Julia vs Python for a cell simulation

Hampus Fröjdholm & Carmen Lee
hampus.frojdholm@gmail.com, mailtocarmenlee@gmail.com
UPPSALA UNIVERSITET

1. Introduction

Python has gained considerable popularity since its introduction in 1991 and has a large ecosystem for scientific computing.

Julia, a relatively young language designed for scientific programming, has also gained traction in recent years.

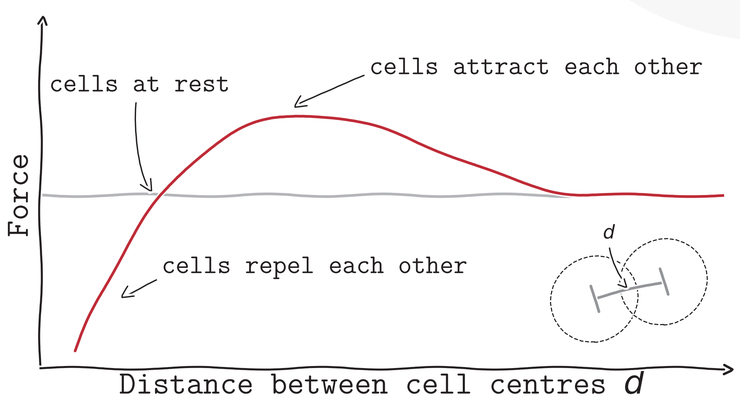
The performance difference between the two is measured using a centre-based model for tissue mechanics simulation.

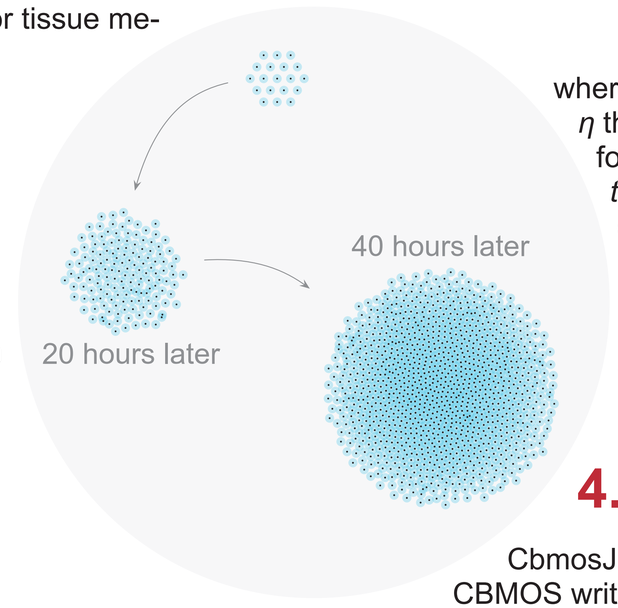
2. Cell interaction

Cells are modelled as spheres.

Forces are determined by the distance between sphere centres and a predetermined force law.

Cells divide according to a normally distributed cycle (arbitrarily set to N(6, 1)).





3. Equations of motion

Under the condition of low Reynolds numbers, the velocity of an object is proportional to the sum of total force.

For a cell *i* its motion is governed by the following equation:

$$\eta \frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t} = \sum_{j \in \mathcal{N}_i(t)} \mathbf{F}_{ij}(t),$$

where $\mathbf{r}_i$ is the position of the cell centre, η the viscosity of the fluid, $\mathbf{F}_{ij}(t)$ the force exerted on cell *i* by cell *j* at time *t*, and $N_i(t)$ the set of neighbours of cell *i*.

This yields a first order ODE system for the motion of the cells, which is solved with numerical integration.

4. CbmosJulia

CbmosJulia is based on the Python package CBMOS written by researchers at Uppsala University. CBMOS is a tool for centre-based cell simulation.

Users can set the initial cell sheet configuration, choose force law and the type of solver. Other parameters such as step size and arguments for the force law function are also up to the user.

The graphic in the centre is made with data generated by CbmosJulia.

5. Results

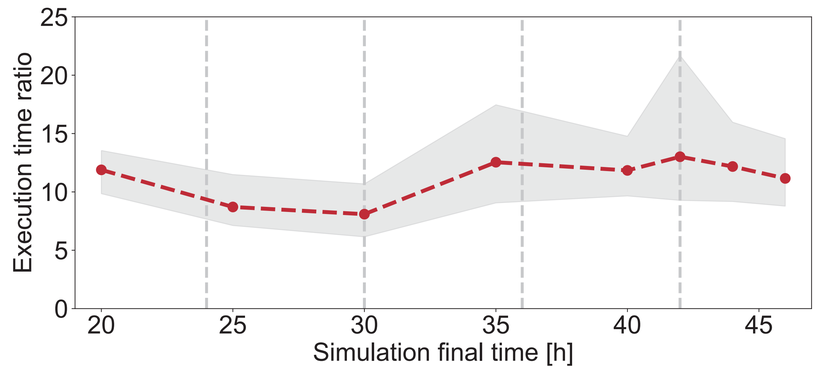


Figure 1: Speedup of CbmosJulia compared to CBMOS. The speedup of CbmosJulia compared with CBMOS is consistently around 10X. The ratio was calculated using the mean execution time of 10 samples at each final time. The grey regions represent the maximum and minimum ratio measured at each final time. The vertical grey dashed lines indicate mean division cycles.

6. Conclusions on Julia

+ Powerful programming paradigm for modelling mathematical problems.
+ Can achieve “C-like” performance.
+ Comprehensive standard library for scientific computing.
+ High-quality numerical packages.
+ Good official language documentation.
− Slow start-up time and JIT-compilation takes time.
− Need to unlearn vectorisation.
− Packages are not as mature, e.g. for plotting.
− Editor support is lacking.