## Computational Framework

We have developed a dedicated high-performance computational framework, HiPerLiFE (High Performance Library for Finite Elements), which serves as the core numerical engine for the simulations presented in this study. HiPerLiFE is a general-purpose parallel finite element library designed with a particular focus on problems in cell and tissue mechanobiology, where the interplay of nonlinear elasticity, active stresses, fluid–structure interaction, and surface geometry is essential. The library is openly distributed to the community and is available online at https://hiperlife.gitlab.io/hiperlife/.

At its foundation, HiPerLiFE is written in modern `C++`, adopts the MPI (Message Passing Interface) paradigm for distributed-memory parallelism, and is built upon multiple modules from the Trilinos Project, which provides robust solvers, preconditioners, and tools for scalable scientific computing. This choice of design ensures the three fundamental goals of the library:

- **Parallel scalability**: The code can run seamlessly from individual workstations to medium-size clusters and large-scale high-performance computing facilities.
- **Geometric flexibility**: The framework manages unstructured finite element meshes, which are essential to represent the complex and curved geometries typical of cellular and tissue systems, including membranes, cortices, and epithelial monolayers.
- **High-order accuracy and multiphysics support**: Arbitrary high-order basis functions can be employed to capture curvature and surface effects, and the framework accommodates multiphysics coupling (e.g., elasticity, active stresses, fluid interaction), which is characteristic of mechanobiological problems.

Within our group, HiPerLiFE is currently used to address diverse problems in biomechanics and mechanobiology. It has been extensively tested across a wide range of platforms, from local laptops running Linux distributions to national-level supercomputing clusters. This versatility makes it a robust tool both for method development and for large-scale scientific applications.

## Code Organization and Project Setup

For the present work, we implemented the problem-specific code within the HiPerLiFE ecosystem by creating a top-level project folder named `wrinkling_epithelial_continuum`. This project folder contains a global `CMakeLists.txt` file, which controls the compilation, and one or more subprojects (applications), each corresponding to a specific simulation setup. For example, the application `inflate_hold_deflate` implements the model of interest here and resides in its own folder with a local `CMakeLists.txt`. The code used in this study is openly available at https://github.com/pradeep927/mechanics_of_epithelial_domes.

The build system relies on CMake to configure and manage the compilation. Two additional files are kept in the project root: `cmake.project.ubuntu.20.04.sh`, a shell script to automate the configuration process on Ubuntu systems; and `userConfig.cmake`, a configuration file where paths to HiPerLiFE, Trilinos, and other dependencies are specified. The typical installation procedure is as follows: Install HiPerLiFE following the instructions at https://hiperlife.gitlab.io/hiperlife/Installation/index.html. For our simulations, the libraries were installed on a workstation running Ubuntu 22.04.

Place the two files inside the top-level project folder `wrinkling_epithelial_continuum`. From the terminal, make the script executable and launch the build configuration:

```
chmod 777 cmake.project.ubuntu.20.04.sh
./cmake.project.ubuntu.20.04.sh
```

Enter the build directory and compile:

```
cd build
make -j4 install
```

This process generates the executable binary at:

`/home/ubuntu/wrinkling_epithelial_continuum/source_compiled/bin/hlinflate_hold_deflate`

## Running Simulations

To execute a simulation, we create a dedicated folder `run_simulation` that contains the mesh files and the configuration files. Meshes are provided in VTK format and typically correspond to epithelial footprints with prescribed geometries. For example, a mesh named `aspect_26` corresponds to a circular domain with aspect ratio 26. Boundary conditions are imposed by carefully fixing the exterior nodes of the domain using elastic springs, which mimic physical anchoring constraints.

The simulation parameters, including references to the mesh files, are specified in a configuration file named `config.cfg`. This file allows the user to set model parameters, time-stepping controls, solver tolerances, and material constants.

A simulation is launched in parallel using MPI as:

```
mpirun -n 4 /home/ubuntu/wrinkling_epithelial_continuum/
source_compiled/bin/hlinflate_hold_deflate config.cfg
```

Here, the option `-n 4` specifies the number of processors. This value can be adjusted according to the available computational resources and the problem size.

## Postprocessing and Visualization

During execution, the code generates output files in VTK format (e.g., `sol_dis_.vtk`), which store the displacement, stress, and other field variables at different time steps. These files can be directly visualized and post-processed using ParaView. To streamline postprocessing, we provide a ParaView state file (`paraview_state.pvsm`), which loads the output files, applies predefined visualization settings, and enables rapid analysis of simulation results, including deformation fields, wrinkling patterns, and stress distributions.

## Code Availiability

The code is openly available at https://github.com/pradeep927/mechanics_of_epithelial_domes.