This read-me is associated with the XZ Mechanocellular Model. The model is written as a set of functions in python, which we run in a jupyter notebook. This read-me will include a list of each function, with all inputs and outputs as well as a brief mention of where things will be saved or requirements for locations to read in files if necessary. Two columned arrays are frequently used in this simulation. Unless otherwise specified, the first column corresponds to X positions and the second column corresponds to Z positions.

make_cells (seed = 2*21*1995, scatter = natural_cell_radius*0.1, compression = False)
***One requirement for this function is a variable saved in your workspace named 'centers', saved as an array of (n,2) where n is the number of cells you intend to simulate, and the rows of this array correspond to the location of the center of your starting cells.**
INPUTS:

- seed: float, optional – The seed variable represents the random seed used to initialize a random state for the cells to begin the simulation. The default is 2*21*1995.
- scatter: float, optional – The scatter variable represents the maximum displacement the initial cell nodes will have from perfectly circular. The default is 10% of the cell's radius.
- compression: list, optional – The compression variable allows the user to specify a limited amount of substrate if desired. Input a 2 element list, where the first element is the left X-cutoff, and the second is the right X-cutoff. The default is a substrate that extends out 2 times the minimum and maximum displacements from 0 to either side. The intention is that the user will place cells roughly symmetrically about 0, but if you specify your own compressions this is not important.

OUTPUTS:

- floor: numpy array – Floor is an (mx2) array where m is the number of substrate points available for cells to interact with. It is a stiff substrate which will not be changed in time with the cells.
- list_of_cells: list – This list will be of length n as specified in the 'centers' variable by the user. Each element of the list will be a (p,2) numpy array, where p is the number of nodes in the starting cell. As is, p is set to 40.

plot_cells (list_of_cells, connections = False, CCid = 0.16, CSid = 0.1, save_name = False)
This function will make a 2D plot of whatever list of cells you input. This is looped through to make a movie and can be used to check any time points of simulation.
INPUTS:

- list_of_cells: list – This list is of length n where n is the number of cells in the simulation, with each element being an (…x2) array corresponding to the nodes (which can vary in number between cells) of the cell.
- connections: bool, optional – If true, the plot will include blue lines for cell-substrate connections and green lines for cell-cell connections.

- CCid: float, optional – This scalar sets the interaction distance for cell-cell connections in arbitrary units. The default value is 0.16.
- CSid: float, optional – This scalar sets the interaction distance for cell-substrate connections in arbitrary units. The default value is 0.1.
- save_name: string, optional – If you want to save the image as a pdf. The default value of False results in not saving the image. The file will be saved in the same directory the main code is saved in.

OUTPUTS:

- The only output of this function is a plot which may or may not be saved.

make_movie_images (simulation_timepoints, save_name, frame_freq = 1, connections = False)

This function will generate a folder full of PDFs in the same directory as the file used to run this function.

INPUTS:

- simulation_timepoints: list – This is a list of lists. Each element in the main list is a list of cells of length n where n is the number of cells in the simulation, with each element being an (…x2) array corresponding to the nodes (which can vary in number between cells) of the cell.
- save_name: string – The save name is required and will create a folder in the same directory as the file used to run this function. The name of the folder will be save_name. Each file in this folder will be a pdf saved as the save_name with a suffix corresponding to the saved timepoint.
- frame_freq: int, optional – A rate which will determine how frequently timepoints will be saved as PDFs. Default is set to every 5^{th} timepoint.
- connections: bool, optional – If true, the plot will include blue lines for cell-substrate connections and green lines for cell-cell connections.

OUTPUTS:

- This function will create a series of plots within the jupyter notebook, all saved within a folder as well.

area (cell)

This function will calculate the area of a cell input as a 2D numpy array.

INPUTS:

- cell: array – The input will be a (…x2) array corresponding to the nodes (which can vary in number between cells) of the cell.

OUTPUTS:

- area: scalar – The area of the input cell as calculated by the shoelace formula.

distance_matrix (array1, array2)

This will calculate all distances between the points in two 2D arrays, using each row as an XZ coordinate.

INPUTS:

- array1: array (mx2) – The input will be an array corresponding to m points in 2D space.
- array2: array (nx2) – The input will be an array corresponding to n points in 2D space.

OUTPUTS:

- distance_matrix: array (mxn) – A matrix whose (p,q) element corresponds to the distance between the p^{th} point of array1 and the q^{th} point of array2.

cell_substrate_interactions (list_of_cells, CSid)

This will determine all interactions between all cells in the list of cells and the predefined floor variable (see make_cells).

INPUTS:

- list_of_cells: list – This list is of length n where n is the number of cells in the simulation, with each element being an (…x2) array corresponding to the nodes (which can vary in number between cells) of the cell.
- CSid: float – This scalar sets the interaction distance for cell-substrate connections in arbitrary units.

OUTPUTS:

- array_of_cell_substrate_pairs: array – This array is an (nx1) array with elements ranging from 0 to len(list_lf_cells)-1. The j^{th} element of this array is the location of the cell that is interacting in the j^{th} pair of the second output of this function.
- array_of_node_pairs: array – This array is an (nx2) array which contains the position of the floor node (1^{st} column) and the position of a cell node (2^{nd} column). The j^{th} row of this array will be one interaction between the cell as identified using the first output of this function and the floor.

cell_cell_interactions (list_of_cells, CCid)

This will determine all interactions between all pairs of cells in the list of cells.

INPUTS:

- list_of_cells: list – This list is of length n where n is the number of cells in the simulation, with each element being an (…x2) array corresponding to the nodes (which can vary in number between cells) of the cell.
- CCid: float – This scalar sets the interaction distance for cell-cell connections in arbitrary units.

OUTPUTS:

- array_of_cell_cell_pairs: array – This array is an (nx2) array with elements ranging from 0 to len(list_lf_cells)-1. The j^{th} row of this array contains the location of the cells that are interacting in the j^{th} pair of the second output of this function. The individual interactions (i.e. [0,2] and [2,0]) are repeated in this array for ease of computation.
- array_of_node_pairs: array – This array is an (nx2) array which contains the position of the first cell's node (1^{st} column) and the position of the second cell's node (2^{nd} column).

The j^{th} row of this array will be one interaction between two cells as identified using the first output of this function. The nodes in the 1^{st} column of this function correspond to the cell in the 1^{st} column of the first output of this function. The nodes in the 2^{nd} column of this function correspond to the cell in the 2^{nd} column of the first output of this function.

internal_force_on_cell (cell, kA, kL)
This will calculate the forces due to the area and perimeter constraints on each node of a cell.

INPUTS:

- cell: array (nx2) – The input will be an array of nodes for a single cell.
- kA: float – The input will be a scalar value corresponding to strength of the area constraint in arbitrary units.
- kL: float – The input will be a scalar value corresponding to strength of the perimeter constraint in arbitrary units.

OUTPUTS:

- force_on_cell: array (nx2) – An array the same shape as cell containing the force on each node in the X and Z directions.

spreading (cell, cell_substrate_interactions, spreading_strength)
This will calculate the forces due to active spreading on each node of a cell. Will have either 0,1, or 2 non-zero rows.

INPUTS:

- cell: array (mx2) – The input will be an array of nodes for a single cell.
- cell_substrate_interactions: array (nx1) – This array will have the length of the number of interactions cell has with the substrate. This is a subset of the 2^{nd} column of array_of_node_pairs created in cell_substrate_interactions.
- spreading strength: float – The input will be a scalar value corresponding to strength of the cell spreading in arbitrary units.

OUTPUTS:

- force_on_cell: array (nx2) – An array the same shape as cell containing the force on each node in the X and Z directions.
- left_point: int – The location of the node adjacent to (to the left of) the leftmost cell-substrate interaction point.
- right_point: int – The location of the node adjacent to (to the right of) the rightmost cell-substrate interaction point.

total_force_on_cells (list_of_cells, kA, kL, gCS, gCC, CSid, CCid, spreading strength)
This will calculate the total force on the cells at a given timepoint in the simulation.

INPUTS:

- list_of_cells: list – This list is of length n where n is the number of cells in the simulation, with each element being an (…x2) array corresponding to the nodes (which can vary in number between cells) of the cell.
- kA: float – The input will be a scalar value corresponding to strength of the area constraint in arbitrary units.
- kL: float – The input will be a scalar value corresponding to strength of the perimeter constraint in arbitrary units.
- gCS: float – This input will be a scalar value corresponding to the strength of the cell-substrate interaction force in arbitrary units.
- gCC: float – This input will be a scalar value corresponding to the strength of the cell-cell interaction force in arbitrary units.
- CSid: float – This scalar sets the interaction distance for cell-substrate connections in arbitrary units.
- CCid: float – This scalar sets the interaction distance for cell-cell connections in arbitrary units.
- spreading strength: float – The input will be a scalar value corresponding to strength of the cell spreading in arbitrary units.

OUTPUTS:

- total_forces: list – This list will have the same length as list_of_cells and its elements will have the same shape as the elements in list_of_cells. Each element will contain the force on each node of the corresponding cell in list_of_cells in the X and Z directions.

cell_remodeling (list_of_cells, kA, kL)

This applies the viscous nature of the cell cortex to the cells in list_of_cells. Any space which is greater than twice the natural node spacing is bisected, and any space which is less than half of the natural node spacing causes one node to be removed.

INPUTS:

- list_of_cells: list – This list is of length n where n is the number of cells in the simulation, with each element being an (…x2) array corresponding to the nodes (which can vary in number between cells) of the cell.

OUTPUTS:

- remodeled_list: list – A list the same length as list_of_cells, with the remodeling condition applied to each cell.

move_points (list_of_cells, kA, kL, gCS, gCC, CSid, CCid, spreading strength, time_step, viscosity)

This will take the cells at one timepoint and update them according to the total force and time_step to get the positions of each node at the next timepoint.

INPUTS:

- list_of_cells: list – This list is of length n where n is the number of cells in the simulation, with each element being an (…x2) array corresponding to the nodes (which can vary in number between cells) of the cell.

- kA: float – The input will be a scalar value corresponding to strength of the area constraint in arbitrary units.
- kL: float – The input will be a scalar value corresponding to strength of the perimeter constraint in arbitrary units.
- gCS: float – This input will be a scalar value corresponding to the strength of the cell-substrate interaction force in arbitrary units.
- gCC: float – This input will be a scalar value corresponding to the strength of the cell-cell interaction force in arbitrary units.
- CSid: float – This scalar sets the interaction distance for cell-substrate connections in arbitrary units.
- CCid: float – This scalar sets the interaction distance for cell-cell connections in arbitrary units.
- spreading strength: float – The input will be a scalar value corresponding to strength of the cell spreading in arbitrary units.
- time_step: float – This input will control how much 'time' passes for the updated iteration of the simulation in arbitrary units.
- viscosity: float – This input will control how much each node will move at a given force in arbitrary units. Increasing viscosity or decreasing time_step have the same effect on the model.

OUTPUTS:

- new_cell_positions: list – This list will have the same length as list_of_cells Each element will contain the new location, shape, and size of cells in list_of_cells.

time_evolve (list_of_cells, kA=1, kL=0.0005, gCS=0.1, gCC=0.001, CSid=0.1, CCid=0.16, spreading strength=0.008, time_step=1, viscosity=1, iterations=15000)
This will take run the move_points function on the list_of_cells a number of times equal to the number of iterations and create a final list with all timepoints appended sequentially.

INPUTS:

- list_of_cells: list – This list is of length n where n is the number of cells in the simulation, with each element being an (…x2) array corresponding to the nodes (which can vary in number between cells) of the cell.
- kA: float, optional – The input will be a scalar value corresponding to strength of the area constraint in arbitrary units. Default value of 1.
- kL: float, optional – The input will be a scalar value corresponding to strength of the perimeter constraint in arbitrary units. Default value of 0.0005.
- gCS: float, optional – This input will be a scalar value corresponding to the strength of the cell-substrate interaction force in arbitrary units. Default value of 0.1.
- gCC: float, optional – This input will be a scalar value corresponding to the strength of the cell-cell interaction force in arbitrary units. Default value of 0.001.
- CSid: float, optional – This scalar sets the interaction distance for cell-substrate connections in arbitrary units. Default value of 0.1.
- CCid: float, optional – This scalar sets the interaction distance for cell-cell connections in arbitrary units. Default value of 0.16.

- spreading strength: float, optional – The input will be a scalar value corresponding to strength of the cell spreading in arbitrary units. Default value of 0.008.
- time_step: float, optional – This input will control how much ‘time’ passes for the updated iteration of the simulation in arbitrary units. Default value of 1.
- viscosity: float, optional – This input will control how much each node will move at a given force in arbitrary units. Increasing viscosity or decreasing time_step have the same effect on the model. Default value of 1.
- iterations: int, optional – This input controls how long to run the simulation. With a time_step of 1, the value of iterations will equal the number of times move_points is applied. Default value of 15000.

OUTPUTS:

- simulation_timepoints: list – This is a list of lists. Each element in the main list is a list of cells of length n where n is the number of cells in the simulation, with each element being an (…x2) array corresponding to the nodes (which can vary in number between cells) of the cell.