Installation guide for MaBoSS, NeKo and PhysiCell/PhysiBoSS

Marco Ruscone

February 10, 2025

The following guide have been written specifically for the Hands-On session organized at the PRIMe Institute in Osaka. The content of this guide has partially been taken from [1, 2].

Contents

S1 Installing MaBoSS

MaBoSS is a C++ software that features a Python interface. The most straightforward way to install MaBoSS is through package managers such as anaconda or mamba (information on how to install it at the following link: `https://docs.anaconda.com/anaconda/install/`). If you have anaconda installed then you can install MaBoSS through:

```
conda install -c colomoto pymaboss
```

or alternatively with pip:

```
pip install maboss
python -m maboss_setup
```

Mind that if you install MaBoSS through pip, it will be necessary to compile separetely the C++ code to generate the MaBoSS executable that will be called though python by doing "python -m maboss_setup".

Once you have installed MaBoSS, you can verify its integrity by opening a python shell and running:

```
import maboss
```

If you get no error when importing maboss, then the installation was successful.

Further information about MaBoSS installation can be found here: `https://maboss.curie.fr/`.

S2 Installing NeKo

NeKo is a python package with many dependencies, in particular Omnipath `https://github.com/saezlab/omnipath` and Pypath `https://github.com/saezlab/pypath`. As per MaBoSS, I recommend creating an empty virtual environment using anaconda or mamba, and then installing NeKo using pip:

```
pip install nekomata
```

Once the package is installed, its dependencies can be installed through git:

```
pip install -r https://raw.githubusercontent.com/sysbio-curie/Neko/main/
    requirements.txt
```

If during the installation you encounter problems with the installation of Graphviz, you could be missing basic Graphiz installation on your machine. You can install it on Linux system with the following command:

```
sudo apt-get install python3-dev graphviz libgraphviz-dev
```

or on Mac system:

```
brew install python3-dev graphviz libgraphviz-dev
```

Once you have installed NeKo, you can verify its integrity by opening a python shell and running:

```
from neko.core.network import Network
from neko.inputs import Universe
```

If you get no error when importing the Network and Universe class (the main objects of the package), then the installation was successful. Note that the first time, importing NeKo will take much time. This is because Pypath and Omnipath will download many cache files that will allow importing and loading databases much faster the next time you will use the package.

Further information about NeKo installation can be found here: `https://github.com/sysbio-curie/Neko`.

A series of notebook can guide you through the main steps of using NeKo in the main documentation: `https://sysbio-curie.github.io/Neko/`.

NeKo is still in development so it is possible to encounter bugs and other problems during the installation. Please do not hesitate to report these bugs in the github page: `https://github.com/sysbio-curie/Neko/issues`.

S3 Setting up a PhysiBoSS project

S3.1 Installation of PhysiCell and PhysiCell-Studio

PhysiCell and PhysiCell-Studio can be installed using the following commands:

```
git clone https://github.com/MathCancer/PhysiCell
git clone https://github.com/PhysiCell-Tools/PhysiCell-Studio.git
```

Two folders should be created in a working directory: PhysiCell and PhysiCell-Studio.

S3.2 Installation of the environment for PhysiCell-Studio

To ensure the PhysiCell-Studio work properly, an environment that contains all the dependencies of PhysiCell-Studio should be created and activated. This can be done through, for example, the anaconda package (`https://www.anaconda.com/`). To install the environment provided with PhysiCell-Studio, navigate into the folder from a terminal and use the following commands:

```
cd PhysiCell-Studio
conda env create -f environment.yml
cd ..
```

This command will start the creation of the environment and the download of the dependencies. To activate the environment simply execute:

```
conda activate studio
```

Once the set-up of PhysiCell-Studio is done, we can move to the PhysiBoSS folder to set up a model.

```
cd PhysiBoSS
```

S3.3 Using packaged models

We are providing, in the latest release, packaged models with pre-compiled binaries for Linux, Mac, and Windows to run the different models described in this article (and that could be compatible with many more PhysiCell models), to simplify the steps to reproduce our work. These binaries are still highly experimental and might not work on some machines, in which case you need to follow the instructions in the following section to load the model and compile yourself PhysiBoSS.

To download the packaged model for the PhysiBoSS template, and start building a PhysiBoSS model, you can run the following Python command from the PhysiCell root directory :

```
python beta/download_binary.py template_BM
```

This will recognize your operating system, download the appropriate binary, and put it in your root directory under the name `project`. It will also download associated config files and source files in case you want to modify the code

and/or recompile the binary. The template_BM model is the starting point for this tutorial. However, if you want to directly run the examples of this tutorial, you can also use this script to download them.

To download the packaged model for the three main examples, you can run :

```
python beta/download_binary.py physiboss-tutorial
```

If you want to run the cancer invasion model, you can run the following command :

```
python beta/download_binary.py physiboss-tutorial-invasion
```

In Linux, these binaries are compiled for running with GLIBC 2.32 to 2.34 (Ubuntu 22 LTS). A system with older versions of GLIBC will produce an error:

```
libc.so.6: version ‘GLIBC_2.32‘ not found.
```

In this case, please compile locally your project using the following section.

In recent MacOS versions, these binaries are by default prevented from running. After a first try to run it, you need to explicitly allow them to run in the Settings > Privacy and security options, where a notification will appear and an option will be available to authorize them. In some cases, binaries will fail to run, without any specific error message. In this case, please compile locally your project using the following section.

S3.4 Loading a model into the directory structure

PhysiCell models provided by the development team are archived in the folder `sample_projects` and `sample_projects_intracellular`. To use them, we first need to load them into the proper directory structure by using the command :

```
make template_BM
```

Here we are loading the `template_BM` project, which is the template for PhysiBoSS projects. This template can be modified to build new PhysiBoSS models. Once loaded, the directory structure should look like the one in S1, except for the `project` binary that we will explain in the next two sections.

S3.5 Compilation of the PhysiBoSS project

We can set a model starting from the template_BM provided by PhysiCell. This template already includes the function necessary to incorporate a Boolean network in each cell type. In a terminal, we use the following command to compile the template_BM project:

```
make -j
```

If this is the first time compiling a project with PhysiBoSS, it will automatically download the required library to run MaBoSS inside PhysiCell.

It is important to note that the template_BM model can only run MaBoSS models up to 256 nodes. To increase this value, you can modify the value of the `MABOSS_MAX_NODES` variable in the Makefile (line #17).

By default, the compilation of the template model will produce a binary called `project`. If you want to change this, you need to edit the Makefile and change the value of the variable `PROGRAM_NAME` (line #2).

Once the project is compiled, it is necessary to copy the .bnd and .cfg files in the folder boolean_network, located inside the config folder. The structure should look like in Figure S1.

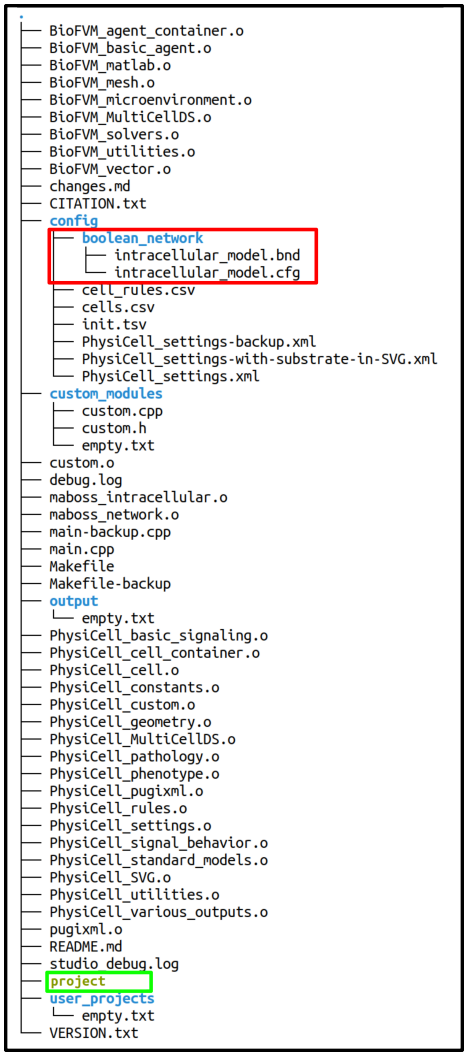

Figure S1: Tree structure of the PhysiBoSS folder once compiled the template. The rectangle in red indicates the position of the intracellular files, while in green the name of the executable.

If everything is set correctly, we are ready to open PhysiCell Studio and

start building the model.

S3.5.1 Troubleshooting

Linux Some Linux installations will come with only minimal packages, lacking the dependencies of PhysiBoSS. On Ubuntu systems, you can install these dependencies using :

```
apt install gcc g++ make flex bison python3
```

On some Linux installations, the available GCC version will be older than the one MaBoSS libraries were built with, which might lead to this type of error message :

```
undefined reference to ‘std::__throw_bad_array_new_length()‘
```

In this case, you need first to recompile the MaBoSS library using the locally installed GCC :

```
make Compile_MaBoSS
make
```

MacOSX MacOSX comes by default with a version of Clang C++ compiler which is not compatible with OpenMP, the parallel computing library used by PhysiCell. This will lead to this error message :

```
error: unsupported option ’-fopenmp’
```

To solve this, you need to install the GCC C++ compiler using Brew, and specify that you want to use this compiler when compiling PhysiCell :

```
brew install gcc@11
make PHYSICELL_CPP=g++-11
```

In some cases, there will be an incompatibility between the locally installed GCC version and the GCC version used to compile MaBoSS library. This will result in this type of error message :

```
ld: symbol(s) not found for architecture x86_64
```

In this case, you need first to recompile MaBoSS library using the locally installed GCC:

```
make Compile_MaBoSS PHYSICELL_CPP=g++-11
make PHYSICELL_CPP=g++-11
```

This supposes that you have GCC 11 installed on your machine. If it is another version, replace `g++-11` by `g++-<version>`.

Windows PhysiBoSS can be compiled on Windows using MinGW-w64 packages. To install them, you first need to download and install MSYS2, available at `https://msys2.org`. Then, after installing MSYS2, you need to open an MSYS2 Command Prompt and run the following command :

```
pacman -S mingw-w64-x86_64-binutils mingw-w64-x86_64-gcc mingw-w64-x86_64-headers
    -git mingw-w64-x86_64-gcc-libs mingw-w64-x86_64-libwinpthread-git mingw-w64-
    x86_64-winpthreads-git mingw-w64-x86_64-lapack mingw-w64-x86_64-openblas
    mingw-w64-x86_64-libxml2 mingw-w64-x86_64-bzip2 mingw-w64-x86_64-python git
    make flex bison
```

This will install the necessary dependencies. You can then use the commands mentioned in the compilation section in a MinGW-w64 Command Prompt.

A more detailed installation tutorial for Windows is available at: https://github.com/physicell-training/ws2023/blob/main/setup/PhysiCell/ws2023_Windows_setup.pdf.

S3.6 Using PhysiBoSS with PhysiCell Studio

PhysiCell Studio supports the development of models using PhysiBoSS. It provides the user with all the necessary tools to set an intracellular model and connect it to the agent's signals/behaviors. We can open PhysiCell Studio from the terminal using the following command:

```
python ../PhysiCell-Studio/bin/studio.py \
       -e project -c config/PhysiCell_settings.xml
```

Not that here, `project` refers to the name of the compiled executable, and `config/PhyiCell_settings.xml` refers to the PhysiCell XML settings files describing the model. These values can be changed depending on the specific model.

If no errors occur, the welcome page of PhysiCell should show up as in Figure S2.

Most of the tabs in PhysiCell Studio offer functionalities for editing the model (Config Basics, MicroEnvironment, Cell Types, User Params, Rules, ICs). The last two tabs are dedicated to running the model (Figure S3) and visualizing the results (see next section).

S3.7 Visualizing results in PhysiCell Studio

PhysiCell Studio allows the visualization of the simulation results generated by PhysiCell in a simple, intuitive graphical interface. It allows the visualization of any time points, play the simulation as a video, and choose the element of your model you want to emphasize.

By default, its visualization only shows cells, colored by cell type. This is the default mode for the SVG files generated by PhysiCell (Figure S4A).
A useful additional visualization is the plotting of the substrate concentration in space (Figure S4B). This allows the user to see the gradient and is extremely practical when defining the parameters of diffusion/decay of your substrate. If you want to look deeper into the state of each individual cell, you can select the .mat representation, which allows you to look for many different quantities in your cell and choose them to color the cells. In Figure S4C you can see an example of this, with the cells being colored according to their current phase

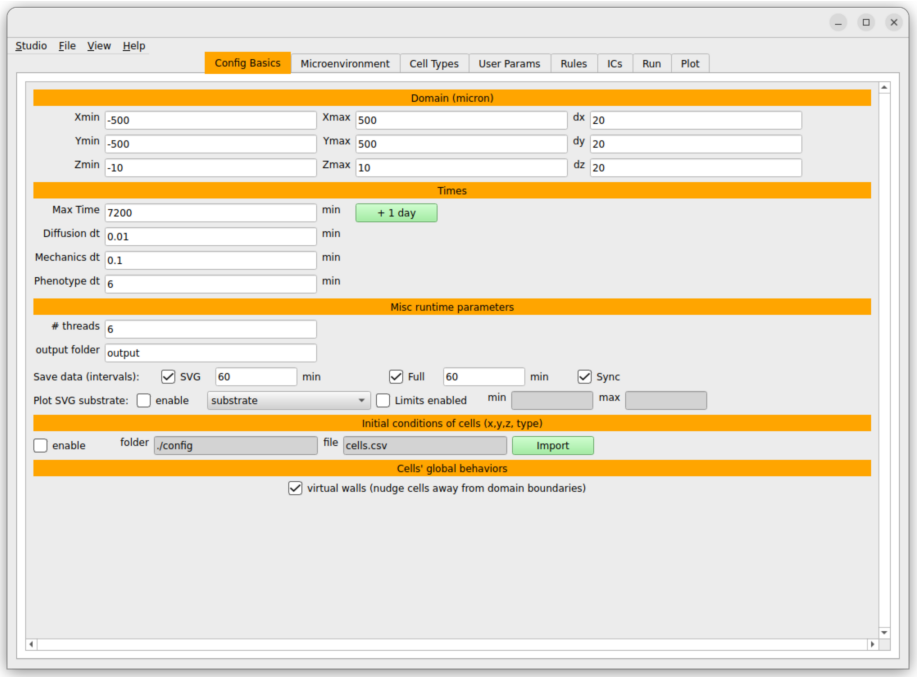

Figure S2: Welcome page of PhysiCell Studio

(live, apoptotic, or necrotic).

Moreover, with PhysiBoSS we added another visualization model, which uses the state files PhysiBoSS is generating, saving the state of the Boolean network for each cell at each time point. With this, you can choose any cell type, and within its network any Boolean node, and represent its activity in each cell by coloring it in green/red (Figure S4D). Note that only the nodes defined as output (non-internal) in the MaBoSS model CFG file will be included in the list of nodes that can be used.

Finally, PhysiCell Studio allows us to print the population size in time, according to different aspects of the model (cell types, current phase, etc.) (See Figure S5A). With PhysiBoSS we also included this possibility, by plotting the trajectory of the population size for each Boolean state (Figure S5B). Note that with too many nodes declared as output nodes, the state space will explode and this option will not be usable.

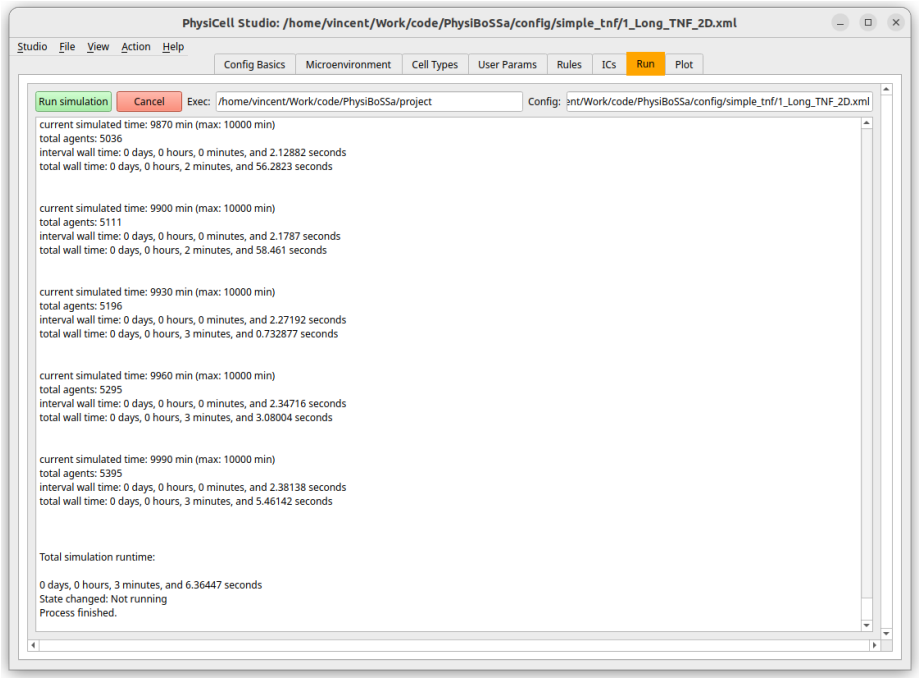

Figure S3: Running a model in PhysiCell Studio. After choosing the proper binary and XML settings file to execute, you can run the model by pressing the top left button. Logs of the simulation will appear in this tab.

References

[1] Ruscone M, Checcoli A, Heiland R, Barillot E, Macklin P, Calzone L, et al. Building multiscale models with PhysiBoSS, an agent-based modeling tool. ArXiv. 2024 Jun:arXiv:2406.18371v1.

[2] Stoll G, Caron B, Viara E, Dugourd A, Zinovyev A, Naldi A, et al. MaBoSS 2.0: an environment for stochastic Boolean modeling. Bioinformatics. 2017 Jul;33(14):2226-8. Available from: `https://doi.org/10.1093/bioinformatics/btx123`.

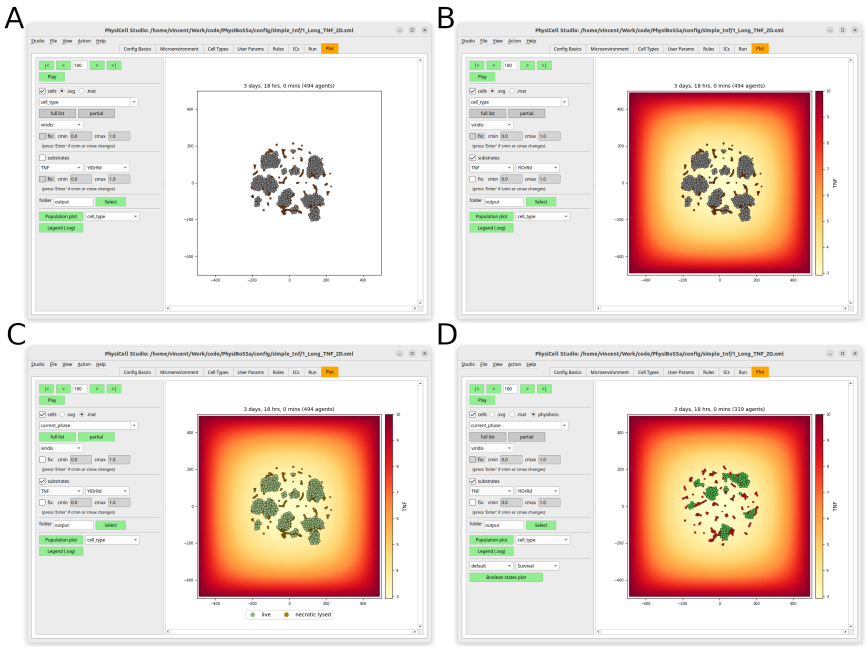

Figure S4: Visualizing a simulation result in PhysiCell Studio. In the *Plot* tab, you can look at an existing simulation folder, and visualize most components of the model. A: Default visualization, using the SVG images generated by PhysiCell. B: Additional visualization of the substrate concentrations. C: Visualizing the content of the .mat files, containing more information about the PhysiCell model. D: Visualizing the content of the PhysiBoSS state files, by coloring a cell according to the activity of a particular node.

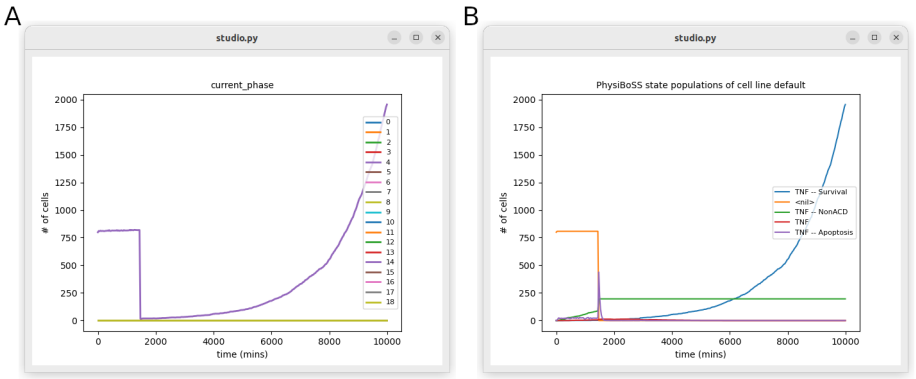

Figure S5: Visualizing the time-series of population size by current phases (A) and by Boolean state (B).