Multiscale kinematic growth coupled with mechanosensitive systems biology in open-source software

Authors and developers:
Steven A. LaBelle, Ph. D.,
Mohammadreza Soltany Sadrabadi, Ph. D.,
Amirhossein Arzani, Ph. D.

1-Introduction:

This repository contains the input files required to reproduce nonlinear transient structural mechanics problems presented in LaBelle et al. 2024. Input files are provided for reproduction in Python (FEniCS) or Finite Elements for Biomechanics (FEBio). These simulations couple the kinematic growth and remodeling (G&R) of a biological tissue/structure with systems biology and cell signaling pathways. At the cell scale, we implement a system of ordinary differential equations (ODEs) or partial differential equations (PDEs) representing the reactions between the biochemicals causing the growth. We implemented the weak form of the governing continuum mechanics equations using finite element analysis representing the G&R at the tissue level. Three test subjects (cube, aneurysm, and aortic valve) are implemented in the package.

2- FEniCS Installation:

FEniCS interfaces with MPI and linear algebra backends like PETSc to solve finite element problems in Python. To work with this package, a user should know the basic knowledge of how to solve PDEs through FEniCS and basic programming skills in Python. An excellent introduction to FEniCS can be found at https://fenicsproject.org/tutorial/

The input files provided are compatible with the legacy version of DOLFIN/FEniCS, not the current release versions of DOLFINx/FEniCSx. To run the software, users need to install FEniCS, an open-source finite element software written in Python: the best reference to install FEniCS is: https://fenicsproject.org/download/

The simulations were originally generated using a containerized version of DOLFIN/FEniCS via docker. The container contains a build of the software with most of the necessary dependencies already installed. To install via Docker, the following command may be used (note, we use stable:latest not stable:current):

```
$ docker run -ti quay.io/fenicsproject/stable:latest
```

A local repository can be added as a volume to the container through:

```
$ docker run -ti -p 127.0.0.1:8000:8000 -v $(pwd):/home/fenics/shared -w /home/fenics/shared quay.io/fenicsproject/stable:latest
```

Alternatively, the package can be installed via conda. We tested our code using the miniforge3 distribution of conda (https://github.com/conda-forge/miniforge). The project is created and activated with:

```
$ conda create -n fenicsproject -c conda-forge fenics=2019.2
$ source activate fenicsproject
```

- The solver can be installed simply by cloning the GitHub repository to your own computer:
  ```
  $ git clone https://github.com/amir-cardiolab/Multiscale_mechanobiology_GR.git
  $ cd code
  ```
- Alternatively, you can download software from the link below:
  ```
  $ https://github.com/amir-cardiolab/Multiscale_mechanobiology_GR.git
  $ unzip code.zip
  $ cd code
  ```

3-Files and folders:

The software contains folders for the cube and aneurysm models. All of the codes are implemented in Python.

The solver can be installed simply by cloning the GitHub repository to your own computer:

```
$ git clone https://github.com/amir-cardiolab/Multiscale_mechanobiology_GR.git
$ cd code
```

4- Running models:

Once in your python environment, models may be run from the command line with

```
$ python3.6 [input_file.py] > [log_file.txt]
```

5- Post-processing:

You can see the results in the open-source software ParaView. To download the software, you can use the link below:

https://www.paraview.org/download/

To see the displacement and the growth for each time span, you need to use the “warp by vector” option in Filters->alphabetical->warp by vector.