the cells for each case after all 168 (or 360) divisions, e. g. with `drawpolys.m`, as in Fig. 2E, 2H, and 2K in Farhadifar *et al.*.

11. **[Output]** Calculate and plot a histogram of the number of sides per histogram, as in Fig. 2F, 2I, and 2L in Farhadifar *et al.*. Compare with the experimental data, given in Table S1 (in the Supplementary Material).

    - Code for the experimental data:

    ```
    exp_polys = [1, 6.78, 34.61, 38.28, 14.28, 2.17, 0.06] / 100;
    exp_polys_err = [0.77, 4.176, 4.06, 6.29, 3.36, 1.76, 0.24] / 100;
    errorbar(3:9, exp_polys, exp_polys_err, ...
        'o-', 'Color', [0.3,0.7,0.3], 'Linewidth', 3);
    ```

12. **[Output]** Calculate and plot $\frac{\langle A_n \rangle}{\langle A \rangle}$, the relative average area per polygon class, as in Fig. 2G, 2J, and 2M in Farhadifar *et al.*. Compare with the experimental data, given in Table S2 (in the Supplementary Material).

    - Code for the experimental data:

    ```
    exp_areas = [0.56, 0.82, 1.08, 1.36, 1.52];
    exp_areas_err = [0.02, 0.01, 0.01, 0.02, 0.05];
    errorbar(4:8, exp_areas, exp_areas_err,  ...
        'o-', 'Color', [0.3,0.7,0.3], 'Linewidth', 2);
    ```

13. **[Output]** Calculate the shape index $p_0 = \left\langle \frac{L_\alpha}{\sqrt{A_\alpha}} \right\rangle$ for each of the three cases. How does this compare with the critical value $p_0^* \approx 3.81$ given in Bi *et al.*? Based on this metric, would you expect the system to be fluid-like, solid-like, or somewhere in between (glassy)? Compare this with the target shape index $\frac{L_0}{\sqrt{A_0}}$, given in Table 1. Note that by completing the square, we can turn the tension and contractility terms into a single perimeter term proportional to $(L_\alpha - L_0)^2$, as shown in Bi *et al.*

$$\begin{aligned}
E^{\text{tension}} + E^{\text{contractility}} &= \sum_{\langle i,j \rangle} \Lambda \ell_{ij} + \sum_\alpha \frac{\Gamma}{2} L_\alpha^2 \\
&= \sum_\alpha \frac{\Lambda}{2} L_\alpha + \sum_\alpha \frac{\Gamma}{2} L_\alpha^2 \\
&= \sum_\alpha \frac{\Gamma}{2} L_\alpha \left(L_\alpha + \frac{\Lambda}{\Gamma} \right) \\
&= \sum_\alpha \left[\frac{\Gamma}{2} \left(L_\alpha - \frac{-\Lambda}{2\Gamma} \right)^2 - \frac{\Lambda^2}{8\Gamma} \right] \\
E' = \sum_\alpha \left[\frac{\Gamma}{2} (L_\alpha - L_0)^2 - \frac{\Lambda^2}{8\Gamma} \right] & \qquad\qquad L_0 = -\frac{\Lambda}{2\Gamma}
\end{aligned}$$

Case	$\frac{\Lambda}{KA_0^{\frac{3}{2}}}$	$\frac{\Gamma}{KA_0}$	$\frac{L_0}{\sqrt{A_0}}$
I	0.12	0.04	-1.5
II	0.0	0.1	0
III	-0.85	0.1	4.25

Tab. 1: The parameters used in the three main cases presented Farhadifar *et al.*.

Hint: See Figure 2A of Bi *et al.* and its discussion. Note that here and in Farhadifar *et al.*, the variables L_α and A_α denote the perimeter and area of cell α, and in Bi *et al.* they use p and a respectively.

Homework Submission

As before, make a zip (or tar) file with your code in `.m` format, all your graphs in in `.eps` format, and any textual answers in `.txt`, `.rtf`, or `.pdf` format. Upload to the classesv2 **Assignments section**.

References

R. Farhadifar, J.-C. Röper, B. Aigouy, S. Eaton, and F. Jülicher, Current Biology **17**, 2095 (2007).

D. Bi, X. Yang, M. C. Marchetti, and M. L. Manning, (2015a), arXiv:1509.06578 .

D. Bi, J. H. Lopez, J. M. Schwarz, and M. L. Manning, Nature Physics **11**, 1074 (2015b).