Optim – An optimiser for deterministic StochSim models

Leif Gustafsson & Erik Gustafsson.

1. Background and Purpose

Optimisation is finding the *minimal* or *maximal* value of a function or model output with respect to some free parameters. The function or output you want to optimise is called the *objective function*. When you optimise the objective function *V* of a simulation model, it is about finding the values of a set of parameters: *P1*, *P2*, .., *Pn* that will produce the best value of the chosen objective function (*V*). For example, you want to maximise the profit of a crop on a farm by using a certain amount (p_1) of an expensive pesticide to be used at a certain point in time (p_2). Which values of *P1* and *P2* will produce the maximal profit ($V(p_1', p_2')$)?

Optim is a simplex optimiser of Nelder and Mead type for optimisation and parameter estimation of StochSim models.

An optimiser serves two different tasks in modelling and simulation. First, it can be used to tune the parameters of your model so that the model behaves as similarly as possible two the system under study. This is called *parameter estimation*.

Second, it cam be used to find the set of parameter values that optimises (minimises or maximises) an outcome (objective function) of your model.

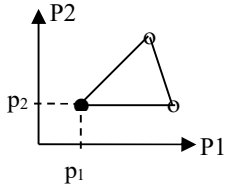
The purpose of Optim is to be an all-purpose, robust, efficient and easy-to-use optimiser for parameter estimation as well as optimisation of StochSim models.

2. The simplex optimiser

2.1 What is a simplex optimiser?

A simplex in *n*-dimensional space is a geometrical figure constructed by *n+1* linearly independent points (vertices); all points connected by straight lines. In two dimensions you get a triangle, in three a tetrahedron etc. - not necessarily with equally long edges. This triangle, tetrahedron, etc. will systematically be 'rolled over' towards the searched optimum.

Each of the *n* parameters in the optimisation problem (your model) is represented by one dimension in parameter space. Thus, let a vertex (corner) of the simplex has *n* coordinates (p_1, p_2, .., p_n) representing the values of the *n* parameters *P1*, *P2*, .., *Pn*. The value of an objective function $V(p_1, p_2, .., p_n)$ is calculated by a simulation run.



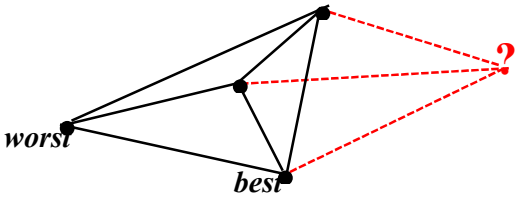
For every iteration the vertex with the worst objective function value is replaced by a better vertex until the process is ended because of localisation if the optimum. In the Nelder and Mead version there are four mechanisms: *Reflection*, *Expansion*, *Contraction* and *Shrinkage* of the simplex, which successively replaces the worst point(s) with better one(s). Each iteration requires one or several simulation runs to create a new simplex. For further details, see the literature.

2.2 How the simplex method works

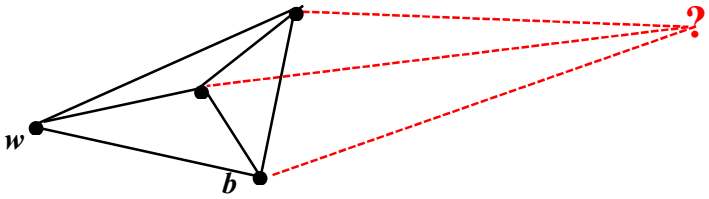
To optimise *V*(*P1, P2,…,Pn*), place *n+1* starting points (●) in the *n*-dimensional parameter space. In two dimensions (3 parameters) the simples is a triangle, in three (4 parameters) it is a tetrahedron, etc. This is called a *simplex*. Make a simulation for each vertex to calculate the objective function, V, there.

The new search for the optimum is obtained by replacing the worst (*w*) point by a new, using the mechanisms shown in Figure 1:

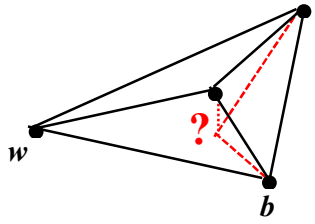
a) Reflection away from the worst point.



b) Expansion after Reflection (if still better).



c) Contraction away from worst point.



d) Shrinkage towards best point ***b***.

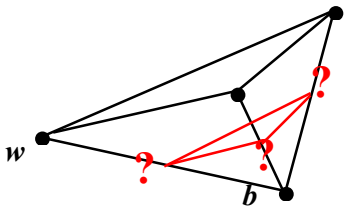


Figure 1. How the simplex method works.

Based on the mechanisms shown in Figure 2, the simplex is first moving towards the optimum and then starts to shrink. This process can be terminated when 1) the

simplex is sufficiently small, or 2) when the difference between the best and worst value of the objective function is sufficiently small. In Optim the latter condition is used.

2.3 Why a simplex optimiser?

All efficient optimisers are based on search methods. Such methods are usually classified as methods using only function values (e.g. obtained from simulations) and those that in addition use first or higher order derivatives. For smooth functions a method using first and second order derivatives converges very efficient towards the 'exact' value. Such an optimiser is an option when you need very high precision and a number of nice conditions are fulfilled. However, there are several reasons why such a method has limited value in the context of simulation.

First, it is only close to the theoretical value a method based on higher order derivatives becomes favourable. Usually a simulation model is directed to problems in disciplines like biology, medicine, economics etc. where the knowledge about reality has limited accuracy. In these cases there is no gain to use methods based on higher order derivatives.

Second, optimisation of a simulation model (compared to a mathematical function) is often characterised by complexity including discontinuities from e.g. if-statements of some kind, making the objective function non-smooth, which will fool a method based on derivatives.

Thirdly, the parameter space is often restricted. Parameters like investment, labour supply, accepted amount of a pesticide, etc. may be additional constraints. A simplex optimiser easily handles such a restriction. The trick is to give the objective function a very bad value (high when minimising and low when maximising) in the forbidden part of the parameter space, which will be respected by the simplex optimiser. (When using methods based on higher derivatives, a complicated, so-called boundary or punishment function has to be involved and modified during the search.) Limiting the ‘volume’ of parameter space is also a practical method when preventing computational overflow in a simulation, which easily can be used together with a simplex optimiser.

The simplex algorithm seems the best choice for a robust, reasonably fast and easy to handle, all purpose optimiser for simulation models.

3. Some comments on the model

3.1 Recommendations

Successful optimisation at a certain specified accuracy (Required Accuracy) also assumes that the model itself is sufficiently accurate! This requires a step size that is small enough and/or use of a efficient integration routine.

To find a proper step size, do a simulation. Then halve or double that step size to judge the accuracy because of the selected step size.

3.2 Different kinds of parameters

(We here only discuss continuous parameters in deterministic models)

Optimisation or parameter estimation may use different kinds of parameters like Start values of stocks, Proportional constants, Time constants for rates, and Event times for an action. The first two are not problematic.

When an Event time used to trigger an action is used as a parameter, you must be aware that the accuracy of that point in time is limited by the size of the time-step used. In this case you may have to reduce the step size more than otherwise is motivated.

3.3 Optimisation

To optimise a model, you just pick or create a proper objective function, say *V*, and let the optimiser do the job to find the best set of values for your selected set of parameters *P1, P2,..., Pn*. You can also choose to minimise or maximise *V* as function of the parameters.

3.4 Parameter estimation

When fitting a model to real data from the system, you need both a description of the system's behaviour an of the corresponding model's behaviour.

The system's behaviour can e.g. be described by real data tabulated in a *Converter* (here called SYSTEM) in StochSim.

The corresponding model quantity (called MODEL) is compared to that of the system, and the difference between them is squared. This is sketched in Figure 2.

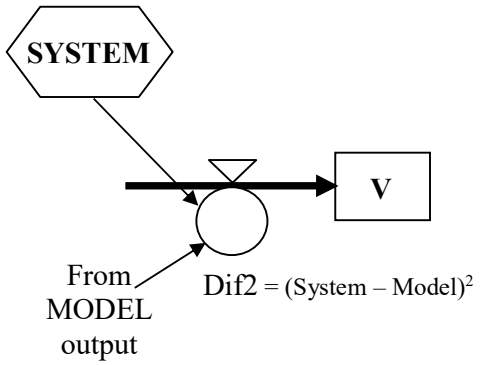


Figure 2. Model structure for parameter estimation.

During the simulation, the difference between the system and model quantity values is calculated for each time step. In least-square-estimation the square of this difference is used as the inflow to a stock (which is initialised to zero) that cumulates the squares of difference over the simulation time. This stock (V) is the objective function to be minimised with respect to the chosen parameters.

After the simulation the squared differences between the System and Model quantities ($DIF2=(SYSTEM-MODEL)^2$) are cumulated in objective function V. By repeated simulations we find the set of parameter values (p_1', p_2',..., p_n') that minimises the value of V.

4. Instructions for Optim

Before running Optim, finish and validate your StochSim model and be sure that the parameters and objective function exist in your model, as described above.

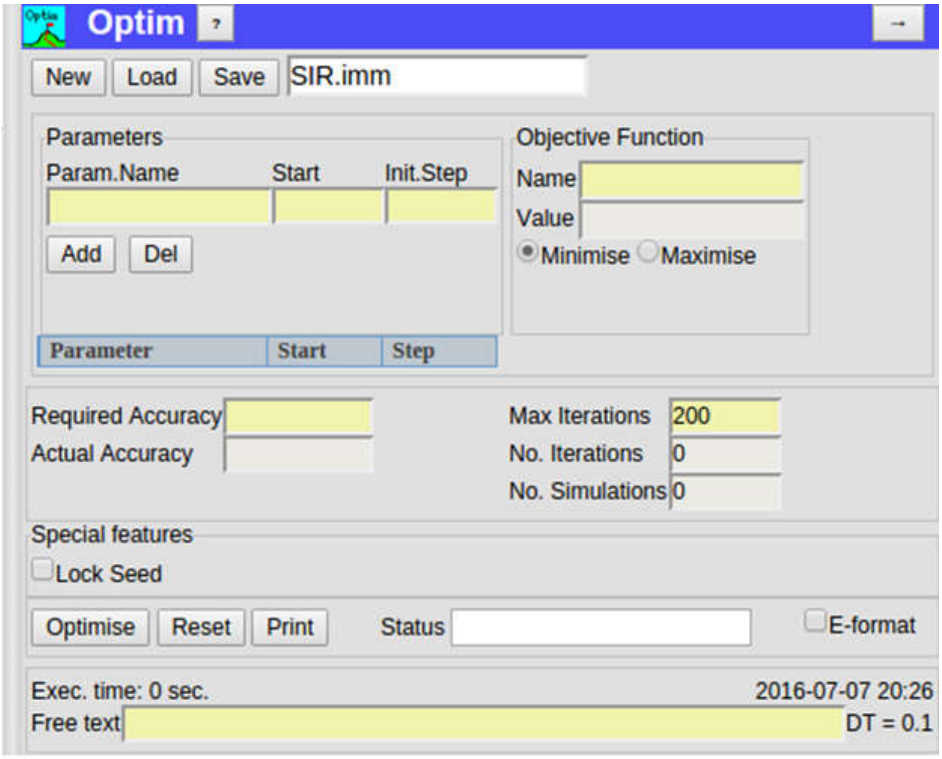


Figure 3. The Optim form.

When you have a deterministic StochSim model with an objective function and have specified the time-step and breaking criterion, you can use Optim:

1. Build or open a StochSim model.
2. Specify and **Add** the Parameters:
   a. Name the Parameter.
   b. Specify starting values of the parameter.
   c. Specify size of the initial simplex in the parameter's direction.

d. Press **Add**.

3. Enter the Name of the Objective Function.
4. Select **Minimise** or **Maximise**. (Default=Minimise)
5. Enter Required Accuracy.
   (When *Actual Accuracy* calculated as: $|V_{best} - V_{worst}|$ becomes smaller than *Required Accuracy* Optim will terminate. Here V symbolises the Objective Function.)

[6. Enter Maximum Number of Iterations (Default=200).]
[7. **Lock Seed** is only required for a stochastic model (i.e. a model using]
[random numbers). The Seed can then be chosen and locked from Optim.]

8. Press the **Optimise** button.

[9. When ended, the optimisation may be continued after change of]
[Required Accuracy or Max Iterations.]
[You may also **Halt** and/or **Reset** the optimisation.]

10. The **Print** button prints the Optim form as shown.

Notes: You can only write in the light yellow fields.
The [ ] around the primitives in StochSim should here be excluded.
The variable names in StochSim are not case sensitive.
Use decimal point, *not* comma, when specifying the values.

Additional features and comments

- **Help button [?]** at the blue title bar gives you information of how to make the specifications of Optim.

- The **arrow** button [→] hides Optim to give more space for the Model Window.

- **Parameters** are specified by Name, Starting value and Size of the initial simplex in the parameter's direction of the parameter space.

  A parameter may be **Added** or **Deleted**. Be careful to spell the names correctly.

  Note: A tested parameter value may become zero. If that parameter appears in the denominator in the StochSim model you get: "Division by zero". For example, if a flow is defined by *Stock / T*, where *T* is the time constant, this might happen. (You can avoid this by writing *Flow = Stock*c* where *c* is the inverted value of *T*.)

  Be careful to select the correct button to **Minimise** or **Maximise**. Note also that a model may have several minima or maxima. The optimum you get may be a local (as distinct from a global) optimum. If unsure, start the optimisation from different points in parameter space.

- The **Actual Accuracy** is calculated as the difference between highest and lowest function value of the simplex vertices. For each iteration the Actual Accuracy is compared to the Required Accuracy. When the calculated Accuracy becomes smaller than the Required Accuracy the optimisation is done.

Note 1: The accuracy of your calculations is only related to your model and the accuracy in your calculations. If the model is only a coarse description of the real system, the high accuracy is only fictional. Even if the model is a 'mathematically perfect' description of the real system, the selected step size restricts the accuracy. Always test the accuracy of your obtained results by comparing to a simulation with halved step size. Then you find out whether your results depend on the selected step size or not. Higher accuracy may require a shorter step size and/or a more efficient integration method.

Note 2: If you are more interested in estimating the parameter values than the value of the objective function, you may have to set a sufficiently small value for the Required Accuracy.

- The **Max Iterations** has the default value 200. If the process has not converged, you may increasing 'Max Iterations' and pressing 'Continue'.
- The **Optimise/Halt** button starts the optimisation. This button also is used to Halt the process, and to Continue it.
- **Print** button prints the Optim form as shown.
- **Status** shows the status of the optimisation process.
- **E-format** can be used to get the results in the form ±X.XXE±X, which means $\pm X.XX \cdot 10^{\pm X}$.
- **Exec. time** shows the time since you pressed optimise until the optimisation is done.
- A **Free text** field is also provided for commenting the model or optimisation.
- **DT =** documents the step size used in the StochSim model.

5. References

- Nelder J.A. and Mead R. (1965) A Simplex Method for Function Minimisation. Computer Journal, 7, 308.
- Gill P.E., Murray W. and Wright M.H. (1981) Practical Optimisation, Academic Press.
- Press W.H., Flannery B.P., Teukolsky S.A. and Vetterling W.T. (1989) Numerical Recipes (in FORTRAN, Pascal or C), Cambridge University Press.

6. Responsibility

The user is fully responsible for the use of this product. The producer and the supplier of this code take no responsibility for the use or functioning of Optim.