Universidad Zaragoza 1542

XI REUNIÓN DEL CAPÍTULO ESPAÑOL DE LA SOCIEDAD EUROPEA DE BIOMECÁNICA

PhysiCell: An hands-on introduction to cell-based modelling

24 de octubre de 2022

Contents

Installing PhysiCell

Installing PhysiCell

PhysiCell is available through GitHub and SourceForge.

GitHub

https://github.com/MathCancer/PhysiCell

SOURCEFORGE

https://sourceforge.net/projects/physicell/

PhysiCell works on **Linux**, **Mac** and **Windows**.
However, PhysiCell is written in **C++** and the code must be compiled before running it.
Thus, it is not always an easy task to set up the right **development environment**.

MSYS2 + make
Compile the C++ code

core (required)

ImageMagick
Processing results

Python
Processing results + model builder

recommended

MATLAB/ParaView/...
Processing results

optional

Installing PhysiCell

In this course, we are going to use an online environment that already includes all the required dependencies.

Gitpod

docker

PhysiCell ubuntu

For more information on how to install PhysiCell in your machine, see:

- **PhysiCell Workshop 2022: Setting up PhysiCell (Windows/Mac)** (available on GitHub)
- **PhysiCell Roadmap:** https://iggoncalves.github.io/physicell-roadmap/

Installing PhysiCell

💻 Documentation:

“Running PhysiCell online”

Image source: https://dribbble.com/shots/8630894-Programmer-cat

Introduction to PhysiCell

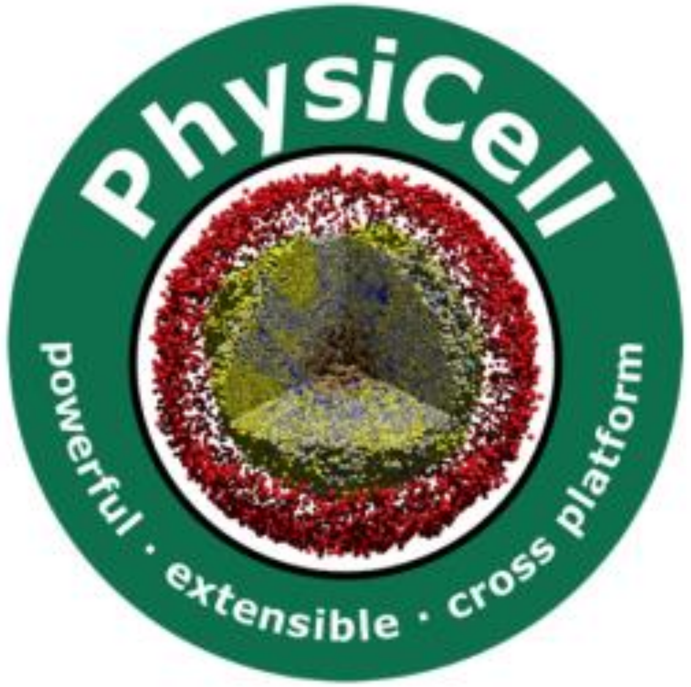
Introduction

PhysiCell is an **off-lattice**, **centre-based** modelling platform which **aims to simulate millions of cells** on desktops.

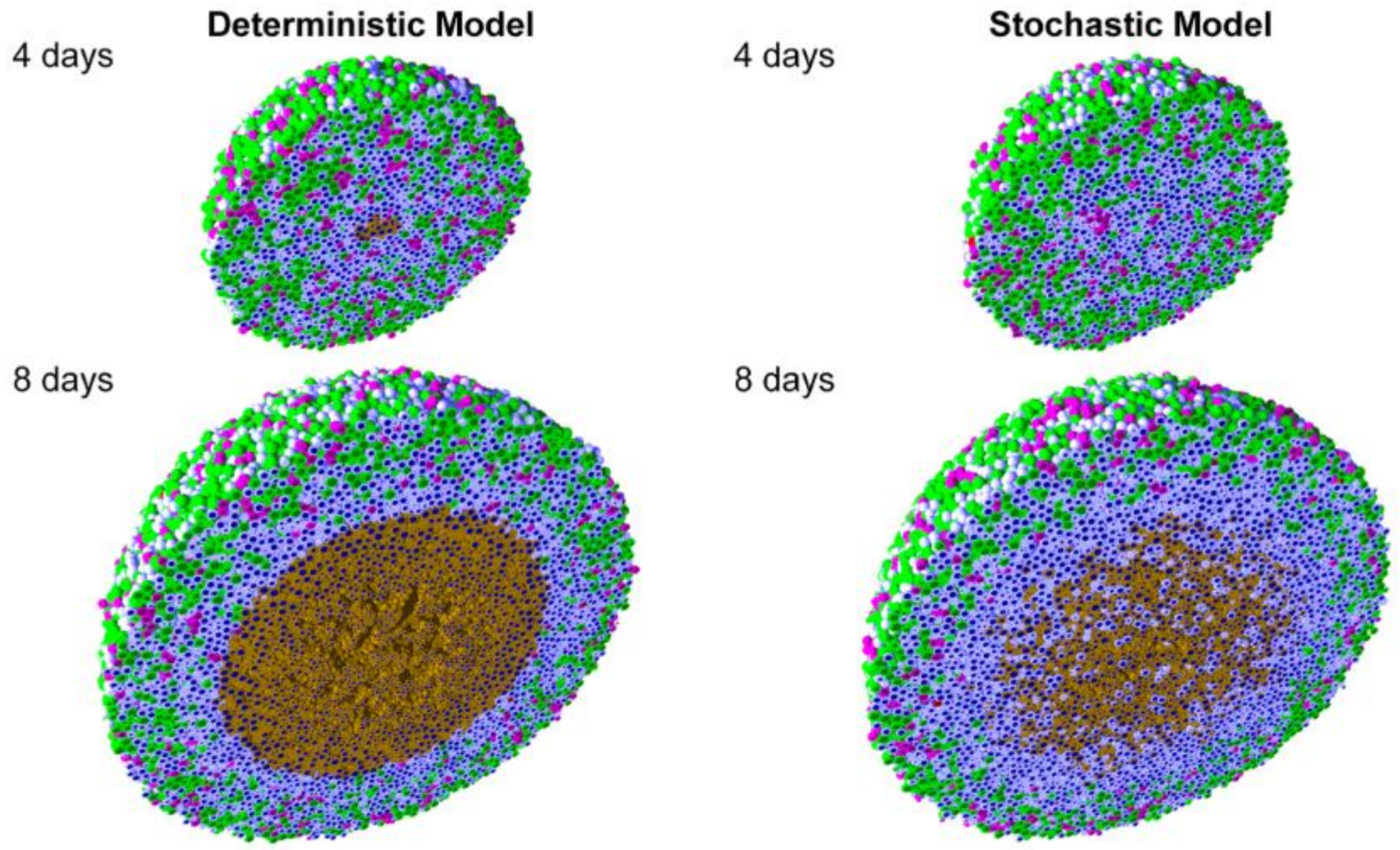
Cell agents are able to:

- Proliferate (divide);
- Die:
  - apoptosis – naturally;
  - necrosis – in response to harsh conditions;
- Interact with other cells:
  - Adhesion;
  - Repulsion;
  - Create cell-cell adhesions;
- Generate locomotive forces:
  - Random walk;
  - Chemotaxis;
- Secrete and consume substances;

Most of the cell rules can follow **deterministic** or **stochastic** models, which can be helpful to capture tissue heterogeneity.



Introduction



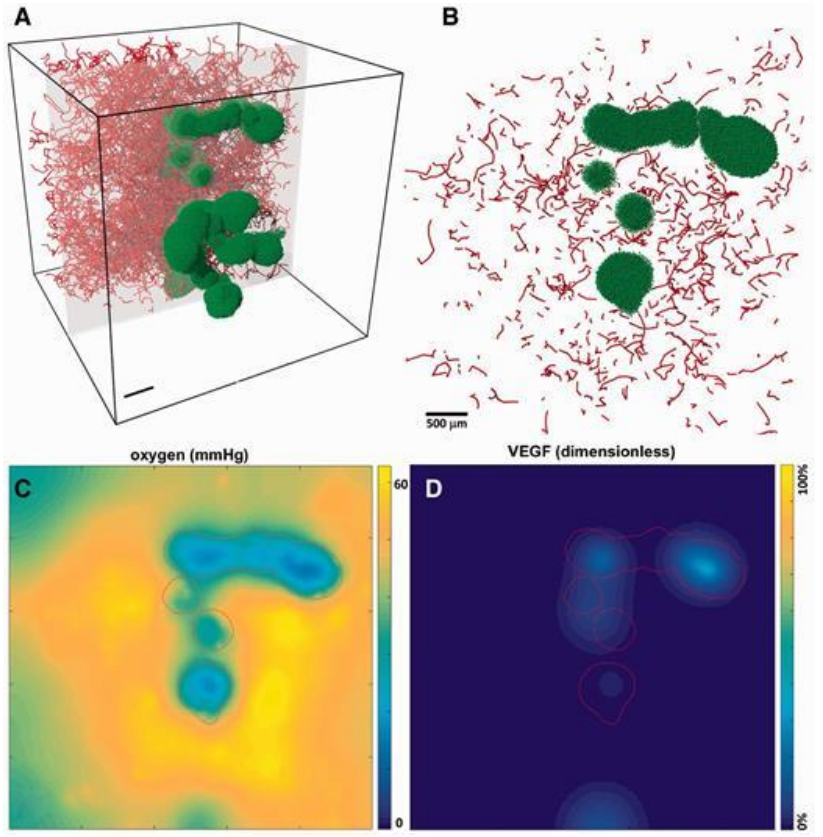
Reference: Ghaffarizadeh et al., PLoS Computational Biology (2018); [10.1371/journal.pcbi.1005991](https://doi.org/10.1371/journal.pcbi.1005991);

Introduction

PhysiCell also simulates the **diffusion and consumption/secretion of substances** in the microenvironment through **partial differential equations**.

$$\frac{\partial \vec{\rho}}{\partial t} = \overbrace{\vec{D}\nabla^2\vec{\rho}}^{\text{diffusion}} - \overbrace{\vec{\lambda}\vec{\rho}}^{\text{decay}} + \overbrace{\vec{S}(\vec{\rho}^{*} - \vec{\rho})}^{\text{bulk source}} - \overbrace{\vec{U}\vec{\rho}}^{\text{bulk uptake}}$$

$$+ \overbrace{\sum_{\text{cells}\,k} 1_k(\vec{x}) \left[\vec{S}_k(\vec{\rho}_k^{*} - \vec{\rho}) - \vec{U}_k\vec{\rho} \right]}^{\text{sources and uptake by cells}} \text{ in } \Omega$$

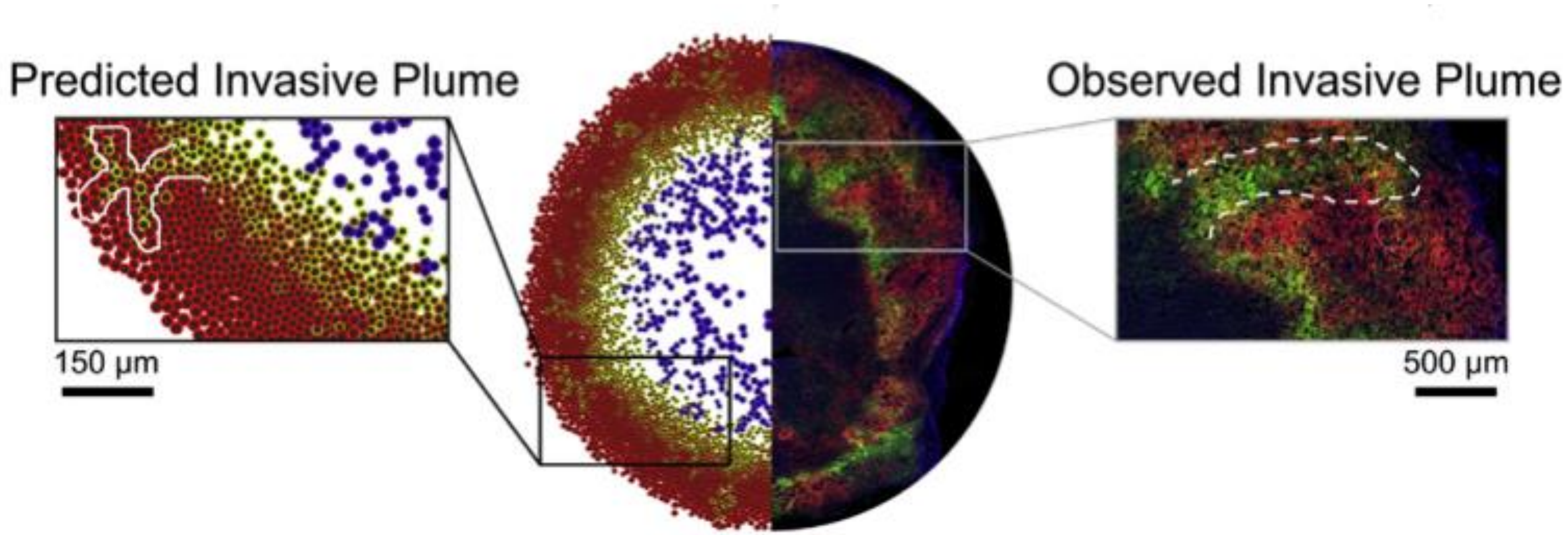


- The domain is **discretized as a grid** (2D or 3D);
- Simulations can be performed with domains at the **milimeter scale**, with a **resolution of 20 microns**

Reference: Ghaffarizadeh et al., Bioinformatics (2016); 10.1093/bioinformatics/btv730;

Applications

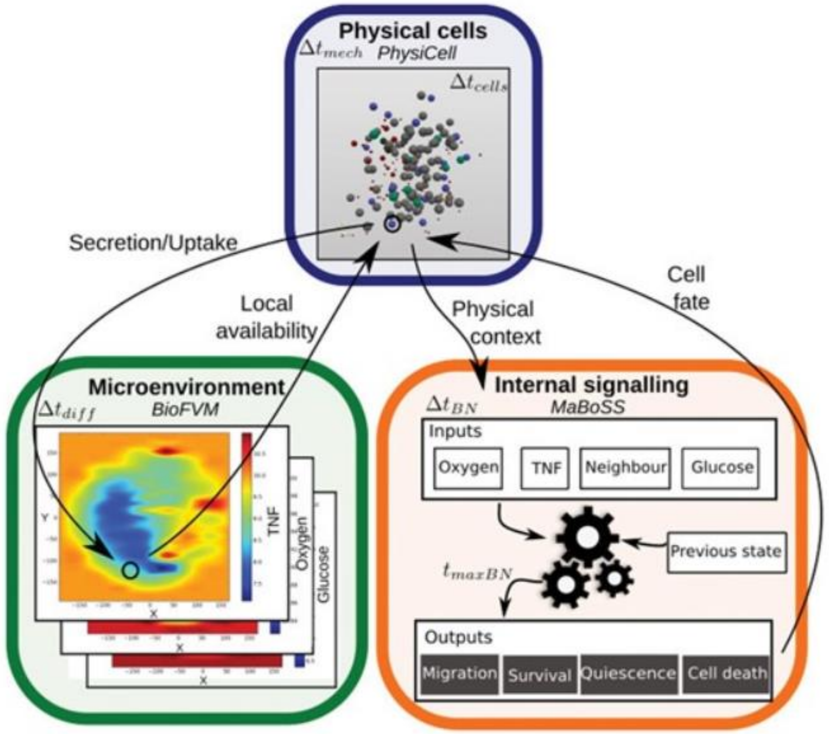
“A persistent invasive phenotype in post-hypoxic tumor cells is revealed by fate mapping and computational modeling”

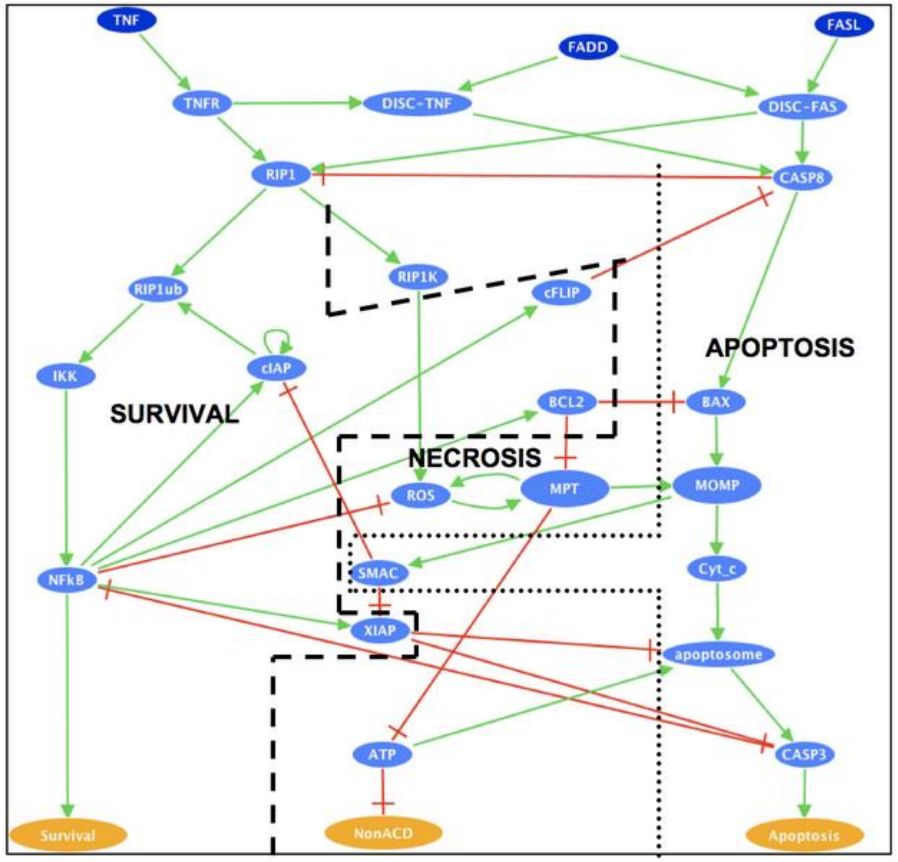


Reference: Rocha et al., iScience (2021); 10.1016/j.isci.2021.102935;

Applications

“PhysiBoSS: a multi-scale agent-based modelling framework integrating physical dimension and cell signalling”





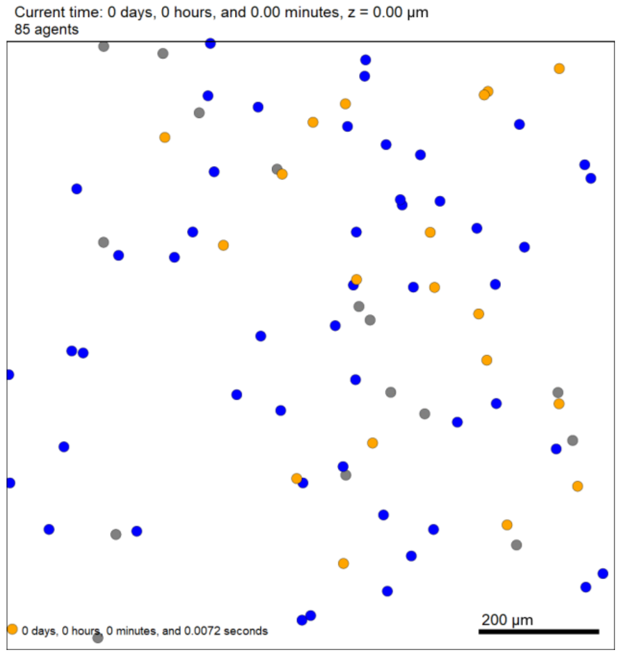
Reference: Letort et al., Bioinformatics (2019); 10.1093/bioinformatics/bty766;

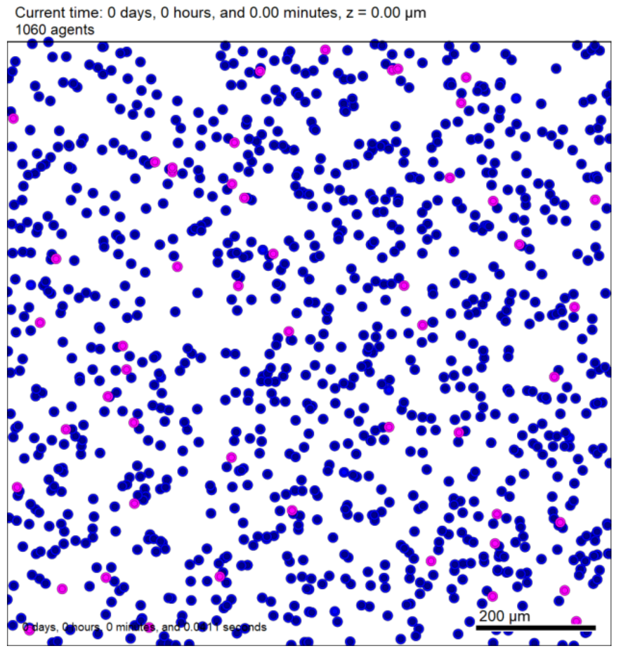
Running your 1st PhysiCell project

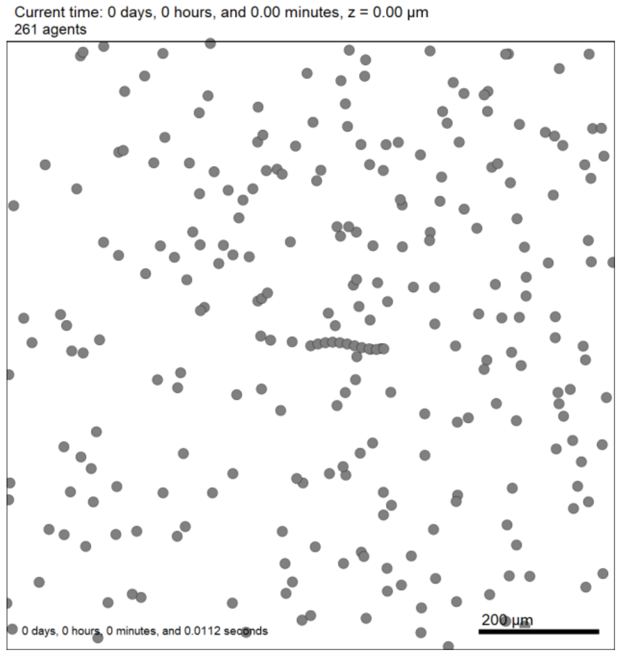
Template projects

PhysiCell comes with a series of **template projects** that run out of the box and show specific features of the code.

Users can run these models "as is", or changes can be made to adapt to a specific biological problem.







● farmer
● prey
● predator

"predator-prey-farmer"
Traditional population dynamics simulation

● epithelial cell
● macrophage

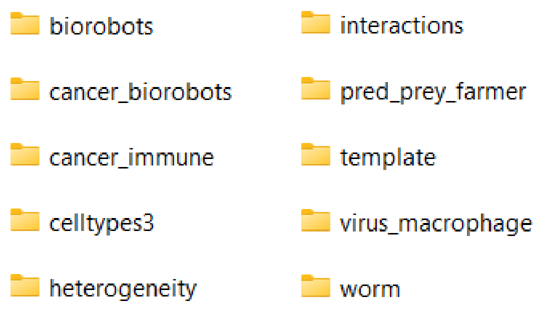
"virus_macrophage"
Internalized substances;
Cells ingesting other cells;

"worm"
Cell-cell adhesion functions;
Collective cell motility;

Running template projects

Despite being written in C++, PhysiCell aims to make it very easy for users to create and run computational models with **minimal coding experience.**

All sample projects can be run with just some **simple commands (3 lines of code).D**



biorobots
cancer_biorobots
cancer_immune
celltypes3
heterogeneity
interactions
pred_prey_farmer
template
virus_macrophage
worm

Select a project → `make <project_name>` → `make` → `./<project_name>`

Get the required files for the project

Compile the project

Run the simulations

Running your 1st project

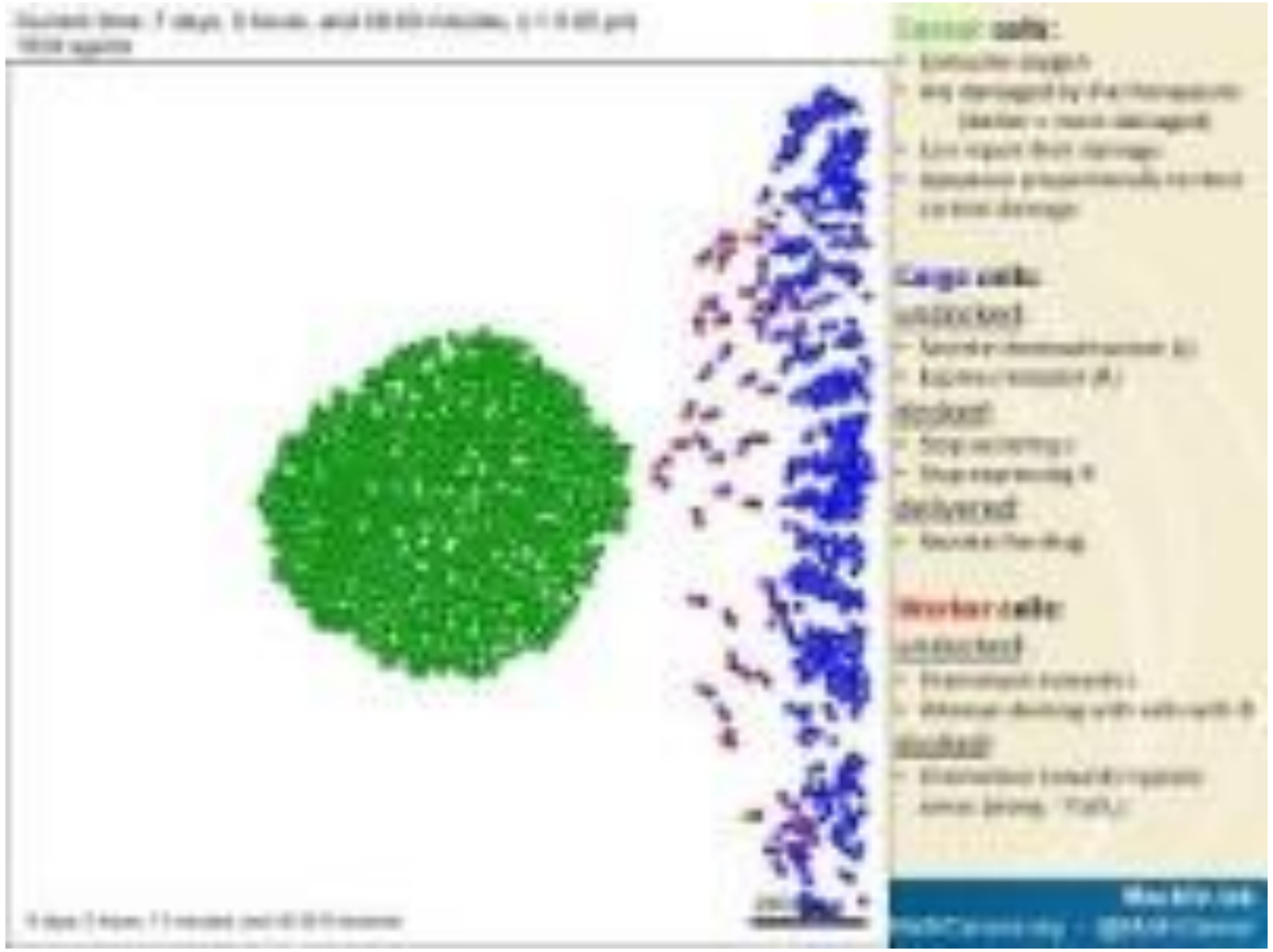
💻 Documentation:

“Running your 1st model”

Image source: https://dribbble.com/shots/8630894-Programmer-cat

Running your 1st project

Let’s open our capsule and run the sample project called “**cancer-biorobots-sample**”!

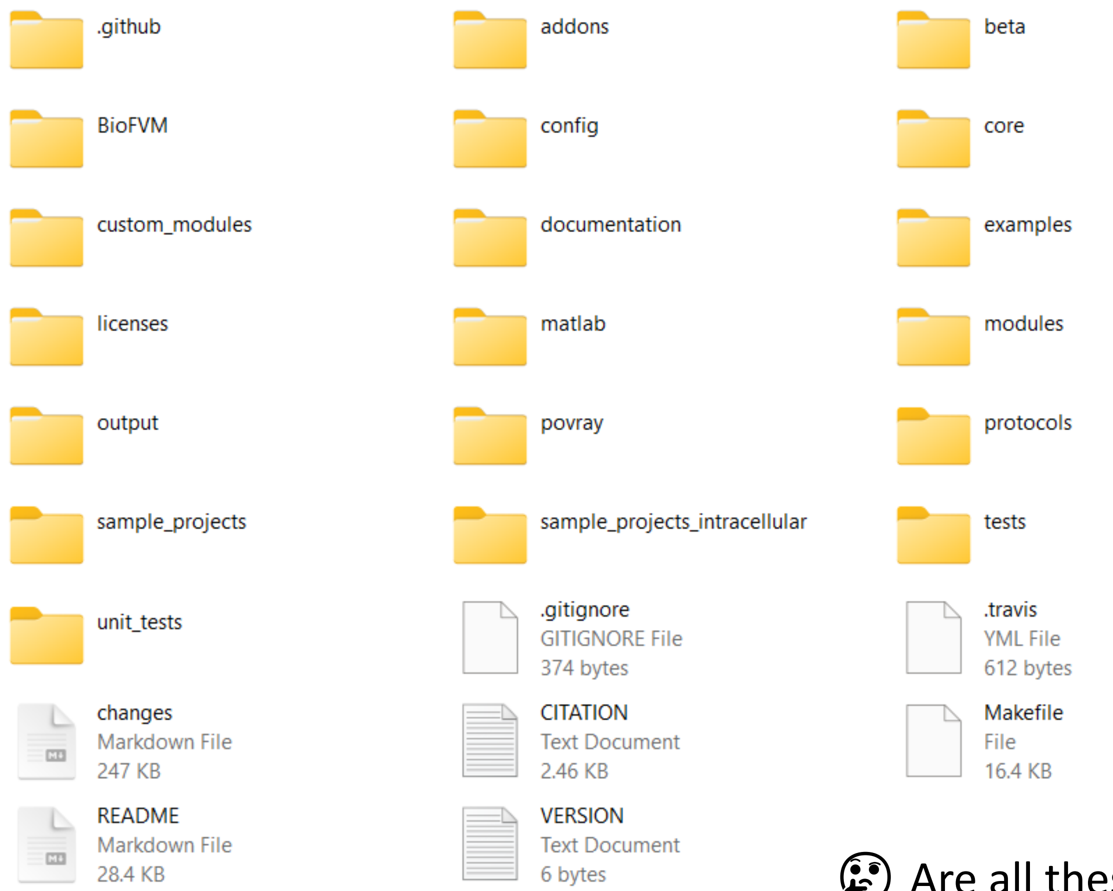


Working on models

Project structure

Once you download PhysiCell, this is the folder structure you will find:

- addons
- beta
- BioFVM
- config
- core
- custom_modules
- documentation
- examples
- licenses
- matlab
- modules
- output
- povray
- protocols
- sample_projects
- tests
- unit_tests



.github
addons
beta
BioFVM
config
core
custom_modules
documentation
examples
licenses
matlab
modules
output
povray
protocols
sample_projects
sample_projects_intracellular
tests
unit_tests

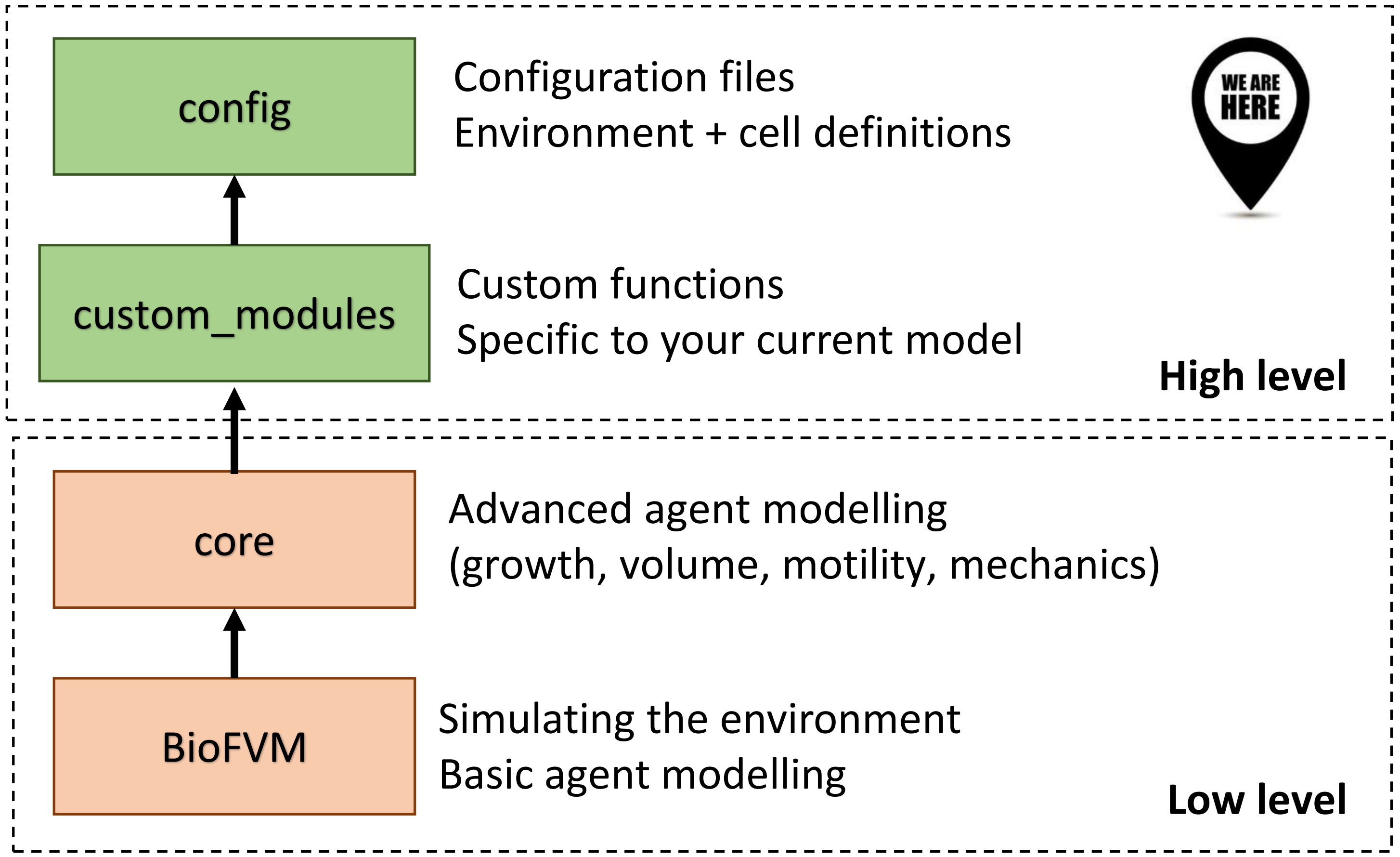
Name	Type	Size
.gitignore	GITIGNORE File	374 bytes
.travis	YML File	612 bytes
changes	Markdown File	247 KB
CITATION	Text Document	2.46 KB
Makefile	File	16.4 KB
README	Markdown File	28.4 KB
VERSION	Text Document	6 bytes

🤔 Are all these files needed?

Project structure

Once you download PhysiCell, this is the folder structure you will find:

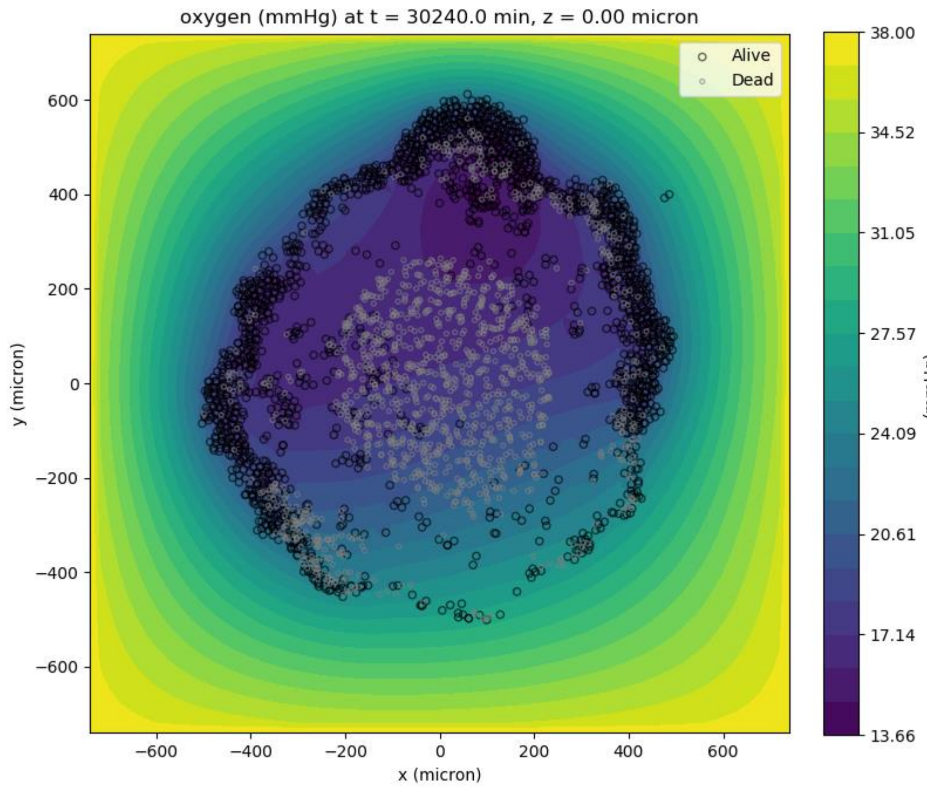
- addons
- beta
- **BioFVM**
- **config**
- **core**
- **custom_modules**
- documentation
- examples
- licenses
- matlab
- modules
- **output**
- povray
- protocols
- **sample_projects**
- tests
- unit_tests



Project structure

In the **output** folder we already used, there are two file types:

- Screenshots of the **cells' positions, sizes** and **states** (given by colors defined by the user);
- Can be turned into **GIFs** and **movies** easily;
- 2D representations;



- Stores **all the data** for the cells and the microenvironment;
- Requires further **post-processing**;

Project structure

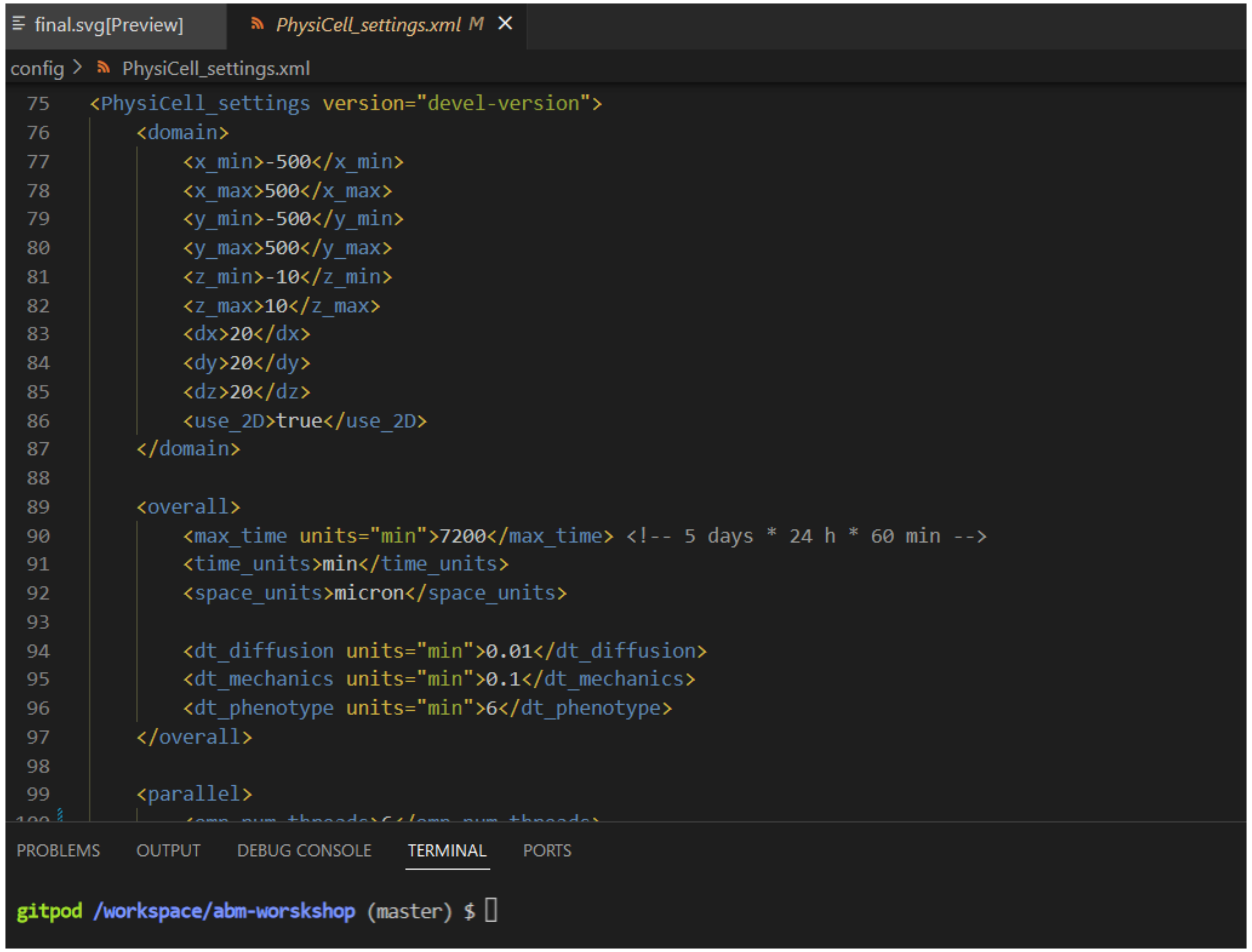
In the **config** folder, there is an XML file that defines the simulation parameters:

General simulation values:
- Domain size
- Simulation time
- Number of threads to use

Microenvironment:
- Diffusion rate
- Decay rate
- Initial and boundary conditions

Cell definitions:
- Proliferation and death rates (apoptosis, necrosis)
- Mechanics (adhesion and repulsion)
- Motility (speed, persistence time, chemotaxis, …)
- Secretion (uptake and secretion rates)
- Custom data

User custom values:
- Initial number of cells, random seed, …



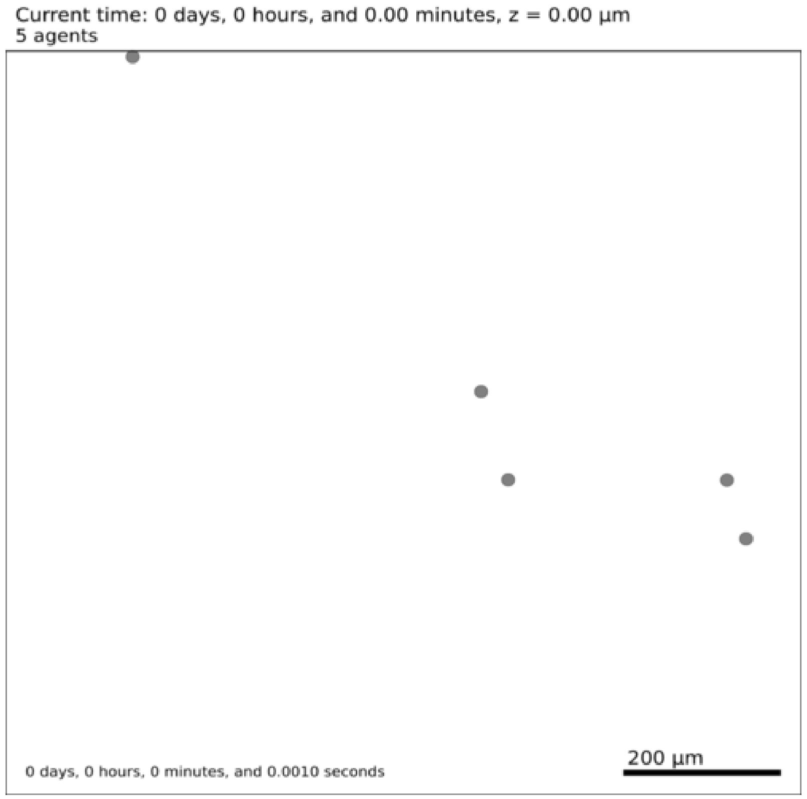
```xml
<PhysiCell_settings version="devel-version">
    <domain>
        <x_min>-500</x_min>
        <x_max>500</x_max>
        <y_min>-500</y_min>
        <y_max>500</y_max>
        <z_min>-10</z_min>
        <z_max>10</z_max>
        <dx>20</dx>
        <dy>20</dy>
        <dz>20</dz>
        <use_2D>true</use_2D>
    </domain>

    <overall>
        <max_time units="min">7200</max_time> <!-- 5 days * 24 h * 60 min -->
        <time_units>min</time_units>
        <space_units>micron</space_units>

        <dt_diffusion units="min">0.01</dt_diffusion>
        <dt_mechanics units="min">0.1</dt_mechanics>
        <dt_phenotype units="min">6</dt_phenotype>
    </overall>

    <parallel>
```

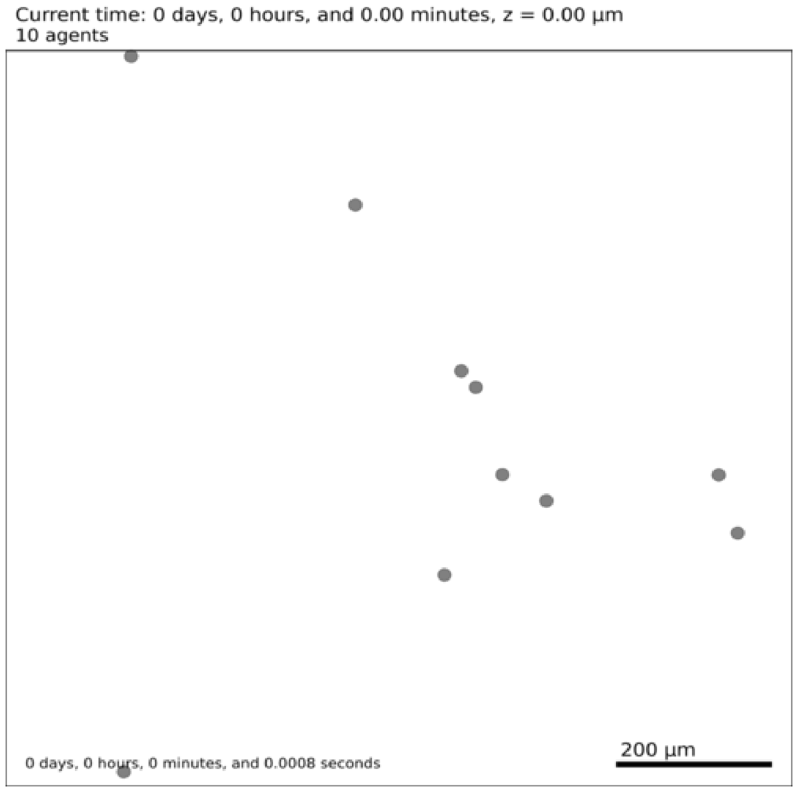
```
gitpod /workspace/abm-worskshop (master) $ 
```

Changing model inputs

Let's now run the **"template"** project, which is a very simple model with just **proliferation** and **death** rules.

Here are the results for a simulation run with **no modifications** to the config file.





Running parameter studies

💻 Documentation:

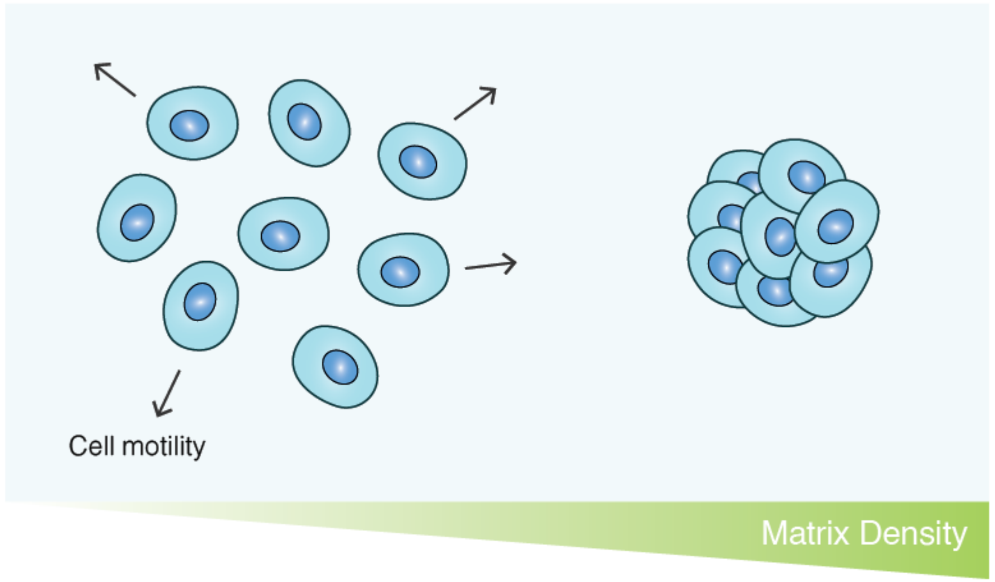
“Working with models”

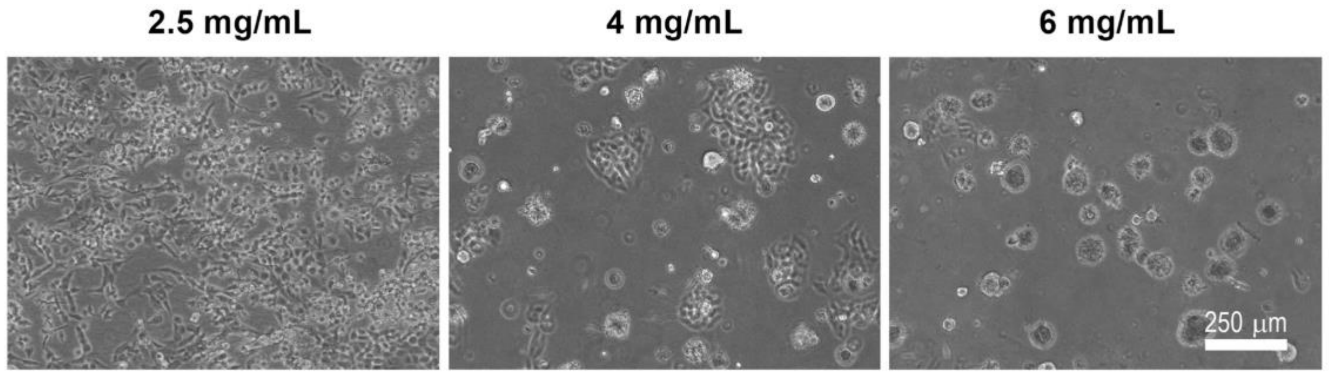
Image source: https://dribbble.com/shots/8630894-Programmer-cat

Adding model extensions

PhysiCell can also **be extended** to implement changes that cannot be defined in a simple XML file.

In this workshop, we will go through how to **add a continuous representation of the extracellular matrix (ECM)** to the microenvironment and have it **influence cell motility**.



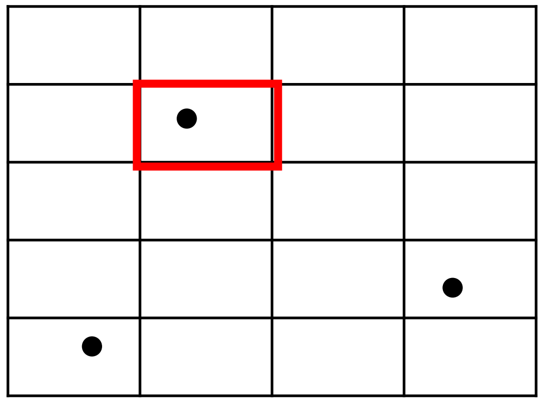


$$\mathbf{v}_i = \frac{1}{\mu}\left(\sum_{j\in\mathcal{N}(i)}\left(\mathbf{F}^{ij}_{cca} + \mathbf{F}^{ij}_{ccr}\right) + \mathbf{F}^{i}_{loc}\right)$$

Reference: Gonçalves et al., PLoS Computational Biology (2021); 10.1371/journal.pcbi.1008764;

Adding model extensions

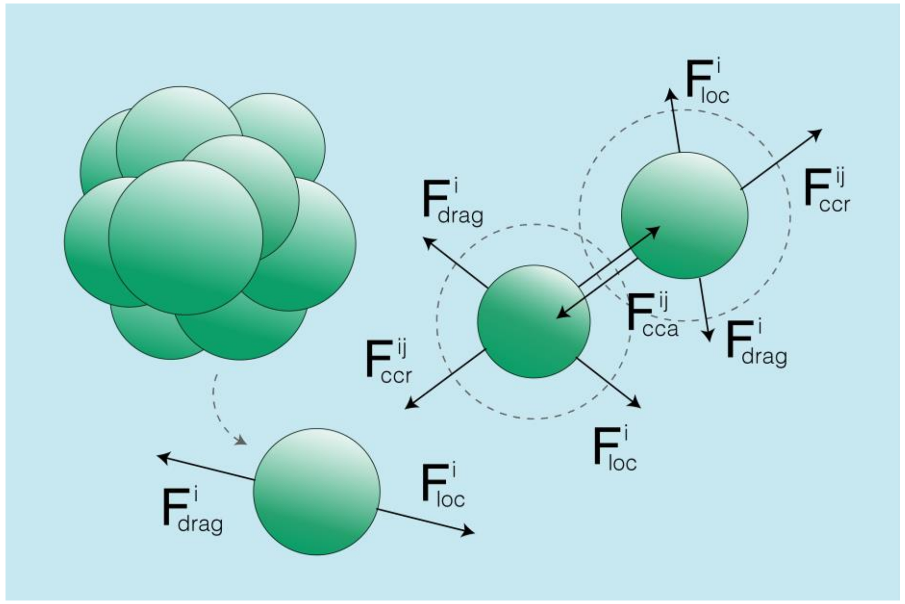
In this workshop, we will go through how to **add a continuous representation of the extracellular matrix (ECM)** to the microenvironment and have it **influence cell motility**.



Sampling the concentration at the current voxel

Concentration (mg/mL)	Viscosity (Pa·s)
2.5	7.96
4.0	18.42
6.0	39.15

Estimating a viscosity value based on concentration



$$\mathbf{v}_i = \frac{1}{\mu}\left(\sum_{j\in\mathcal{N}(i)} (\mathbf{F}^{ij}_{cca} + \mathbf{F}^{ij}_{ccr}) + \mathbf{F}^{i}_{loc}\right)$$

Adding model extensions

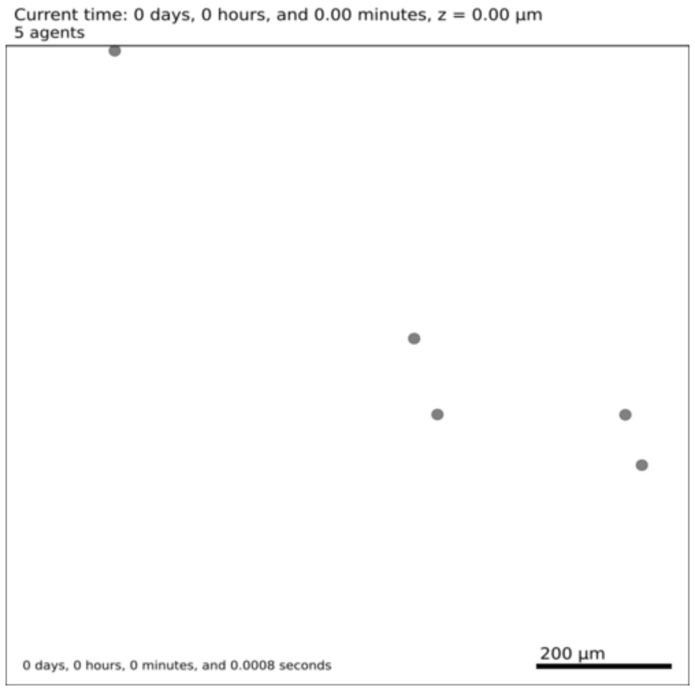
💻 Documentation:

“Adding model extensions”

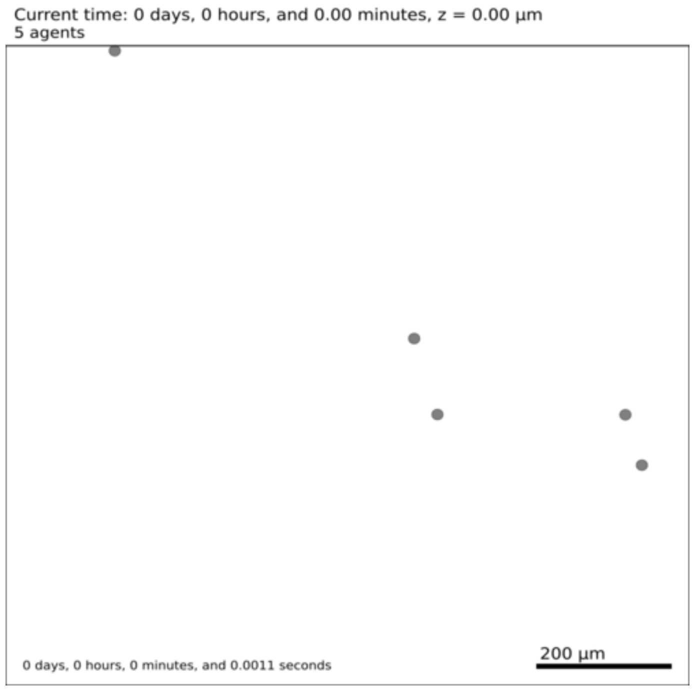
Image source: https://dribbble.com/shots/8630894-Programmer-cat

Adding model extensions

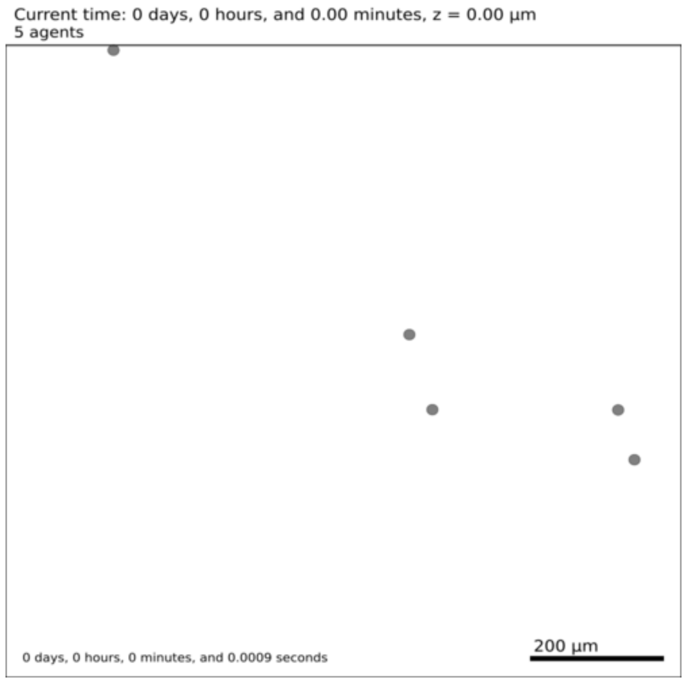
Collagen: 2.5 mg/mL



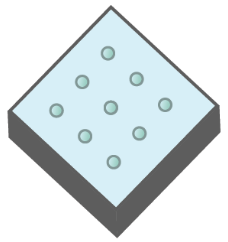
Collagen: 4.0 mg/mL



Collagen: 6.0 mg/mL

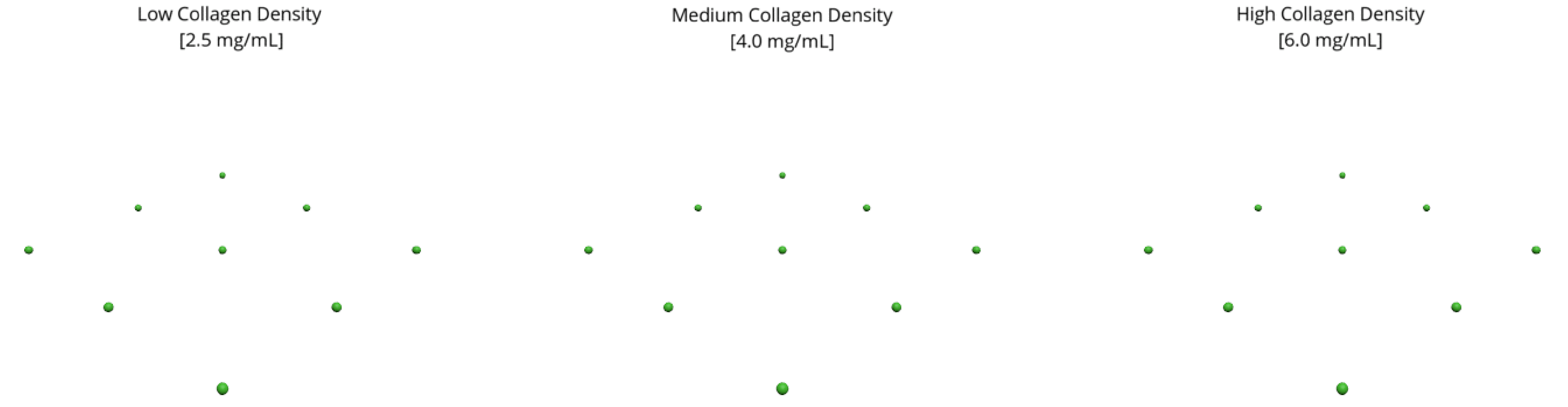


Advanced visualization



Setup 2 (Cell-cell interactions)

Nine initial cells
Cell death
Cell duplication
168 hour simulations



Reference: Gonçalves et al., PLoS Computational Biology (2021); [10.1371/journal.pcbi.1008764](https://doi.org/10.1371/journal.pcbi.1008764);

Running parameter estimation studies

Model characterization

Up until now, we have been running single simulations. However, it is important to be able to characterize **how a model responds** to **changes in parameter values**

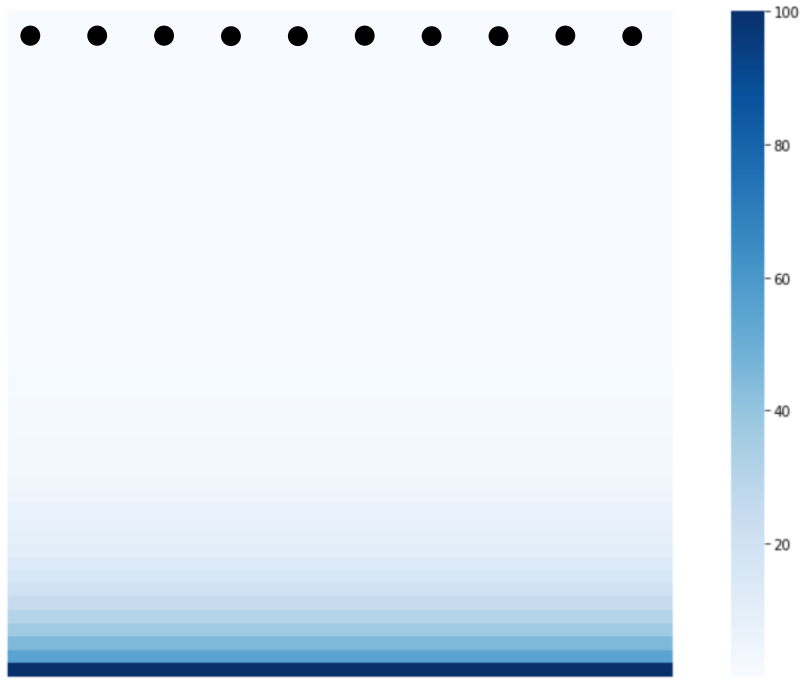
$$\begin{tikzcd} x1 \arrow[r] & \boxed{\text{Model}} \arrow[r] & y1 \\ x2 \arrow[r] & & y2 \end{tikzcd}$$

Model to be tested: PhysiCell motility example

Using the "template" project by PhysiCell;

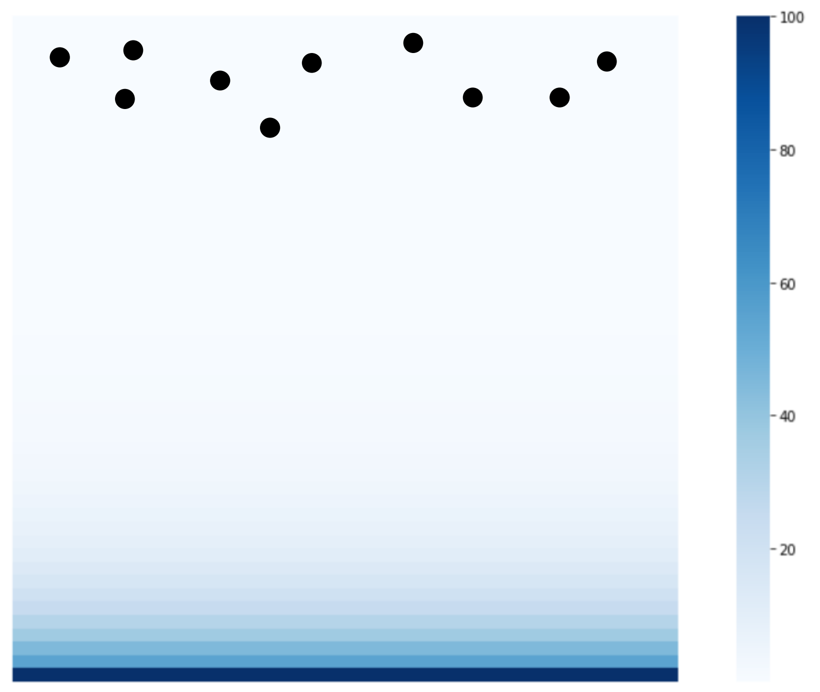
Induce a chemotactic gradient by setting a boundary condition with oxygen on one of the domain walls;

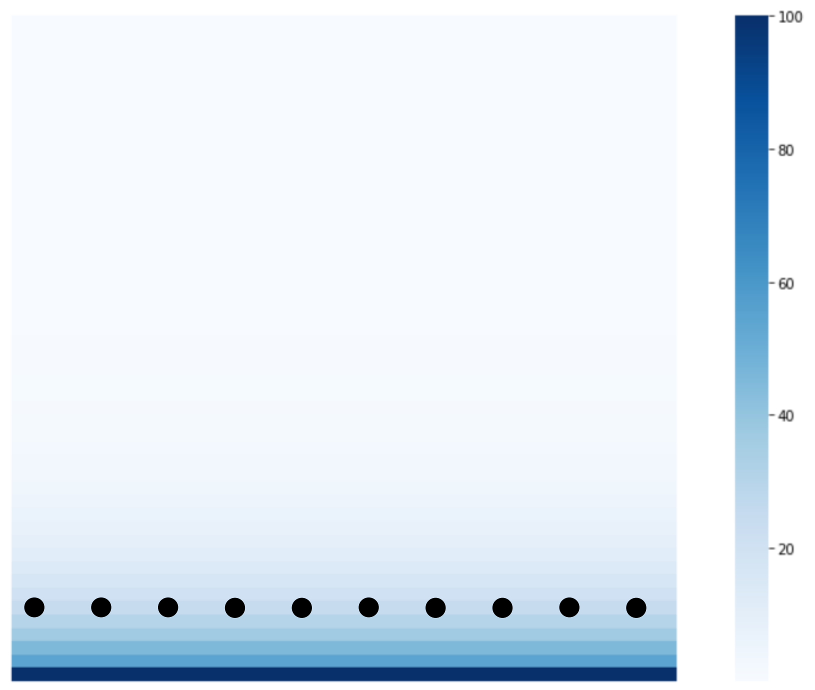
Change the chemotactic response with the motility parameters;



Oxygen conditions
A source is placed on the wall opposite to the cells;
Cells are expected to migrate towards the gradient (if bias > 0);

Model characterization





Low migration bias (low sensitivity to oxygen)

Cells will move randomly and present almost null net displacement

High migration bias (high sensitivity to oxygen)

Cells will move towards the source and travel long distances in that direction

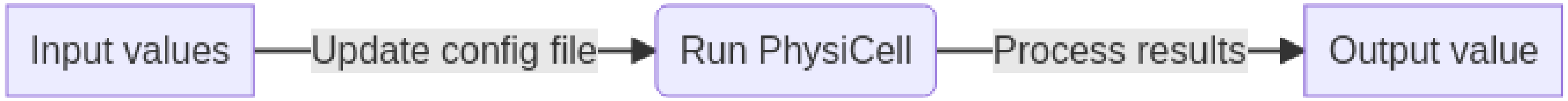
PhysiCell "black-box"

Running these types of studies manually is **time consuming** and **inefficient**.

PhysiCOOL acts as a Python wrapper to the PhysiCell C++ code so that simulations can be run through Python. This enables us to **connect our models to powerful** and **well-established optimization libraries**, or to **our own scripts**.

PhysiCell-based "black box"

- Receives input parameters
- Updates the XML configuration file
- Runs project (calls the compiled PhysiCell file)
- Computes and returns a given output metric



Running your 1st project

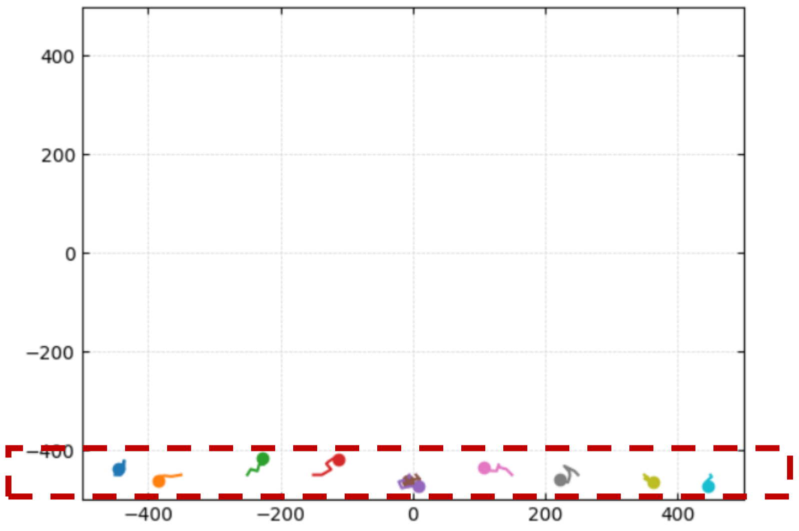
💻 Documentation:

“Running parameter estimation studies”

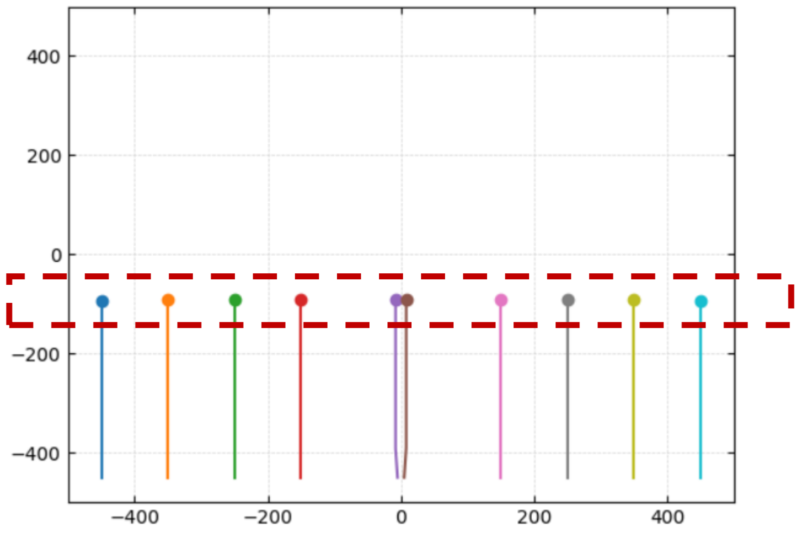
Image source: https://dribbble.com/shots/8630894-Programmer-cat

Model optimization

Speed: 1 micron/min; migration bias: 0.0



Speed: 1 micron/min; migration bias: 1.0



Can we **pose the question in reverse**? In other words, if **given some data**, can we **estimate what model parameter value** originated it?

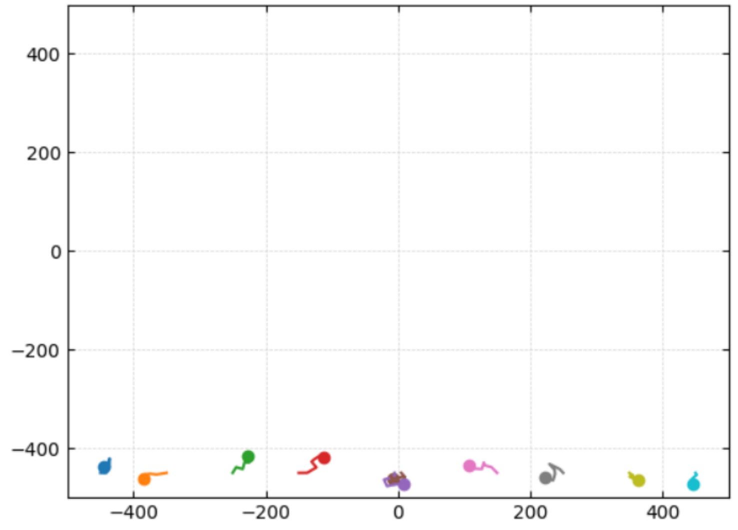
Model optimization

Given some data, can we **estimate what model parameter value** originated it?

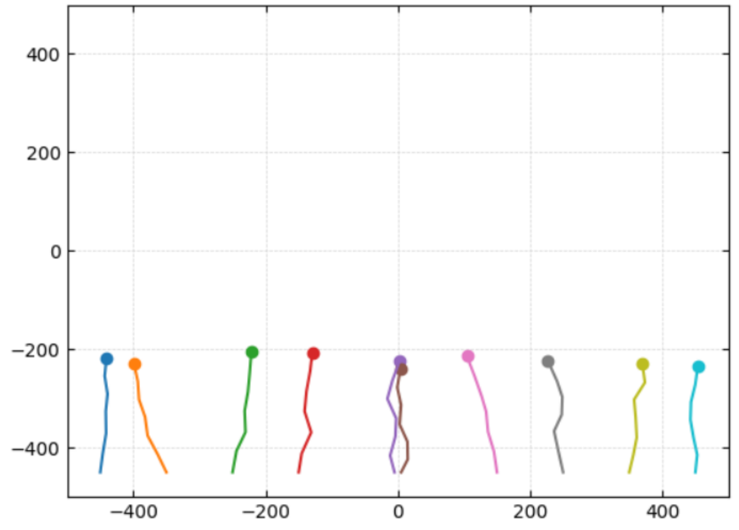
0.00	0.25	0.50	0.75	1.00

Optimization routine ("brute-force"):

- Created a parameter space;
- For each cell of the grid:
  - Run the model;
  - Compare it with target data;
  - Quantify the error;
- Choose the value with the best results;



Simulated data



Target data

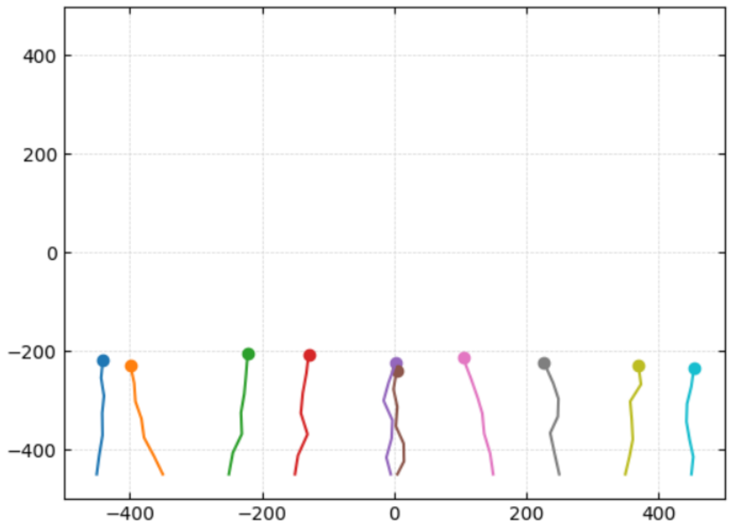
Model optimization

Given some data, can we **estimate what model parameter value** originated it?

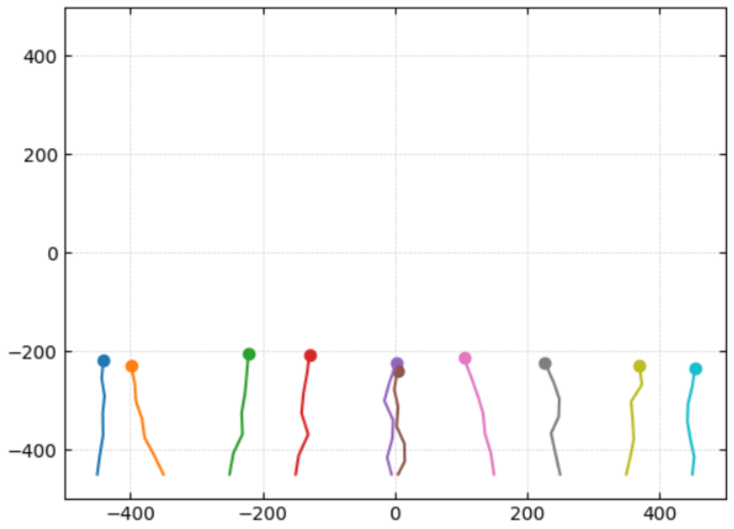
0.00	0.25	0.50	0.75	1.00

Optimization routine ("brute-force"):

- Created a parameter space;
- For each cell of the grid:
  - Run the model;
  - Compare it with target data;
  - Quantify the error;
- Choose the value with the best results;



Simulated data



Target data

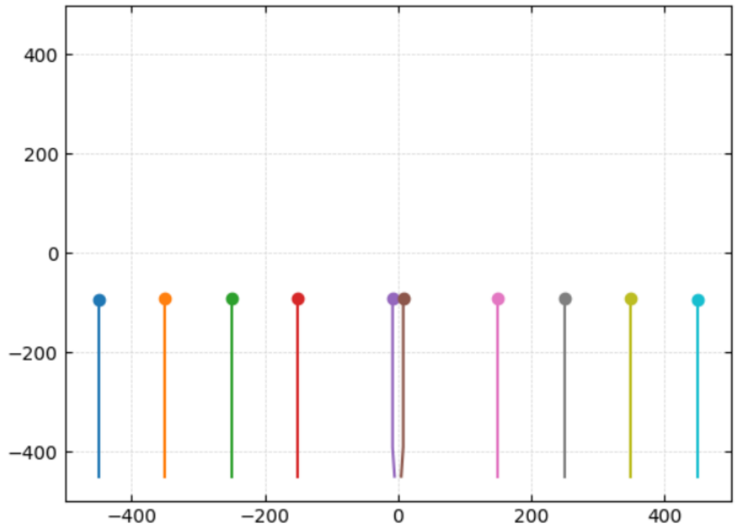
Model optimization

Given some data, can we **estimate what model parameter value** originated it?

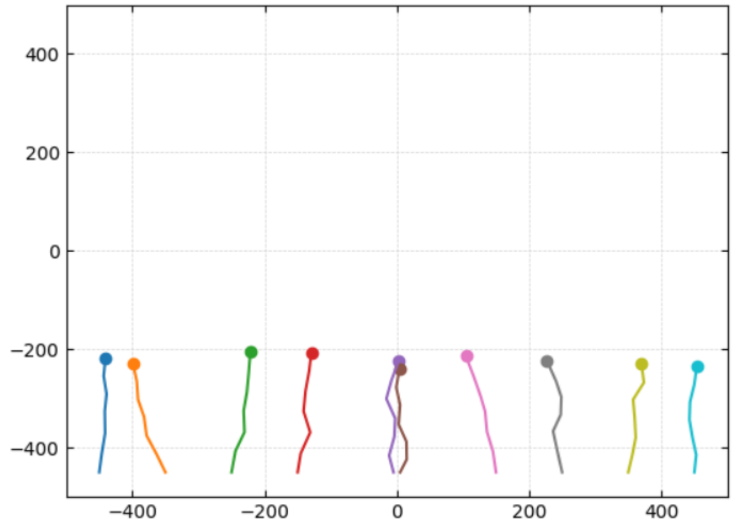
0.00	0.25	0.50	0.75	1.00

Optimization routine ("brute-force"):

- Created a parameter space;
- For each cell of the grid:
  - Run the model;
  - Compare it with target data;
  - Quantify the error;
- Choose the value with the best results;



Simulated data



Target data

Model optimization

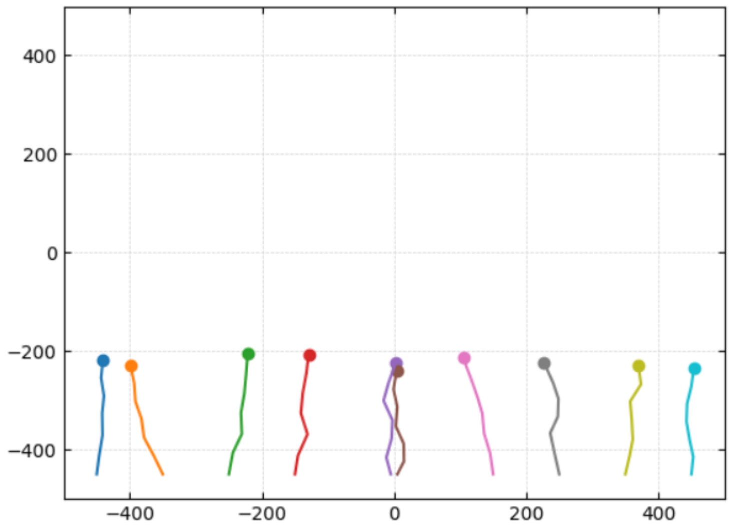
Given some data, can we **estimate what model parameter value** originated it?

0.00	0.25	0.50	0.75	1.00

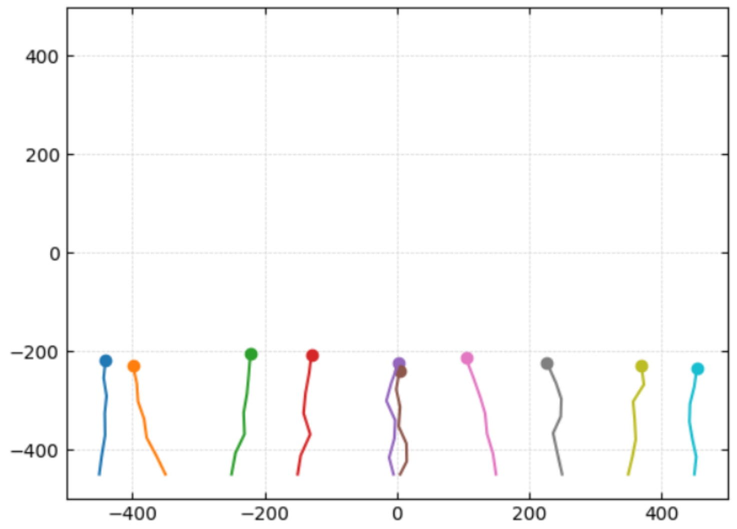
Parameter value found 🥳

Optimization routine ("brute-force"):

- Created a parameter space;
- For each cell of the grid:
    - Run the model;
    - Compare it with target data;
    - Quantify the error;
- Choose the value with the best results;



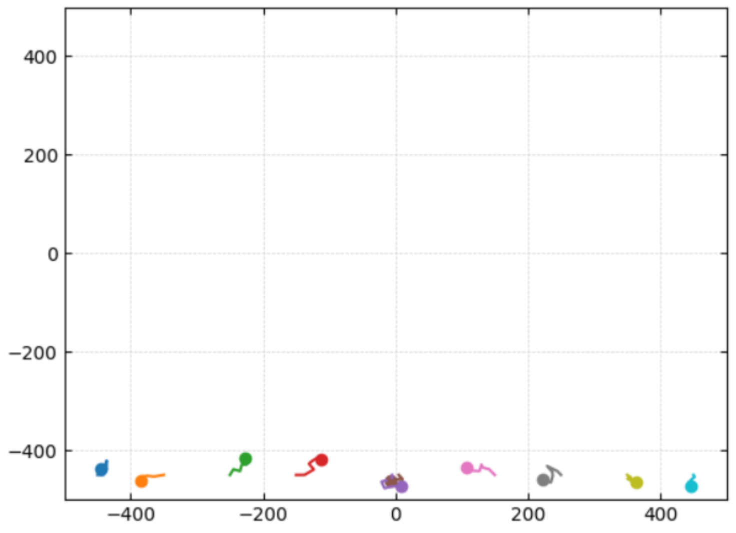
Simulated data



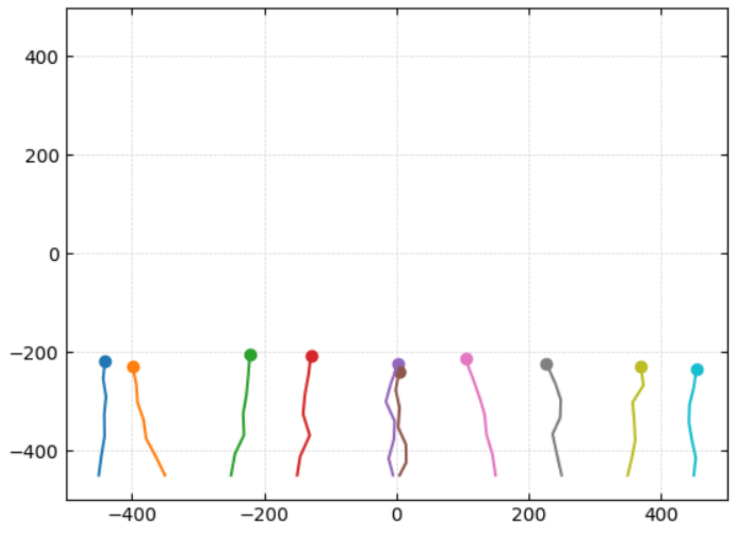
Target data

Model optimization

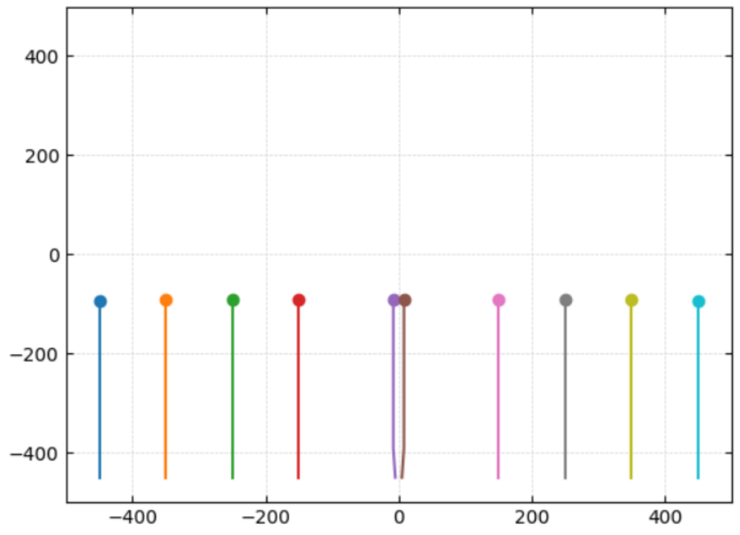
Speed: 1 micron/min; migration bias: 0.0



Speed: 1 micron/min; migration bias: 0.5

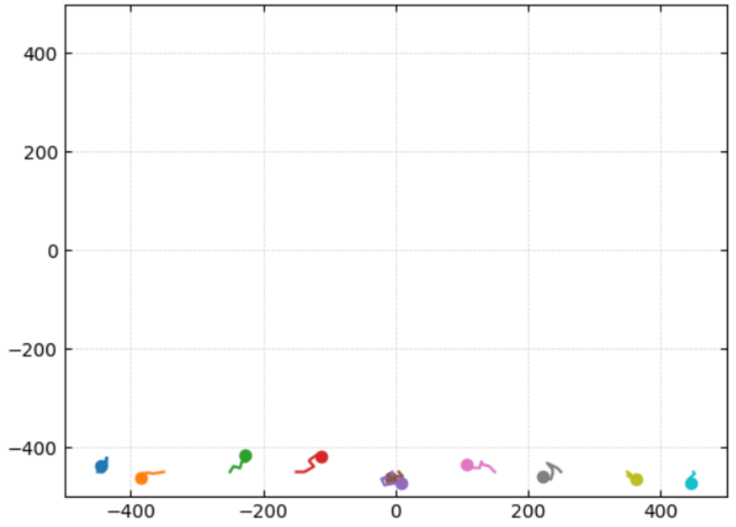


Speed: 1 micron/min; migration bias: 1.0

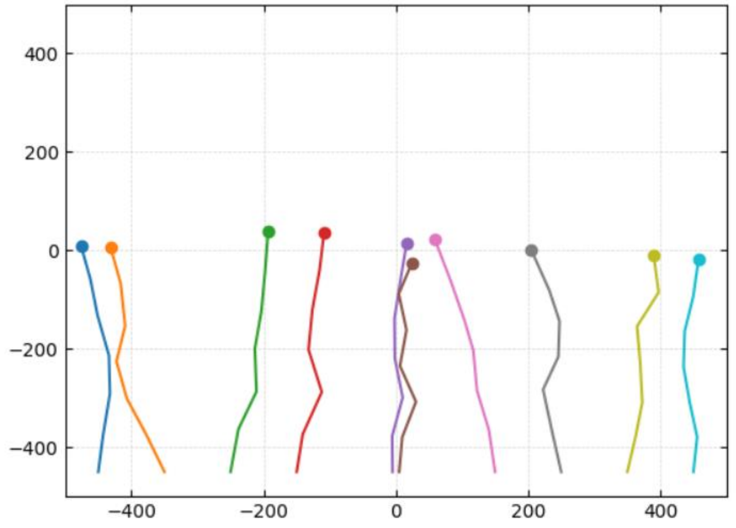


Model optimization

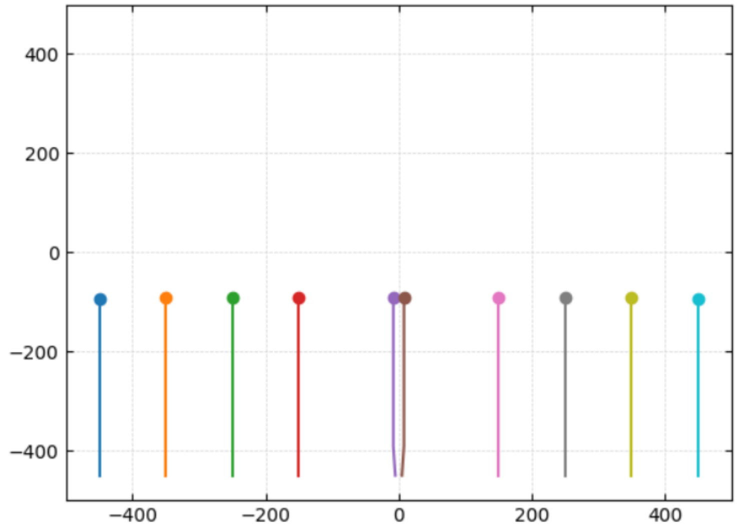
Speed: 1 micron/min; migration bias: 0.0



Speed: 2 micron/min; migration bias: 0.5



Speed: 1 micron/min; migration bias: 1.0



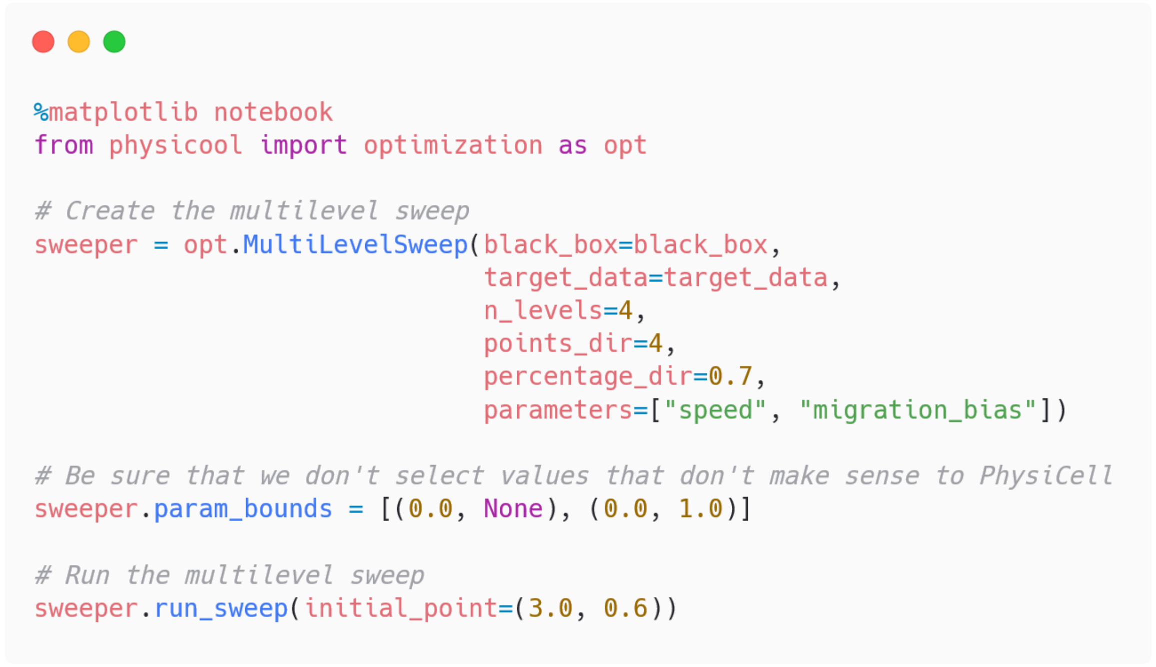
PhysiCOOL implements a class called **MultiLevelSweep** that performs calibration studies with **two variables**.

It also performs the routine in an iterative manner: once it **finds the best value in a parameter space**, it **refines the parameter bounds** and repeats the task.

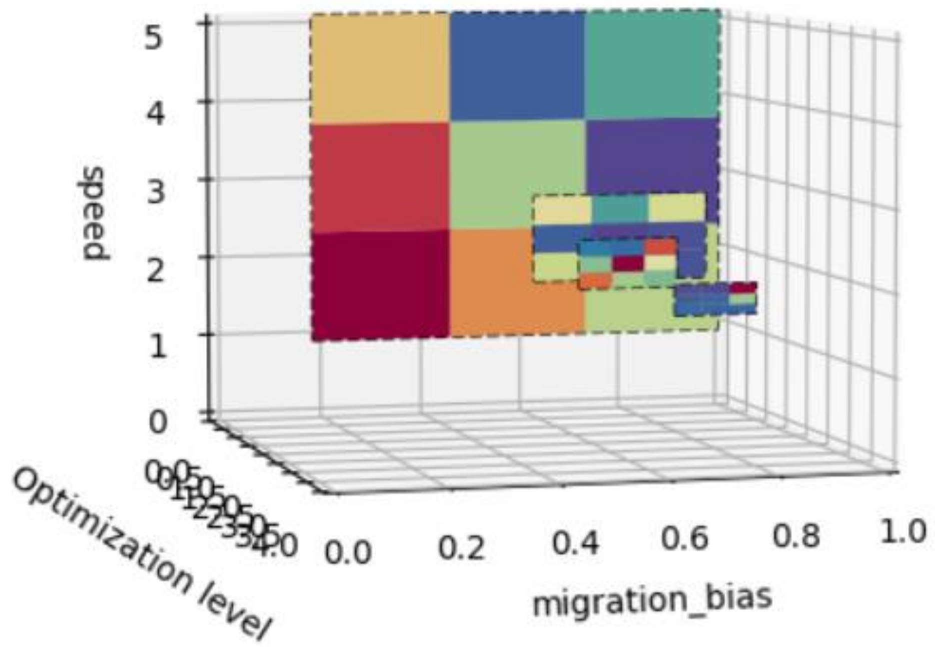
Multilevel sweep

Running the multilevel parameter sweep

Initial values: migration_bias = 0.9, speed = 2.0



```
%matplotlib notebook
from physicool import optimization as opt

# Create the multilevel sweep
sweeper = opt.MultiLevelSweep(black_box=black_box,
                              target_data=target_data,
                              n_levels=4,
                              points_dir=4,
                              percentage_dir=0.7,
                              parameters=["speed", "migration_bias"])

# Be sure that we don't select values that don't make sense to PhysiCell
sweeper.param_bounds = [(0.0, None), (0.0, 1.0)]

# Run the multilevel sweep
sweeper.run_sweep(initial_point=(3.0, 0.6))
```



Optimal values found

migration_bias: 0.84; speed: 1.89

Running your 1st project

💻 Notebook:

“optimizer.ipynb”

Image source: https://dribbble.com/shots/8630894-Programmer-cat

Further reading and online resources

Adding the ECM model

More information on PhysiCell:

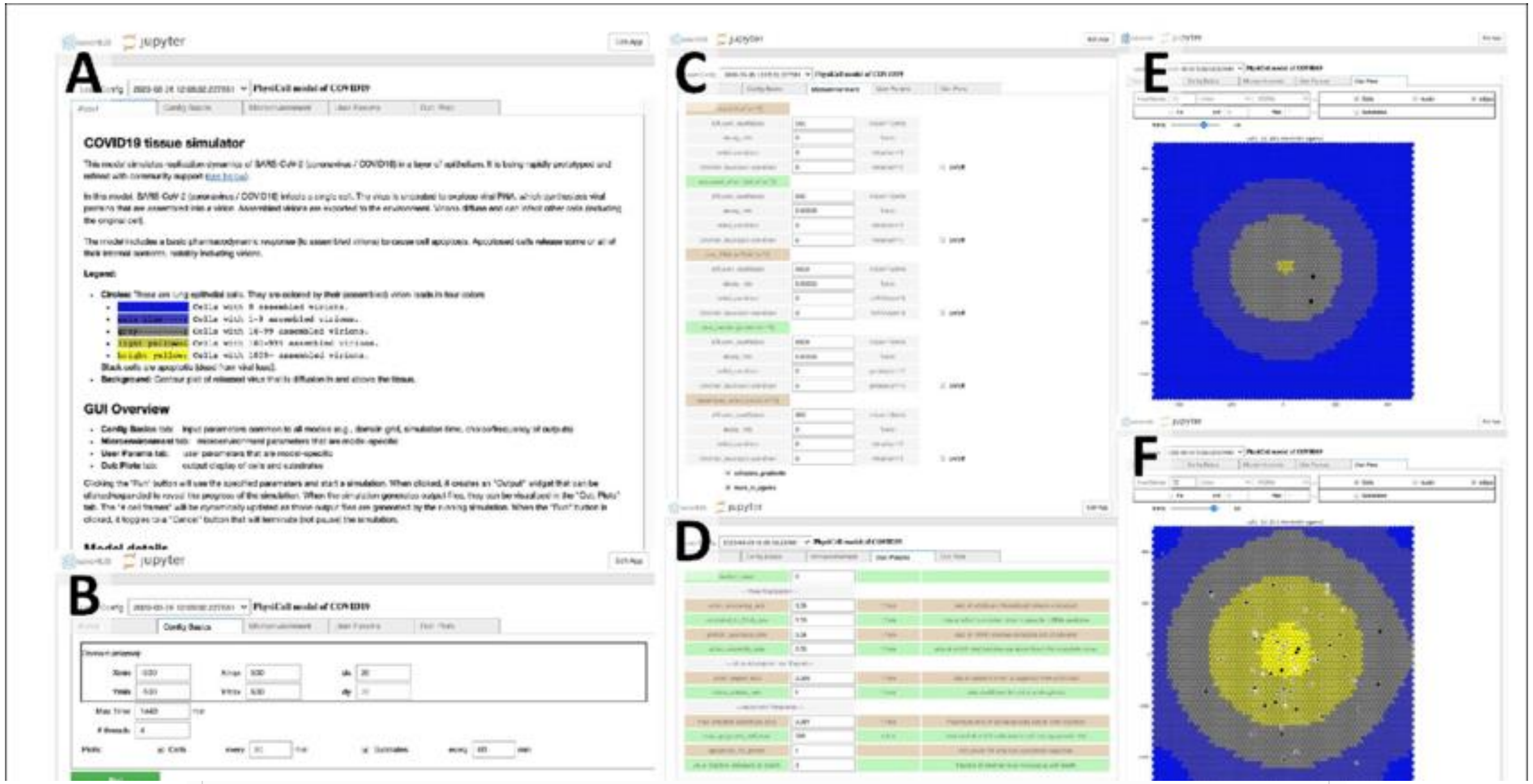
PLOS Computational paper

MathCancer Blog

PhysiCell workshop 2022

- GitHub – code + slides;
- YouTube – recorded lectures;

NanoHub interactive models



All the parameters for cell-cell interactions are stored in `cell.phenotype.cell_interactions`.

Online resources

https://github.com/IGGoncalves/PhysiCOOL

Code is hosted on GitHub (source code + Jupyter exemples)

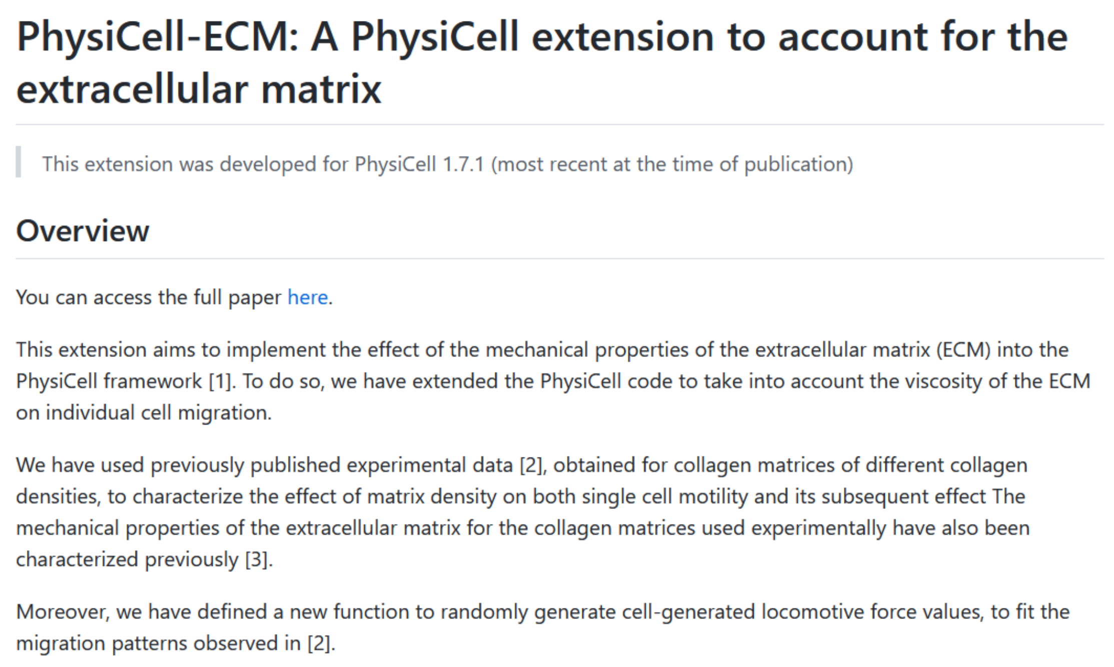
https://pypi.org/project/physicool/

PhysiCOOL is distributed through PyPi and can be installed with pip.

```
pip install physicool
```

https://physicool.readthedocs.io/

Documentation is available on ReadTheDocs.



PhysiCell-ECM: A PhysiCell extension to account for the extracellular matrix

> This extension was developed for PhysiCell 1.7.1 (most recent at the time of publication)

Overview

You can access the full paper here.

This extension aims to implement the effect of the mechanical properties of the extracellular matrix (ECM) into the PhysiCell framework [1]. To do so, we have extended the PhysiCell code to take into account the viscosity of the ECM on individual cell migration.

We have used previously published experimental data [2], obtained for collagen matrices of different collagen densities, to characterize the effect of matrix density on both single cell motility and its subsequent effect The mechanical properties of the extracellular matrix for the collagen matrices used experimentally have also been characterized previously [3].

Moreover, we have defined a new function to randomly generate cell-generated locomotive force values, to fit the migration patterns observed in [2].

XI REUNIÓN DEL CAPÍTULO ESPAÑOL DE LA SOCIEDAD EUROPEA DE BIOMECÁNICA

Thank you for your attention!

Inês G. Gonçalves, PhD Student
iggoncalves@unizar.es

24 de octubre de 2022