# 1 Supplemental Information

## 1.1 Installation and requirements:

The code is entirely programmed in C++, and can be downloaded from gitlab.com/Moonfit/MoonLight.

Running Moonfit requires a C++ compiler and the Qt library to be installed. No external library is required (some are included as C++ files directly inside the folder) [1]. It is possible to use boost solvers, in which case the boost library should be installed.

- *In Linux:* g++ is recommanded (sudo apt-get install g++). The QT framework can either be installed as complete version (sudo apt-get install qt-sdk and sudo apt-get install qtcreator), or as core libraries one by one (qtbase5-dev, libqt5svg5, libqt5printsupport5, qtcreator). If boost is wanted, (sudo apt-get install libboost-dev).
- *In Windows:* A full version of Qt can be downloaded from https://www.qt.io/download, by choosing the Open Source version. More specifically, some offline installers available in https://www.qt.io/offline-installers/ include C++ compilers like MinGW or VisualC++. We recommend the most recent 'Qt for windows 32 (MinGW)', that works on both 32 and 64 bits systems. For boost, download the latest version (http://www.boost.org/users/download/) and unzip it anywhere. See below how to link it.
- *In MAC:* The clang C++ compiler can be installed with brew by running (brew install –with-clang llvm) on command line, and the QT platform can be downloaded from https://www.qt.io/download, by choosing the Open Source version.

Several files have been developed together with the software 'organism' (Developed at CBBP Lund / SLCU Cambridge, see gitlab.com/slcu/teamHJ/organism), which is able to define and simulate and optimize ODEs in a multicellular environment from ODEs defined in a text file, together with mechanical forces and cell population rules (e.g. [18]).

**Running examples:** The main folders of the code include: Docs for documentation, Extreminator for the built-in optimizers, Framework for the minimal classes to simulate a model, Interface for the graphical interface, NewProject as a guidelines to create new models, and Examples that include small models and respective datasets ready to fit. To run the examples, two ways are possible:

1. *with QtCreator:* open Examples.pro with QtCreator, click on 'configure project' if it is opened for the first time here, and then compile and run the code by CTRL+R or by clicking on the run green button. 'Release' mode is preferred. The interface will open directly.
2. *Through Manual compiling:* to this end, open a terminal in the folder of Examples.pro, and execute the commands: qmake Examples.pro (generates a makefile) and make. The executable file is created in an automatically generated folder ('build...') from the parent folder, and can be run from there. Note: on windows, the compiler might have a different command for make, such as mingw32-make, and it might be necessary to add the path of the compiler binaries into the system path (right-click on 'this computer', 'properties', 'advanced system properties', 'environment variables', and inside the 'path' field, add '; folderOfTheCompiler'.)

When launched, a list of different examples is proposed (see Examples/Examples.pdf), leading to the start of the graphical interface (Figure 2).

[1]The library libSRES [22] is included for SRES optimization, and CMA-ES optimization can be used by uncommenting the content of the CMAES.cc file and installing the C++ shark library. The library QCustomPlot is included in the graphical interface to display plots.

## 1.2 Customizing the ODE solver

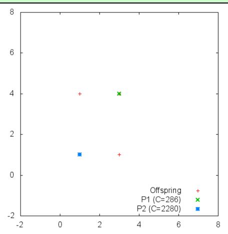
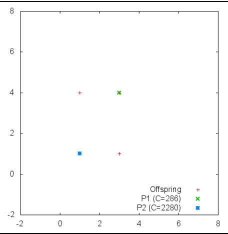
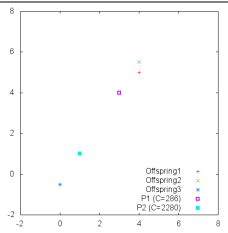
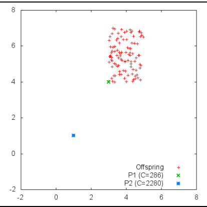
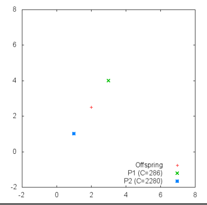
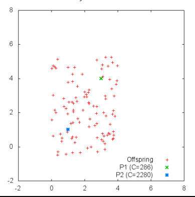
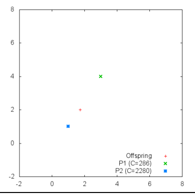
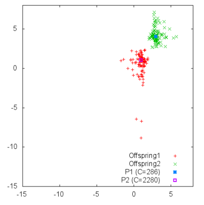
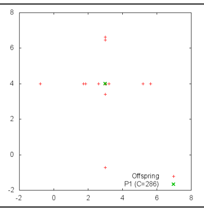
```
// Usual solvers like boost take a struct having an operator () performing the derivatives. So we have built here
// such a struct that just returns the derivatives from the model. Note the overriding/replace to zero of some variables
// to avoid that the solver interpretes a change in a variable as a computation error.
class mySim {
public:
    Modele* mm;
    mySim( Modele* _mm ) : mm(_mm) {}
    void operator()(const vector<double> &_x, vector<double> &_dxdt, const double t){
        mm->applyOverride((vector<double>&) _x, (double) t);  // replaces only variables to be overriden by data
        mm->derivatives(_x,_dxdt,t);
        mm->clearOverride((vector<double>&) _x, (vector<double>&) _dxdt); // put zero at all these data to avoid solver cost
    }
};

// The simulate function is calling the solver at that line (see inside):
// steps += myintegrate( BS , val , (double) t , (double) nxt , (double) dt );
// this is the function to be reimplemented if you want to change the solver. Below is the example to use Boost solver
void Modele::simulate(double sec_max, Evaluator* E){
    mySim BS = mySim(this);
    size_t steps = 0;
    double  nxt = t + sec_max;    // finds the next wanted time point
    clearOverride(val, val);      // cheating so the solver doesn't see the overrided variables (they put to 0)

    #ifdef USE_BOOST_SOLVER
    steps += RK4integrate( BS , val , (double) t , (double) nxt , (double) dt );
    #else
    steps += myintegrate( BS , val , (double) t , (double) nxt , (double) dt );     // built-in solver
    #endif

    t = nxt;
    applyOverride(val, t);        // now that the solver did the job, the variables can be overrided with the real values again
    if(checkDivergence()) return; // can be customized
}
```

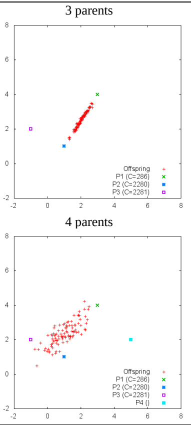
Figure 1: **Changing solver - how the simulate function calls the ODE solver.** The main structure of the simulate function, with the steps to replace certain variables by predefined dynamics (overriders). The syntax to use another solver is shown for the example of boost solvers.

## 1.3 Choice of optimizers / mutations / cross-overs

```
   Part 1 : type of optimizer and its options.
SRES  1     50000
   or
GeneticGeneral  14 # number of arguments
Type of algorithm
      0 -> CEP  (Classical Evolutionary Programming)
      1 -> SSGA (Steady State Genetic Algorithm)
      2 -> GGA  (Generational Gap Genetic Algorithm)

Type of selection for parents
      1 -> Rank-based selection
      2 -> Random selection
      3-6 -> Tournament selection
                  3 : subset size = 5% of population size
                  4 : subset size = 10% of population size
                  5 : subset size = 25% of population size
                  6 : subset size = 50% of population size
      7-12 -> Proportional selection
                  7-8 : Correction from worst, Basic Sampling / Roulette Wheel
                  9-10 : Correction from best, Basic Sampling / Roulette Wheel
                  11-12 : Correction inverse, Basic Sampling / Roulette Wheel

Type of cross-over    Parameter for cross-over
      0 -> One-Point
      1 -> Two-Points
      2 -> Wright nr 1
      4 -> Wright nr 2
      5 -> Arithmetic
      6 -> BLX-Alpha
      7 -> Geometric
      8 -> SBX
      9 -> Randomise One Point
      10 -> UNDX 3 parents
      11 -> UNDX 4 parents

Replacement policy (for GGA) -> 0 = replace worst parent (only implemented)

Type of population selection (to keep constant population size)
      0 -> Select best individuals
      1-12 : same as selection for parents

Type of Strategy      Parameter for strategy
      0 -> Constant
      1 -> Exponential Down
      2 -> Distance Best
      3 -> Distance Best Separated
      4 -> Proportional Normalised
      5 -> LogNormal
      6 -> Mutative
      7 -> Mutative_separated

Number of replicates of the algorithm

Number of total cost calls authorised
      (the number of generations will be computed according to that number)

Proportion of cross-over
      (in [0:1], for the creation of individuals, the remaining
      will be mutations)

Fork Coefficient
      Number of new individuals created each time (in terms of coefficient
      of the population size.)
 Part 2 : Parameter / boundaries / scaling options
Number of parameters to optimise

Indice_parameter_1   lower_init_boundary_param_1     upper_init_boundary_1
Indice_parameter_2   lower_init_boundary_param_2     upper_init_boundary_2
...

Number of initial parameter set

Parameter set 1
Parameter set 2
...

Scaling function      Number of parameters
      "ExponentialBounded   5
      Xmin Xmax (parameter scale), ymin, ymax (forced), coefficient"
or    "NoScaling            0"
or    "InverseAndLinear     0"
or    "Exponential          0"
```

Figure 2: **Choice of optimizers and syntax of the generated optimizer file**. The first part is the optimizer options, that has to be decided by the user (header), and the second part is automatically generated according to the choice of parameters to optimize. The next page is showing the details of the cross-over and mutations [6].

| Cross-Over | N. of Parents | N. of offspring (differents) | Parameter | Rule | Example in 2D (cloud of possible offspring) |
|---|---|---|---|---|---|
| One-Point | 2 | 1 (d-1) | | $\begin{aligned} pos &= U(1..lg-2) \\ offspring &= p_A[0..pos-1] :: p_B[pos..lg-1] \end{aligned}$ |  |
| Two-Points | 2 | 1 (d*(d-1)/2) | | $\begin{aligned} pos1, pos2 &= U(0..lg-1), until pos1 < pos2 \\ offspring &= p_A[0..pos1-1] :: p_B[pos1..pos2-1] :: p_A[pos2..lg-1] \end{aligned}$ |  |
| Wright nr 1 | 2 | 3 (3) | | $\begin{aligned} offspring_A &= p_A + p_B \\ offspring_B &= \frac{3}{2}.p_A - \frac{1}{2}.p_B \\ offspring_C &= -\frac{1}{2}.p_A + \frac{3}{2}.p_B \end{aligned}$ |  |
| Wright nr 2 | 2 | 1 (inf) | | $\begin{aligned} (p_{best}, p_{worst}) &= sort(p_A, p_B) \\ offspring &= U(0..1)[p_{best} - p_{worst}] + p_{worst} \end{aligned}$ |  |
| Arithmetic | 2 | 1 (1) | $\gamma \in [0;1]$ | $\forall i, offspring_i = (1-\gamma).p_{A,i} + \gamma.p_{B,i}$ | Here with $\gamma$ = 0.5  |
| BLX-$\alpha$ | 2 | 1 (inf) | $\alpha \geq 0$ | $\begin{aligned} \forall i, \gamma_i &= (1+2\alpha)U(0..1) - \alpha \\ offspring_i &= (1-\gamma_i).p_{A,i} + \gamma_i.p_{B,i} \end{aligned}$ | Here, $\alpha$ = 0.5  |
| Geometric | 2 | 1 (1) | $\gamma \in [0;1]$ | $\forall i, offspring_i = p_{A,i}^{(1-\gamma)} + p_{B,i}^{\gamma}$ | Here with $\gamma$ = 0.5  |
| SBX-$\eta$ | 2 | 2 (inf) | $\eta$ | $\begin{aligned} &\forall j, \\ rj &= U(0..1) \\ \gamma_j &= if r_j \leq \frac{1}{2}, (2r_j)^{\frac{1}{\eta+1}} \\ &\quad else, \left(\frac{1}{2(1-r_j)}\right)^{\frac{1}{\eta+1}} \\ offspring_{A,j} &= \frac{1}{2}((1+\gamma_j).p_{A,j} + (1-\gamma_j).p_{B,j}) \\ offspring_{A,j} &= \frac{1}{2}((1-\gamma_j).p_{A,j} + (1+\gamma_j).p_{B,j}) \end{aligned}$ | Here, $\eta$=1 (advised value)  |
| Randomise | 1 | 1 (inf) | | $\begin{aligned} pos &= U(0..lg-1) \\ \forall i \neq pos, offspring_i &= p_i \\ offspring_{pos} &= U(MinBoundary_{pos}, MaxBoundary_{pos}) \end{aligned}$ |  |

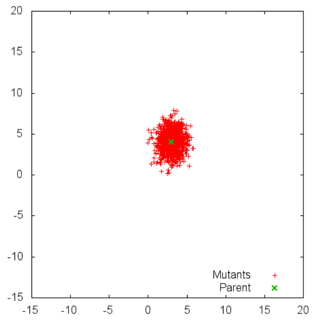
| | | | | | |
|---|---|---|---|---|---|
| UNDX – K parents, K > 2. | K | 1 (inf) | | $\sigma_1 = \frac{1}{\sqrt{dim-2}}$ $\sigma_2 = \frac{0.35}{\sqrt{dim-K-2}}$ $c =$ barycenter of parents except the last one $d_{I,(I=1..K-2)} =$ direction of parent I $= \frac{P_I - c}{\|\|P_I - c\|\|}$ $e_{I,(I=K-1..dimension)} =$ orthogonal supplementary basis of $Vect(d_1...d_{K-2})$ $\delta = dist(P_K, Vect(P_1...P_{K-1})$ : distance to the $K^th$ parent to the subs $offspring = c + \sum_{I=1}^{K-2} N(0,\sigma_1^2).\|\|P_I - c\|\|.d_I + \sum_{I=K-1}^{dim} N(0,\sigma_2^2).\delta.e_I$ |  |

## Mutations (1 parent, 1 offspring)

- The parameter **i** of an infdividual, is mutated by multiplication by **c**, according the following rules, with a given stepsize **$\sigma_i$**.
- Numeric values are and-made for keeping square deviation of $\sigma_i$.
- Examples show in 2D if we mutate all (2) parameters. You can notice that some mutations are axis dependant (ie not rotation-invariant)

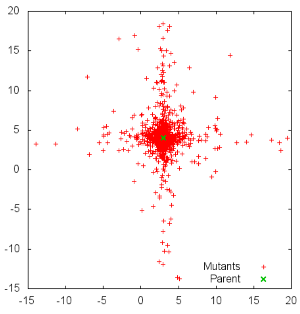
**Normal** distribution :

$$c = \sigma_i.N(0,1)$$

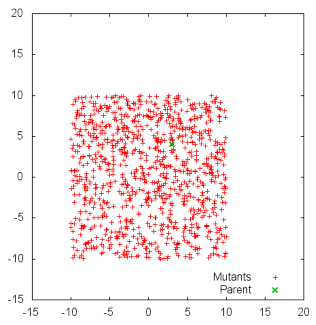


**Cauchy** distribution :

$$c = \sigma_i.C(0,0.0155)$$

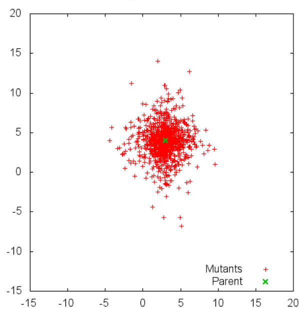


**Randomize** between boundaries



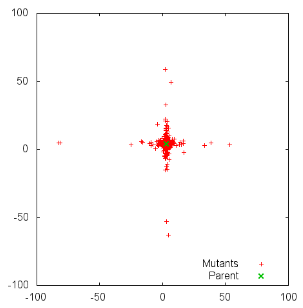
**Exponential** (double)

$$c = \sigma_i.\frac{\sqrt{2}}{2}.(\pm 1)E(0,1)$$



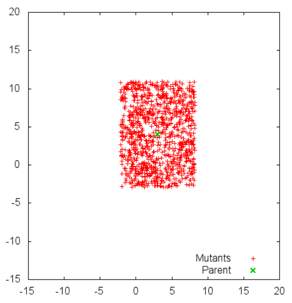
**Combined** Normal and Cauchy

$$c = \sigma_i.0.627(N(0,1) + C(0,0.0155))$$



Randomise in **Box** (on the scale of the parent)

$$c = \sigma_i.\sqrt{12}U\left(\frac{1}{2},\frac{1}{2}\right)$$


Mutants
Parent