PhysiBoSS User Guide (Version 2.0.0)

PhysiBoSS Project

Revision: June 28, 2024

Contents

1 Introduction and citing PhysiBoSS **1**

2 Getting started: Your First Simulation **2**

3 PhysiBoSS XML definition **3**

4 PhysiBoSS API : MaBoSSIntracellular class **5**

5 PhysiBoSS Makefile **7**

6 Examples **9**

6.1 PhysiBoSS Cell Lines model 9

7 Future **10**

8 Acknowledgements **10**

Bibliography **10**

1 Introduction and citing PhysiBoSS

This user guide will teach you how to use PhysiBoSS [2, 3], as well as document the key classes and functions. Wherever possible, it will demonstrate with specific examples. Please note that this guide will be periodically updated.

The original PhysiBoSS paper was published in PLoS Computational Biology [2], and it's official inclusion as an addon on PhysiCell[1] is on BioRxiv [3].

If you use PhysiBoSS, please cite it as:

We implemented and solved our model using PhysiBoSS (Version 2.0.0) [1,2].

[1] G. Letort, A. Montagud, G. Stoll, R. Heiland, E. Barillot, P. Macklin, A. Zinovyev, and L. Calzone. Physiboss: a multi-scale agent-based modelling framework integrating physical dimension and cell signalling. Bioinformatics, 35(7):1188–1196, 2019.
[2] M. Ponce-de Leon, A. Montagud, V. Noël, G. Pradas, A. Meert, E. Barillot, L. Calzone, and A. Valencia. Physiboss 2.0: a sustainable integration of stochastic boolean and agent-based modelling frameworks. bioRxiv, pages 2022–01, 2022.

Because PhysiBoSS is an addon on PhysiCell, we suggest you also cite it:

We implemented and solved the model using PhysiBoSS (Version 2.0.0) [1,2], an addon of PhysiCell[3]

[1] G. Letort, A. Montagud, G. Stoll, R. Heiland, E. Barillot, P. Macklin, A. Zinovyev, and L. Calzone. Physiboss: a multi-scale agent-based modelling framework integrating physical dimension and cell signalling. Bioinformatics, 35(7):1188–1196, 2019.
[2] M. Ponce-de Leon, A. Montagud, V. Noël, G. Pradas, A. Meert, E. Barillot, L. Calzone, and A. Valencia. Physiboss 2.0: a sustainable integration of stochastic boolean and agent-based modelling frameworks. bioRxiv, pages 2022–01, 2022.
[3] A. Ghaffarizadeh, R. Heiland, S.H. Friedman, S.M. Mumenthaler, and P. Macklin. PhysiCell: an Open Source Physics-Based Cell Simulator for 3-D Multicellular Systems, PLoS Comput. Biol. 14(2): e1005991, 2018. DOI: 10.1371/journal.pcbi.1005991.

[Return to Table of Contents.]

2 Getting started: Your First Simulation

PhysiBoSS comes with an easy to use example to quickly start simulating your first PhysiBoSS project. First, to load the project, run :

```
make physiboss-cell-lines-sample
```

This will copy all the project files to the root, config and custom_modules folder. Then, to build the project, run :

```
make
```

Finally, to execute the project, run :

```
./PhysiBoSS_Cell_Lines
```

[Return to Table of Contents.]

3 PhysiBoSS XML definition

Here is an example of the XML configuration of a PhysiBoSS model :

```
<cell_definition name="..." ID="...">
   ...
   <phenotype>
      ...
      <intracellular type="maboss">
         <bnd_filename>./config/model_0.bnd</bnd_filename>
         <cfg_filename>./config/model.cfg</cfg_filename>
         <time_step>10</time_step>
         <scaling>60</scaling>
         <time_stochasticity>0.0</time_stochasticity>
         <initial_values>
            <initial_value node="A">1</initial_value>
            <initial_value node="C">0</initial_value>
         </initial_values>
         <mutations>
            <mutation node="C">0.0</mutation>
         </mutations>
         <parameters>
            <parameter name="$time_scale">0.2</parameter>
         </parameters>
      </intracellular>
      ...
   </phenotype>
   ...
</cell_definition>
```

This example use all available options of PhysiBoSS 2.0.0. Let's look at them one by one :

```
<bnd_filename>./config/model_0.bnd</bnd_filename>
<cfg_filename>./config/model.cfg</cfg_filename>
```

A MaBoSS model is defined by two files : one for the model definition (BND file) and one for the simulation settings (CFG file). Here we configure the MaBoSS model which will be used in the PhysiBoSS simulation.

```
<time_step>10</time_step>
```

PhysiCell will update its internal mechanisms regularly, depending on the time scale of the process described: diffusion of molecules in the environment will be governed by a short time scale, so its update will have to be performed very often (usually every seconds), while cell cycle progression will be governed by a large time scale so its update will be performed less frequently (usually every few minutes). A similar mechanism is used for PhysiBoSS, where the status of the MaBoSS model will be updated at every time_step. Since this is model-dependent, we leave it to the user to define the appropriate time_step.

```
<scaling>60</scaling>
```

Now let us suppose that our PhysiBoSS model is updating every 10 minutes, meaning that every 10 minutes, we will update the inputs of the MaBoSS model, simulate it, and the obtain its outputs nodes (phenotypes). The follow up question is how long should we simulate our MaBoSS model. A MaBoSS model has its own time unit since it contains kinetic rates for (in)activation, which is not always the same as PhysiCell unit (usually minutes) and we might have to adapt the simulation duration to PhysiCell unit. This is done using the scaling parameter. Let us suppose that the MaBoSS model's unit is hours. Then we need to set our scaling parameter to 60 so that every 10 minutes, PhysiBoSS will run the MaBoSS model for time_step / scaling = 10/60 = 1/6 hours. Another way to see this scaling parameter is that if you want to run you MaBoSS simulation for 20 (MaBoSS time unit), you need to set scaling = time_step / maboss_simulation_time.

```
<time_stochasticity>0.0</time_stochasticity>
```

Biology is governed by many stochastic processes, and thus there is always noise in the duration of its processes. This is observed by the loss of synchronization of cell populations, for example is study about cell cycle progression. Cells at the beginning of the experiment can show synchronized cell cycle progression, but after a few cycles this synchronization will be lost. In PhysiBoSS, while we still have an heterogenous response of our MaBoSS models, when we select a large enough time_step we can end up in a situation where all our cells are synchronized (by the update at regular time_step). In order to avoid this artificial synchronization, we introduce a parameter called time_stochasticity that will modify the value of the time_step for every cell, so that it follows a Log Normal distribution with an average equal to time_step, with a deviation a exp(time_stochasticity). This allows cells to naturally lose their artificial synchronicity.

```
<initial_values>
   <initial_value node="A">1</initial_value>
   <initial_value node="C">0</initial_value>
</initial_values>
```

While the initial values are defined by default in the model CFG file, we provide the possibility for the user to modify them in the XML configuration. The value can be any real number in [0,1].

```
<mutations>
   <mutation node="C">0.0</mutation>
</mutations>
```

We also provide the possibility of applying mutations in the XML definition. For now, the value can only be 0 or 1, but we are planning to include gradual mutations in future releases.

```
<parameters>
   <parameter name="$time_scale">0.2</parameter>
</parameters>
```

Finally, we are also providing the possibility of modify parameter values directly in the XML definition.

[Return to Table of Contents.]

4 PhysiBoSS API : MaBoSSIntracellular class

Here we expose the functions which are available in the MaBoSSIntracellular class, to query and control the state of the PhysiBoSS model. This is usually done in the custom code, for example in a custom implementation of update_phenotype. Here is a simple example of such an implementation :

```
void custom_update_phenotype( Cell* pCell, Phenotype& phenotype, double dt )
{
   // Checks if it is time to update the intracellular model
   if (pCell->phenotype.intracellular->need_update())
   {
      // Query the index of the density "Ainhib"
      static int ainhib_index = microenvironment.find_density_index( "Ainhib" );

      // Query the threshold from the user parameters to activate the node anti_A
      static double ainhib_threshold = parameters.doubles("ainhib_threshold");

      // Query the concentration of the internalized substrate Ainhib
      double ainhib_cell_concentration =
         pCell->phenotype.molecular.internalized_total_substrates[ainhib_index];

      // Activates the node anti_A if the concrentration of the internalized substrate
      // Ainhib is above the defined threshold
      pCell->phenotype.intracellular->set_boolean_variable_value(
         "anti_A",
         ainhib_cell_concentration >= ainhib_threshold
      );

      // Run the MaBoSS simulation
      pCell->phenotype.intracellular->update();

      // We define the prosurvival phenotype accoding to the boolean value of the
      // PhysiBoSS node C
      double prosurvival_value =
         pCell->phenotype.intracellular->get_boolean_variable_value("C") ? 1.0 : 0.0;

      // We query the indexes of the start and end phases of the live cell cycle model
      static int start_phase_index = phenotype.cycle.model().find_phase_index(
         PhysiCell_constants::live
      );
      static int end_phase_index = phenotype.cycle.model().find_phase_index(
         PhysiCell_constants::live
      );

      // We defined a new value for the cell cycle transition, based on the
      // prosurvival phenotype
      double multiplier = ( ( prosurvival_value * 20 ) + 1 );
```

```
      phenotype.cycle.data.transition_rate(start_phase_index,end_phase_index) =
         multiplier
         * phenotype.cycle.data.transition_rate(start_phase_index,end_phase_index
      );
   }
}
```

This example will activate the input node anti_A of the PhysiBoSS model if the internalized concentration of the sbstrate Ainhib is above the threshold defined in the user parameters. It will then run the MaBoSS simulation, and will modify the cell cycle transition rate according to the prosurvival phenotype defined by the node C.

Now let us look at the methods available in the PhysiBoSS API :

```
void update()
```

This method update the intracellular model : it runs the MaBoSS simulation for the defined time_step.

```
bool need_update()
```

This method checks if it is time to update the intracellular model, i.e. if the last simulation occurred more than a time_step ago.

```
bool has_variable(std::string name)
```

This method checks if a variable exists in the intracellular model.

```
bool get_boolean_variable_value(std::string name)
```

This method returns the boolean value of a given variable in the intracellular model.

```
void set_boolean_variable_value(std::string name, bool value)
```

This method set the boolean value of a given variable in the intracellular model.

```
double get_parameter_value(std::string name)
```

This methods returns the value of a parameter of the intracellular model.

```
void set_parameter_value(std::string name, double value)
```

This method set the value of a given parameter of the intracellular model.

```
std::string get_state()
```

This method returns a string with the current state of the intracellular model.

```
void print_current_nodes()
```

This method prints the current state of the intracellular model to the standard output.

[Return to Table of Contents.]

5 PhysiBoSS Makefile

The first part of the Makefile specific to PhysiBoSS is an header to setup the path of the include and library directories :

```
ifndef MABOSS_MAX_NODES
   MABOSS_MAX_NODES = 64
endif

MABOSS_DIR = addons/PhysiBoSS/MaBoSS/engine
CUR_DIR = $(shell pwd)

ifneq ($(OS), Windows_NT)
   LDL_FLAG = -ldl
endif

LIB := -L$(CUR_DIR)/$(MABOSS_DIR)/lib -lMaBoSS-static $(LDL_FLAG)
INC := -DADDON_PHYSIBOSS -I$(CUR_DIR)/$(MABOSS_DIR)/include -DMAXNODES=$(MABOSS_MAX_NODES)

ifeq ($(shell expr $(MABOSS_MAX_NODES) '>' 64), 1)
  LIB := -L$(CUR_DIR)/$(MABOSS_DIR)/lib -lMaBoSS_$(MABOSS_MAX_NODES)n-static $(LDL_FLAG)
endif
```

Here the only part that a user should modify is the value of the default MABOSS_MAX_NODES, according to the MaBoSS model size, to choose which version of libMaBoSS will be used. We provide libMaBoSS libraries for up to 64 nodes (default), 128 nodes, 256 nodes and 512 nodes.

```
PhysiBoSS_OBJECTS := maboss_network.o maboss_intracellular.o
...
ALL_OBJECTS := $(PhysiCell_OBJECTS) $(PhysiCell_custom_module_OBJECTS) $(PhysiBoSS_OBJECTS)
...
all: MaBoSS main.cpp $(ALL_OBJECTS)
   $(COMPILE_COMMAND) $(INC)  -o $(PROGRAM_NAME) $(ALL_OBJECTS) main.cpp $(LIB)
```

Then, we need to defined the PhysiBoSS_OBJECTS, and add them to the ALL_OBJECTS variable. Finally, we need to add MaBoSS to the all directive, to ensure that it will be installed if needed.

```
...
PhysiCell_cell.o: ./core/PhysiCell_cell.cpp
   $(COMPILE_COMMAND) $(INC) -c ./core/PhysiCell_cell.cpp
...
maboss_network.o: ./addons/PhysiBoSS/src/maboss_network.cpp
   $(COMPILE_COMMAND) $(INC) -c ./addons/PhysiBoSS/src/maboss_network.cpp

maboss_intracellular.o: ./addons/PhysiBoSS/src/maboss_intracellular.cpp
   $(COMPILE_COMMAND) $(INC) -c ./addons/PhysiBoSS/src/maboss_intracellular.cpp

custom.o: ./custom_modules/custom.cpp
   $(COMPILE_COMMAND) $(INC)  -c ./custom_modules/custom.cpp
```

Apart from the PhysiBoSS objects, both PhysiCell_cell and custom code will need to add the INC variable to add the paths to libMaBoSS which are necessary for compilation.

```
MaBoSS:
   ifeq ($(OS), Windows_NT)
      python beta/setup_libmaboss.py
   else
      python3 beta/setup_libmaboss.py
   endif
```

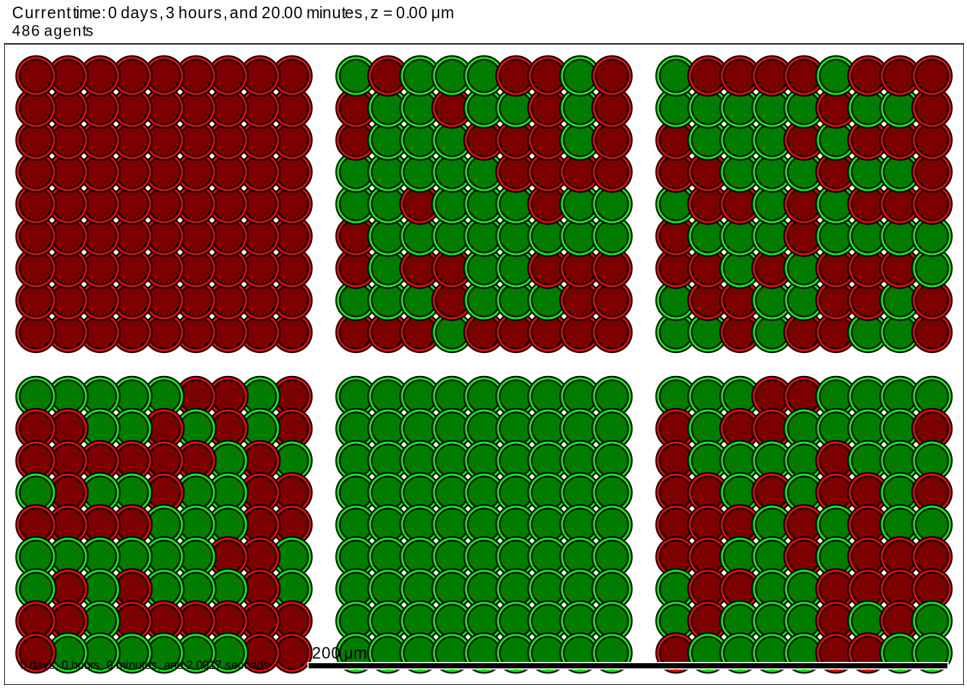
Finally, we add a directive for installing MaBoSS in the addons/PhysiBoSS folder, using the python script setup_libmaboss.py.

[Return to Table of Contents.]

6 Examples

6.1 PhysiBoSS Cell Lines model

PhysiBoSS comes with a very simple toy model. A population of cells has an intracellular model with three nodes (A,B,C) and starts with A active, B random and C inactive. Cells will turn green upon activation of node C.



This population is divided into 6 subpopulations :

- On the bottom left, the cells will need both A and B to be active in order to activate C. Since B is random, around half of the cells will activate B and then C, turning them green.
- On the bottom middle, we modified the boolean model : cells will need either A or B to activate C. Since A is active in all cells, all of them will activate C and turn green.
- On the top left, we will mutate the node C to inactive (representing a gene knock-out, for example), so all cells will stay red. We can force any node to be active or inactive to simulate pharmaceutical interventions.
- On the top middle, we will increase the speed of activation for both B and C : cells will reach steady state faster. Activation and inactivation speed can be controlled by individual parameters, and can also be modified during the simulation.
- On the bottom right, we will modify this parameter during the simulation : Starting at 0 (B won't activate), the activation speed will set to 0.1 at t=100min. Cells will then start the activation of B and then C.

- On the top right, we will change the time scale of the whole MaBoSS model : time will go faster, and cells will reach steady state faster. This is particularly useful to combine existing models, where time scales are not necessarily equal (or in the same units).

7 Future

Several features are planned for upcoming PhysiBoSS releases:

1. Simplifying the configuration of the links between PhysiBoSS and PhysiCell, using an automated mapping.
2. Adding new options to define the inheritance of MaBoSS state at cell division.

[Return to Table of Contents.]

8 Acknowledgements

We thank Paul Macklin and Randy Heiland for their help in the creation of this addon.

[Return to Table of Contents.]

Bibliography

[1] A. Ghaffarizadeh, R. Heiland, S. H. Friedman, S. M. Mumenthaler, and P. Macklin. PhysiCell: an open source physics-based cell simulator for 3-D multicellular systems. *PLoS Comput. Biol.*, 14(2): e1005991, 2018. doi: 10.1371/journal.pcbi.1005991. URL http://dx.doi.org/10.1371/journal.pcbi.1005991.

[2] G. Letort, A. Montagud, G. Stoll, R. Heiland, E. Barillot, P. Macklin, A. Zinovyev, and L. Calzone. Physiboss: a multi-scale agent-based modelling framework integrating physical dimension and cell signalling. *Bioinformatics*, 35(7):1188–1196, 2019. doi: 10.1093/bioinformatics/bty766. URL https://academic.oup.com/bioinformatics/article/35/7/1188/5087713.

[3] M. Ponce-de Leon, A. Montagud, V. Noël, G. Pradas, A. Meert, E. Barillot, L. Calzone, and A. Valencia. Physiboss 2.0: a sustainable integration of stochastic boolean and agent-based modelling frameworks. *bioRxiv*, pages 2022–01, 2022. doi: 10.1101/2022.01.06.468363. URL https://www.biorxiv.org/content/10.1101/2022.01.06.468363v3.