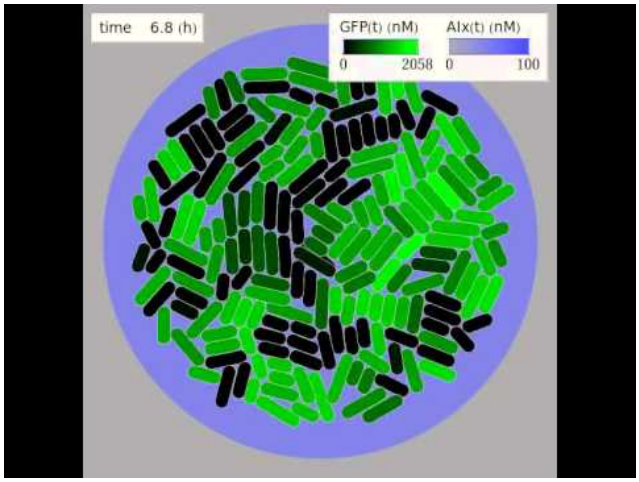
COLONY User Guide



June 26, 2016

Contents

1 Introduction

COLONY is a simulation software developed by Marc Weber and Javier Buceta with the aim of studying the effects of gene expression noise in the variability of the quorum sensing activation in a population of bacterial cells. The code was written in C++ and relies on Wolfram Mathematica to compute the input files defining the biological network and all the parameters of the model. Its key capabilities are the simulation of the growth and division of a cell population which communicate through diffusible molecules (quorum sensing), by also taking into account the stochastic dynamics of the intracellular regulatory network. The latter can be achieved by the use of either the Gillespie algorithm or the chemical Langevin algorithm.

The software was built with the aid of Qt Creator and the Qt library and can be run on Linux, Windows and Mac. The simulations can run either from the command line or the Graphical User Interface (GUI).

2 Software Set-up and Compilation

2.1 Minimum Requirements

- Wolfram Mathematica (version 10 or later)
- Qt Creator (version 3.5 or later)
- Qt Libraries (mostly required for compiling the GUI)

2.2 Recommendations for compilation

For the code compilation additional libraries need to be installed. In Ubuntu these can be easily downloaded as packages from the Ubuntu Software Center. The required libraries are:

- **Blitz++ version 0.10** (used throughout the code to define arrays)

  Ubuntu packages: libblitz0ldbl and libblitz0-dev

  There is an important difference in the array reading method from the input file between version 0.9 and 0.10

- **Boost system and Boost filesystem, version 1.54** (Used in the Output class)

  Ubuntu packages: libboost-system and libboost-system-dev libboost-filesystem and libboost-filesystem-dev

- **GSL library, version 1.16** (Used in the random number generator)

  Ubuntu packages:libgsl0ldbl and libgsl0-dev

- **ODE library, Open Dynamics Engine, version 2:011.1** (Used in the computation of the spatial movement of the cells in the growing colony. Even if the GUI option is not set, the spatial dynamics of the cells can still be computed)

  Ubuntu packages: libode1 and libode-dev

- **libstdc++** (additional runtime library for C++ compiled with the GNU compiler)

  Ubuntu packages: libstdc++6 and libstdc++-4.8-dev

- **ATLAS library** (optimized BLAS library from ATLAS (Automatically Tuned Linear Algebra Software) used in the matrix-vector multiplication in the Langevin algorithm to speed up this computation step. This library has to be compiled for each architecture/machine and result in optimized ATLAS, CBLAS and LAPACK libraries. GFORTRAN and f77BLAS may also be needed.)

  or

- **OpenBLAS** (Supposedly as fast as the ATLAS library but easier to install with no need to compile for specific architecture)

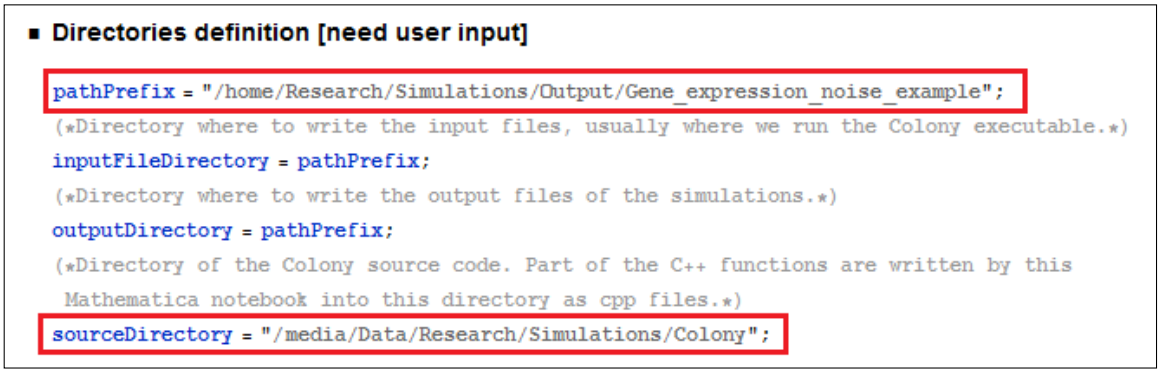
  Ubuntu packages: libopenblas-base and libopenblas-dev

3 Model Creation

COLONY receives the model information from the Mathematica notebook (xxx.nb) in order to generate the input files and update accordingly the C++ files for the propensity functions and increments. The parameters, species and reactions are therefore defined by the user in the appropriate sections of this notebook.

3.1 Directories

In the first section of the notebook the directories where the input and output files will be written are imputed by the user. The first path that should be defined is "pathPrefix" and it is the directory where the input files will be written by the notebook. This directory should be also where the COLONY executable is located. The second path that needs user input is "sourceDirectory", the directory where the COLONY source code is located along with the functions that will be updated by the Mathematica notebook.



The users should take into account that directory paths are defined differently in Windows and Linux.

3.2 Cell growth, cell volume and area

The next step is to choose if the simulation will involve a growing cell colony or a fixed number of cells. This is done by setting the "cellgrowth" entry as "true" or "false". Additionally, the correction of the cell volume at the beginning of the cell cycle is made. This happens in the case of exponential growth, in order to correctly compare the deterministic to the stochastic model. Therefore the initial cell volume V_0 is replaced by the average cell volume over a cell cycle $V_1 = \frac{1}{ln(2)} V_0$.

Definition of the system of reactions [need user input at volume correction]

- **Set cell growth, cell volume and area**

```
cellgrowth = False;
```

First the total volume is determined based on the cell density and the number of cells.

$$Vtot = \frac{nCells}{nconc0\ (*cells/\mu m^3*)}\ (*\mu m^3*);$$

Then the volume ratio Vx is determined based on Vtot, V0 and nCells, inverse of r.

$$Vx = \frac{Vtot - nCells\ V0}{V0};$$

$$Vext = Vx\ V0;$$

V0 is replaced by the average of the cell volume over a cell cycle in the case of the exponential growth, $V1 = \frac{1}{Log[2]}\ V0$,

in order to compare correctly the deterministic model (with fixed cell volume) to the stochastic model (with dynamic growth).

$$nmolesV1 = \frac{1*^9}{6.02214*^23} * \frac{1}{\frac{V0}{Log[2]} * 1*^-15};$$

Additional unit conversions are made in this section (i.e. from number of molecules to concentration). The default concentration units are set in nanomolar (nM), but the conversion factors can be amended accordingly to define the input and output in different units.

Conversion coefficient molecules per cell -> nM

$$nmolesV0 = \frac{1*^9}{6.02214*^23} * \frac{1}{V0 * 1*^-15};$$

$$nmolesVext = \frac{1*^9}{6.02214*^23} * \frac{1}{Vext * 1*^-15};$$

```
(*Conversion function cell density cells/mL <--> number of cells*)
nconc[n_] := n / (Vtot * 1*^-12) /. par;(* n → #cells/mL *)
concn[x_] := (Vtot * 1*^-12) * x /. par;(* #cells/mL → n *)
(* #cells/mL → #cells*)
```

3.3 Parameters

In the next section, the model parameters are defined by the user. The parameters that remain constant throughout the simulation should be written as the symbol of the parameter, followed by an arrow and the value of the parameter (i.e α -> 0.001).

Parameters [need user input]

```
par = {
  (*Reaction rates are defined in terms of concentration nM and dimensionless time units.*)
  (*See below definition of the system of reaction for the meaning of the parameters*)
  α → 0.001, β → 0.25, γ → 0.8
 }
```

The input parameters that change during the simulation or between different simulations should be defined as global parameters. For example, the value of two parameters *p1* and *p2* could change at specific time points of the simulation. The symbols of the parameters are defined in the table "globalParList", the time points the values change are defined in the "simTimePoints" and the values themselves in the "globalParSet".

```
globalParList = {{p1}, {p2}};
simTimePoints = {10, 15, 50};
globalParSet = {{{5., 10.}, {6., 12.}, {7., 9.}}, {{3., 8.}, {3.2, 7.}, {3.5, 6.}}};
```

In another case, the value of some parameters could remain the same for the whole simulation, but change between different simulations (i.e. in the case of ensemble modelling). In this case, the different parameters sets can be defined in a .csv file (which will be located in the same directory with the notebook) and imported into the notebook.

p1	p2	p3	p4
1.1	2.1	3.1	4.1
1.2	2.2	3.2	4.2
1.3	2.3	3.3	4.3

(a) The values of parameters p1, p2, p3, p4 for each simulation are stored in the "par-list.csv" file

```
globalParList = Partition[Prepend[{p1, p2, p3, p4}, n], 1]
importedParTable = Import["par_list.csv"]
globalParSet = Partition[Join[Partition[Range[Length[importedParTable]], 1], importedParTable, 2], {1}]
```

(b) The "par-list.csv" file is imported in the notebook

3.4 Species

The model species are imputted in a table separately for each compartment. The cell species are defined as "speciesCell" and the environment (milieu) species are defined as "speciesMilieu".

List of species [need user input]

```
speciesCell = {U, V};
speciesMilieu = {Ue, Ve};
```

Additionally, an array defining the positions of the genetic species in the overall stoichiometric table is defined, in order to differentiate them from the proteins and other biological species. In this way, it is specified which species are halved and which species are copied during a division event.

Commands for defining the genetic species

```
dnaCell = Table[0, {nSCell}, {nSCell}];
dnaCell[[pos[Gene1, speciesCell], pos[Gene1, speciesCell]]] = -1;
dnaCell[[pos[Gene2, speciesCell], pos[Gene2, speciesCell]]] = 1;
dnaCell[[pos[Gene2, speciesCell], pos[Gene1, speciesCell]]] = 1;
dnaCell[[pos[Gene2, speciesCell], pos[Molecule1, speciesCell]]] = 1;
dnaCell[[pos[Gene3, speciesCell], pos[Gene3, speciesCell]]] = 1;
dnaCell[[pos[Gene3, speciesCell], pos[Gene1, speciesCell]]] = 1;
dnaCell[[pos[Gene3, speciesCell], pos[Molecule1, speciesCell]]] = 1;
dnaCell[[pos[Gene4, speciesCell], pos[Gene4, speciesCell]]] = 1;
dnaCell[[pos[Gene4, speciesCell], pos[Gene1, speciesCell]]] = 1;
dnaCell[[pos[Gene4, speciesCell], pos[Molecule1, speciesCell]]] = 2;
```

Position of genetic species in stoichiometric matrix

dnaCell:

```
{{0 0 0 0 0 0 0 0 0 0 0 0 0},
 {0 0 0 0 0 0 0 0 0 0 0 0 0},
 {0 0 0 0 0 0 0 0 0 0 0 0 0},
 {0 0 0 0 0 0 0 0 0 0 0 0 0},
 {0 0 0 0 0 0 0 0 0 0 0 0 0},
 {0 0 0 0 0 0 0 0 0 0 0 0 0},
 {0 0 0 0 0 0 0 0 0 0 0 0 0},
 {0 0 0 0 0 0 0 -1 0 0 0 0 0},
 {0 0 0 0 0 0 1 1 1 0 0 0 0},
 {0 0 0 0 0 0 1 1 0 1 0 0 0},
 {0 0 0 0 0 0 2 1 0 0 1 0 0},
 {0 0 0 0 0 0 0 0 0 0 0 0 0},
 {0 0 0 0 0 0 0 0 0 0 0 0 0}}
```

3.5 Reactions

The reactions that form the regulatory/ signalling system within the cell or in the external environment are written as a list with the following format:

{{reactants list}, {products list}, {1 if the forward reaction is active or 0 if not, 1 if the backward reaction is active or 0 if not}, {rate of forward reaction, rate of backward reaction}}

The stoichiometric coefficient can be introduced by repeating the species name, as in {{A,A,A,B},{C},{1,1},{kf,kb}}

The reactions of molecule transport across the membrane (diffusion) should be written in the same way. For instance, the following reactions:

Inside the cell: DNA $\rightarrow$ DNA + mRNA (DNA transcription with rate kf), DNA + AI $\rightleftarrows$ DNAAI (Autoinducer molecule AI reversibly binding to DNA and forming a complex, with forward rate Kbind and reverse rate Kunbind)

In the external environment: $AI_e \rightarrow \varnothing$ (Degradation of the external autoinducer with rate AIdeg)

Between cell and external environment: AI $\rightleftarrows$ AI_e (Diffusion of the autoinducer from the cell to the environment with forward and reverse rate Dx)

are imputed in the notebook in the following format:

```
reacCell = {
   {{DNA}, {DNA, mRNA}, {1, 0}, {kf, 0}},
   {{DNA, AI}, {DNAAI}, {1, 1}, {Kbind, Kunbind}}
  };

reacMilieu = {AIe} {} {1, 0} {AIdeg, 0};

reacCellMilieu = {AI} {AIe} {1, 1} {Dx, Dx};
```

3.6 Initial Conditions

The initial conditions for all species of the cell and of the environment are defined in tables. For the internal species the table is named "x0Cell" and for the external it is called "x0Milieu". An example is presented in the following figure where the cellular species are defined as a table of zeros with length "nSCell" (number of species in 1 cell). The initial values for the external species are imputed in a separate table.

Initial conditions

```
x0Cell = Table[0, {nSCell}];
x0Cell[[pos[P00, speciesCell]]] = 1 * nmolesV1;
x0Cell[[pos[Nconc, speciesCell]]] = nconc0;
x0Milieu = {0};
x0 = Join[x0Cell, x0Milieu];
```

Definition of table of zeros of lenght "nSCell"

Change the values of the two species from zero to nmolesV1 and nconc0 respectively

Definition of initial conditions of external species, as separate values.

Joining of the two tables to form the table of total initial conditions.

Output

$$x0 : \left\{0, 0, 0, 0, 0, 0, 0, \frac{1.151}{V0}, 0, 0, 0, 0, nconc0, 0\right\}$$

$$x0Cell : \left\{0, 0, 0, 0, 0, 0, 0, \frac{1.151}{V0}, 0, 0, 0, 0, nconc0\right\}$$

$$x0Milieu : \{0\}$$

3.7 Choice of Algorithm and Experiment

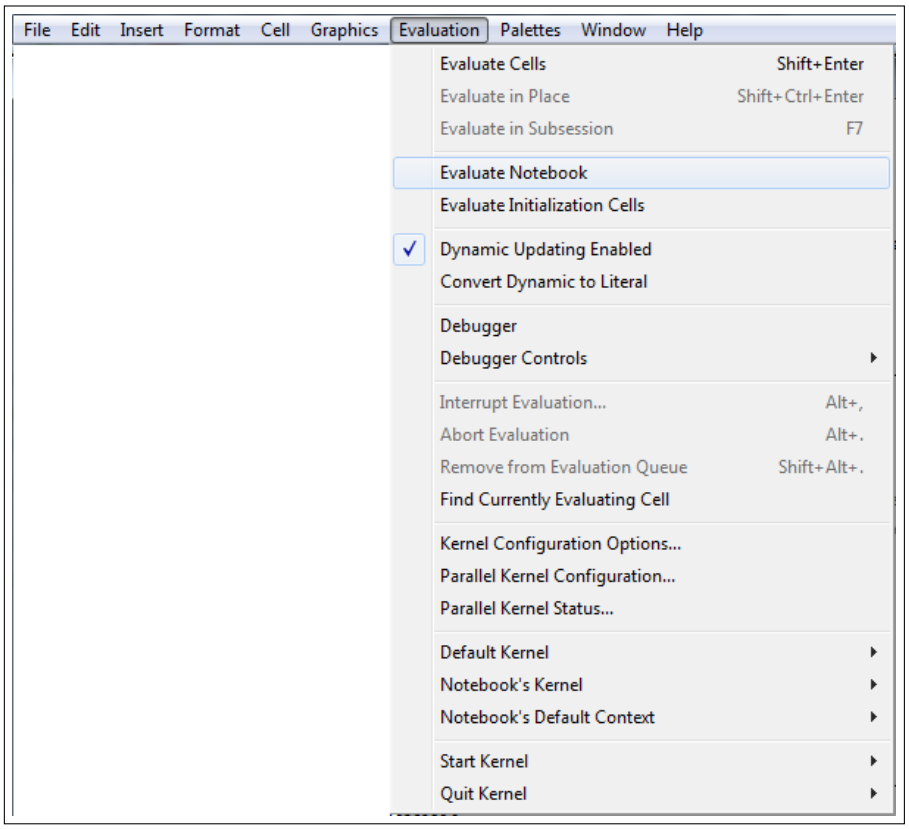
A choice of simulation algorithm can be made between Gillespie and Langevin by the following command:

```
(*Choose algorithm.*)
algorithm = "Langevin";
algorithm = "Gillespie";
```

4 Simulations

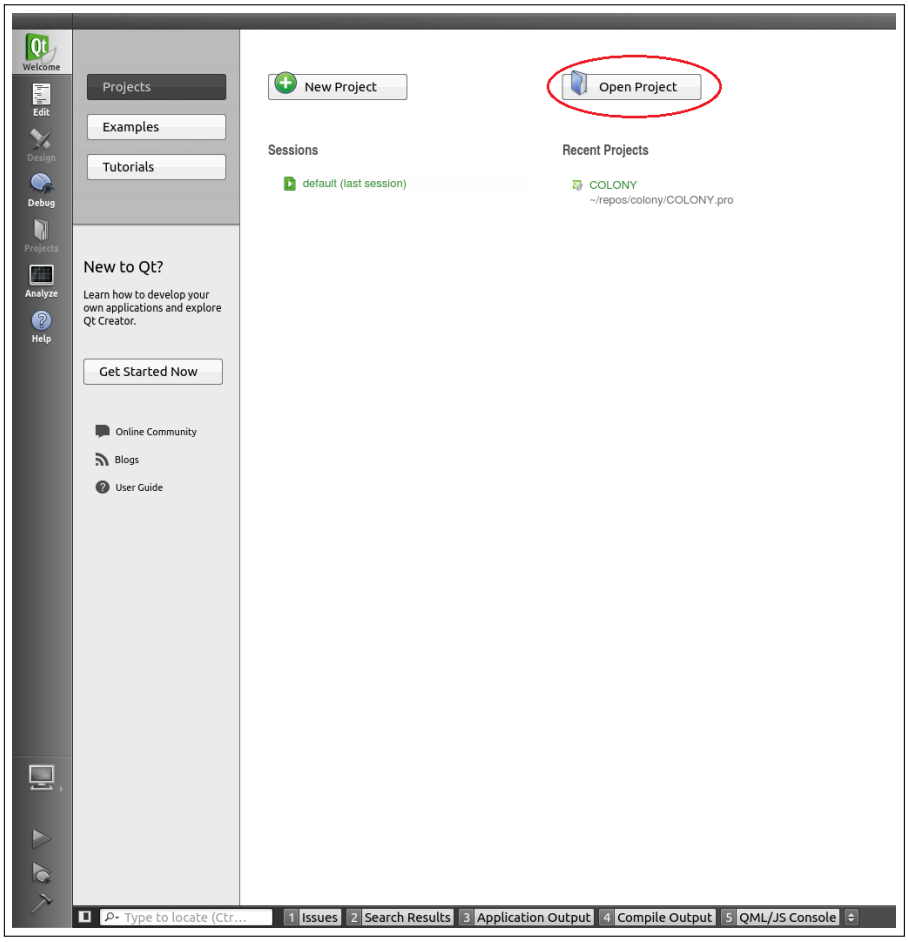
4.1 Setting up the simulations

In order to run the simulations, the input file needs to be created and some of the algorithm functions need to be updated. This can be done from the Mathematica notebook by clicking on the command "Evaluate Notebook" in the Evaluation tab.

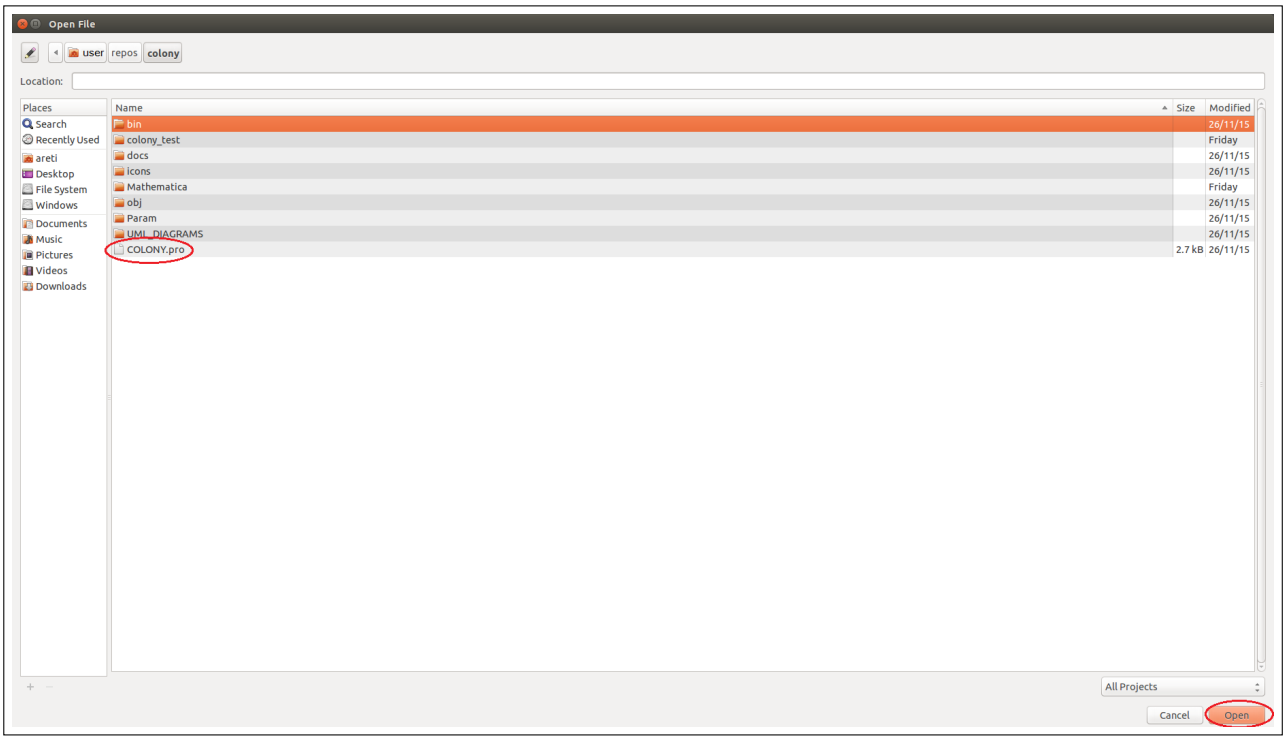


The next step is to compile and run the code from the Qt Creator. The input file must be located in the same directory with the executable "COLONY.pro".

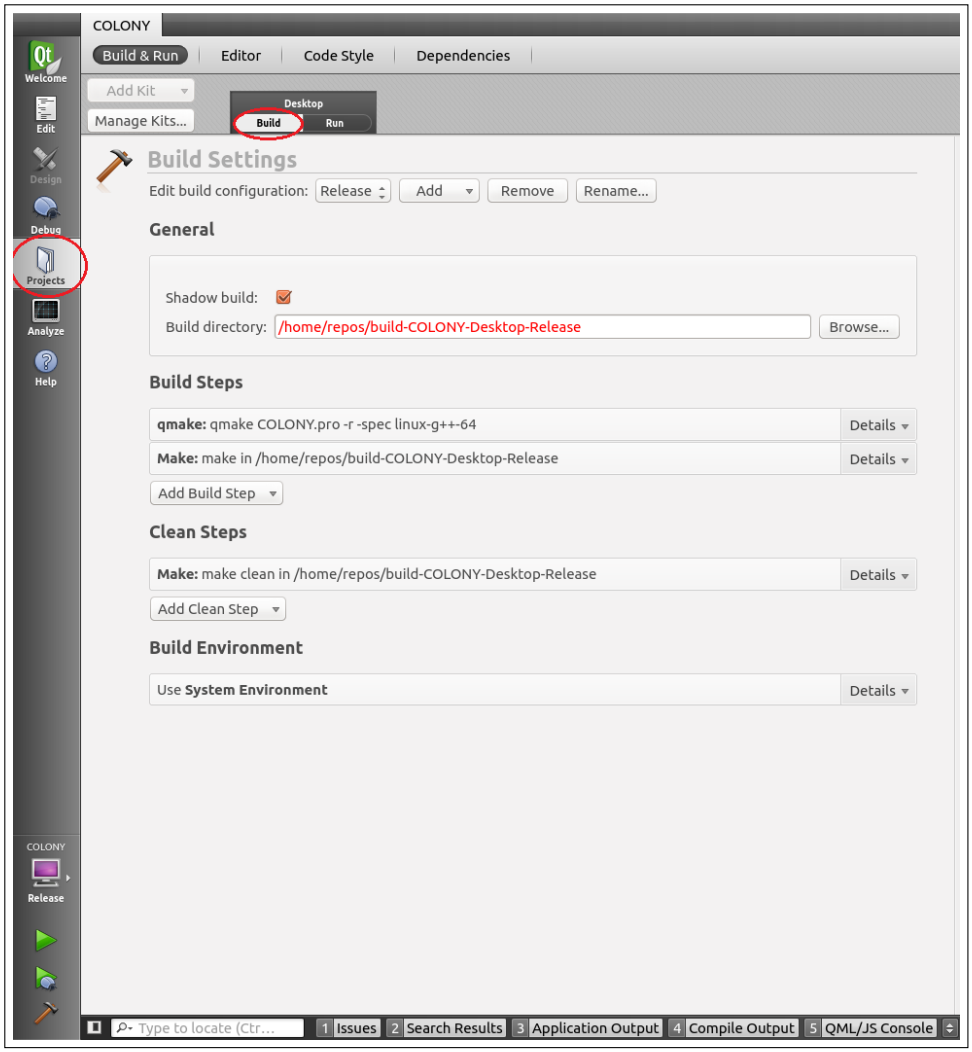
Open a new project in Qt creator as shown in the figure below.



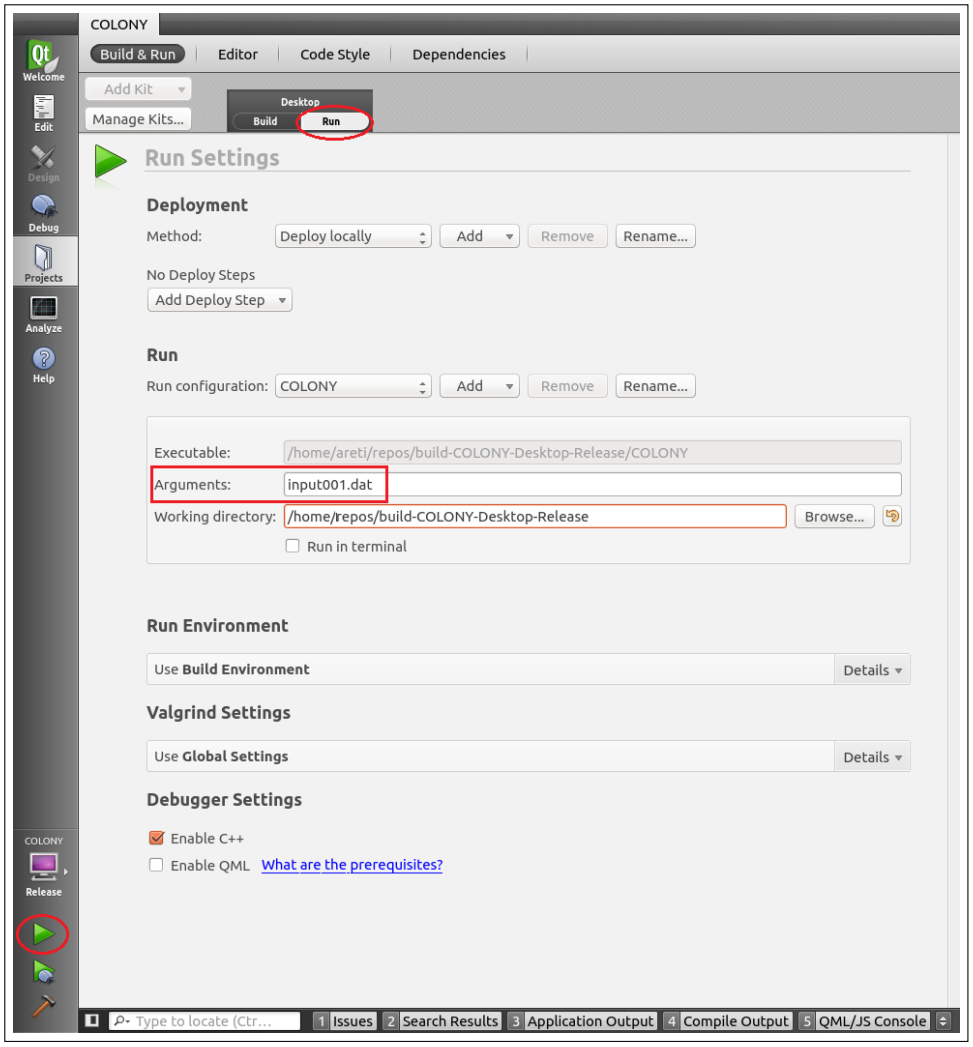
A window will open with a list of directories in the user's hard disc. Navigate to the folder where the executable file "COLONY.pro" is located and choose it from the list of files and directories. Click open to start the new project.



Go to the Projects tab on the left ribbon of Qt creator, choose the "Build" option from the desktop menu at the top of the Qt mask and define the working directory where the simulation results will be stored ("build directory").



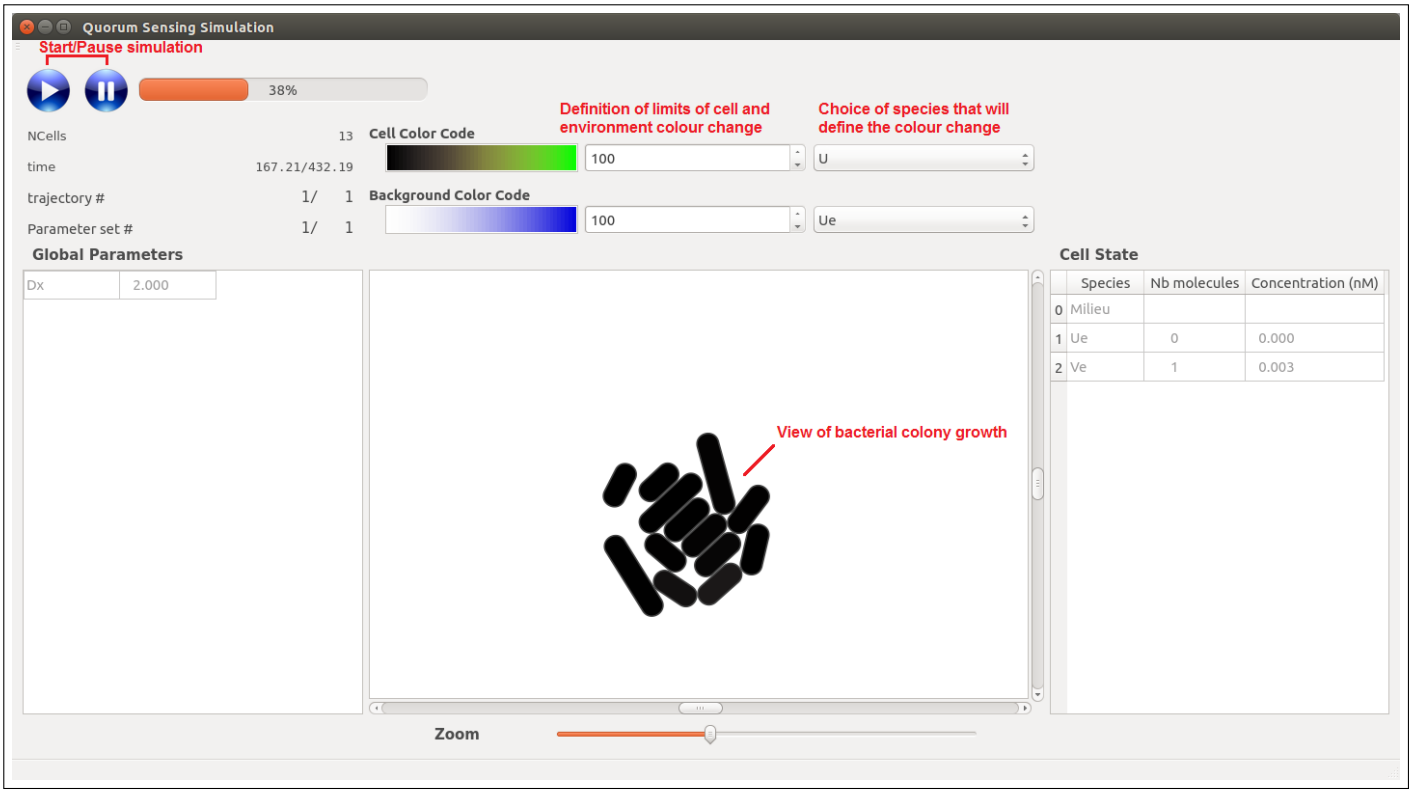
To start running the simulation, choose from the desktop menu the "Run" option and in the "Arguments" section write the name of the input file generated from the Mathematica notebook (input001.dat). Finally, click the green play button to run the simulations.



Avoid debugging from Qt Creator, but only run the simulations.

4.2 The Graphical User Interface

The graphical user interface (GUI) is launched automatically when running the simulation from Qt Creator. The user can start or pause the simulation at any point by clicking the "play" and "pause" buttons. The bacterial colony growth is observed in the main window and on the right the number of molecules and concentration for each species in the cell in the milieu is updated. The limits of specific species as well as the species themselves that define the change in the colours of the cells and of the external background can be changed from the bars "Cell Color Code" and "Background Color Code".


Quorum Sensing Simulation
Start/Pause simulation
38%
NCells
13
time
167.21/432.19
trajectory #
1/ 1
Parameter set #
1/ 1
Cell Color Code
Definition of limits of cell and environment colour change
Choice of species that will define the colour change
100
U
Background Color Code
100
Ue
Global Parameters
Dx
2.000
View of bacterial colony growth
Cell State
Species
Nb molecules
Concentration (nM)
0 Milieu
1 Ue 0 0.000
2 Ve 1 0.003
Zoom