6.2.3 Implementing the ScEM in Chaste

To simulate the ScEM we extend the implementation in Chaste for simulating the overlapping spheres model described in Chapter 3. This is an example of the benefit of Chaste's design as a modular library: much of the functionality including setup, output and the parallel algorithm, as well as the necessary data structures are already implemented and well tested. Consequently, implementing the ScEM is reduced to defining a new force class `SemForce`, children of the mesh and population classes for overlapping spheres (`SemMesh` and `NodeBasedCellPopulationWithSemId`) and two helper classes: a class (`SemParameterScaler`) to scale parameters η_i, u_0 and r_{eq}, and a class (`SemCellsGenerator`) to enable the construction of a population of ScEM cells from a single archive of a cell's initial element locations. The archives for ScEM cells are created with a standalone Python script using an algorithm to place elements close to equilibrium. All classes in Chaste are templated over spatial dimensions, making the transition between 2D and 3D models straightforward.

Creating ScEM cells

To generate a 2D ScEM cell consisting of N elements and of area πR_{cell}^2 inside a region Ω (also of area πR_{cell}^2) we proceed as follows. A location $\mathbf{r}_{\alpha_0} \in \Omega$ is chosen for element α_0. A possible location $\tilde{\mathbf{r}}_{\alpha_1}$ for element α_1 is selected uniformly at random in Ω and accepted if $||\tilde{\mathbf{r}}_{\alpha_1} - \mathbf{r}_{\alpha_0}|| \geq 0.75 r_{\text{eq}}$. This process is iterated, accepting $\tilde{\mathbf{r}}_{\alpha_k}$ only if $||\tilde{\mathbf{r}}_{\alpha_k} - \mathbf{r}_{\alpha_j}|| \geq 0.75 r_{\text{eq}}$ for all $j < k$, until all elements are assigned a location. This is implemented in a standalone Python [Python, 2013] script.

Force class

The `SemForce` class has member variables `mSpringStiffness`, which corresponds to k_0 in equation (6.4), `mNuclearSpringFactor` corresponding to K_{nuc} in equation (6.6) and a map from cell indices i to relative cell stiffnesses K_i in equation (6.7). As a member of the mechanics class hierarchy in Chaste, `SemForce` has a method `AddForceContribution()`, which takes a reference to the cell population as an input argument. From the cell population it obtains a list of neighbouring elements and the force between pairs of elements is evaluated using the appropriate equation from equations (6.6)-(6.7), depending on the element types.

Parameter scaling

Since scaling of the parameters depends on the number of elements per cell, and is used in multiple classes throughout the simulation, we implement a helper method using the singleton design pattern [Gamma et al., 1994]. This ensures that all parameters are scaled consistently across all classes, and brings all scaling into a single code source file, reducing the likelihood of introducing bugs. The class contains a map from cell indices to the number of elements contained in the cell, as well as members for the constants ρ and λ. Two methods accept a cell index and an unscaled damping constant η_i or unscaled spring constant k_0 and return the scaled value of η_i and u_0 respectively. A further method returns the equilibrium distance r_{eq}.

Mesh and cell population classes

The mesh class `SemMesh` inherits from the `NodesOnlyMesh` and differs only by correctly setting the cell and nucleus labels of elements. The cell population child class overwrites output methods to write ScEM cell indices to file in addition to location data. The method for updating element locations is modified to take into account the scaled damping constant η_i in the integration scheme.

Generating cell populations

The class `SemCellsGenerator` constructs initial conditions by accepting the location of a cell archive file generated by the Python scripts and a translation co-ordinate for each cell. A copy of each cell from the archive is then created with its centre of mass translated appropriately. This class also defines the initial nuclear region of a cell as a sphere (or circle) or radius R_{nuc}, centered at $\mathbf{r}_{\text{nuc}}$ in the untranslated cell.

Gaussian noise

We implement the Gaussian noise in equation (6.1) in the force class `SemDiffusionForce` as a random perturbation to the force on each element at each time step. The perturbation at time t to the force on an element α_i is given by $\boldsymbol{\xi}_{\alpha_i}(t) = (\xi^m_{\alpha_i}(t))$ where for all $m = 1, 2$, $\xi^m_{\alpha_i}(t) \sim N(0, \eta_i\sqrt{2D/\Delta t})$, where η_i is the viscosity, Δt the integration time step and D is a diffusion parameter.

6.3 Results

We carry out three compression experiments. First, we consider the dynamics of cell nuclei in a layer of model cells under compression for different values of the nuclear stiffness K_{nuc}. Second, we consider the effect of the different size of Paneth and stem cells observed in the crypt on the buckling of a layer of cells. Finally, we consider whether altering the relative stiffness of Paneth and stem cell cytoplasm significantly alters these results. The common parameter values for each experiment are given in Table 6.1.

In each simulation we consider two metrics to characterise the level of buckling seen within the layer at time t. First, for the buckled state of the whole cell layer, we consider the standard deviation of the height above the basal boundary condition of the centre of mass of each cell, given by

$$\sigma_{\text{cell}} = \sqrt{\frac{\sum_{i=0}^{N_{\text{cell}}-1} (\bar{y}_i - \bar{y})^2}{N_{\text{cell}}}},$$

where $\bar{y}_i$ is the mean height of elements in cell i above the basal boundary condition at time t and $\bar{y}$ is the mean height of all elements above the basal boundary condition at time t. Second, we consider the standard deviation of the height above the basal boundary condition of the centre of mass of each cell nuclei, given by