THIRD PARTY OPTIMIZATION TOOLS - A SURVEY

Florian Judex[1]

[1]Austrian Institute of Technology
Energy Department Sustainable Building Technology
Giefinggasse 2, 1210 Vienna, Austria

florian.judex@ait.ac.at(Florian Judex)

Abstract

Most specialized software packages for simulation lack the capability of optimization (and therefore also identification). Third Party software fills this gap by providing optimization tools that treat the simulation as a black box function that has to be minimized. The range of this tools spans from elaborate commercial products with comfortable GUIs to open source products which are delivered as packages for programming languages. The standards in scientific documentation of the employed algorithms and their exact implementation vary between undocumented propitiatory code with unknown functionally to scientifically disseminated algorithms shipped together with well defined interfaces. In this paper, eight software tools or libraries are described and tested on two problems, a classical optimization problem - the Rosenbrock function - and a benchmark problem from the SNE magazine, which resembles a real life model calibration task. It is shown that commercial products do not outperform open source Software. Therefore open source tools or libraries unsurprisingly excel in cases where transparency is important, while the commercial tools are better suited for users who only care for results.

Keywords: Optimization, identification, ARGESIM Comparison

Presenting Author's Biography

Florian Judex specialised in modelling and simulation during his master and PhD studies in applied mathematics at the Vienna University of Technology. Currently he is a researcher in the field of sustainable building technology in the Energy Department of the Austrian institute of technology, dealing with the modelling and optimization of renewable energy systems.

1 Introduction

Although optimization of systems - including the identification of unknown system parameters to gathered data - is one of the most, if not the premier field where modelling and simulation is used as a tool, most simulation software does not include optimization routines. Third party software fills this gap, treating the simulation software as a black box function which is to be optimized. These software tools are available in a wide range of prices from free Open-Source tools to expensive commercial products. Furthermore the amount of information about the tool itself varies greatly. In the worst case, the optimization software itself is a black box systems where the manufacturer just gives hints at the techniques used, while in some cases not only the code is open to everybody, but is annotated with full references to the appropriate scientific literature.

This paper reviews a number of available tools from both categories with feature lists, numerical testing as well as user experiences with the software.

2 Numerical tests

Two problems were selected to test the software.

2.1 Rosenbrock function

The Rosenbrock function

$$f(x,y) = (1-x)^2 + 100 \cdot (y - x^2)^2$$

is one of the most common test cases for optimization software. It can be seen easily that it has single minimum at $(x,y) = (1,1)$ with $f(x,y) = 0$. This minimum is positioned on the $y = x^2$ parabola, with the gradient in direction of this valley being extremely low in comparison to the gradients orthogonal on the parabola. Therefore simpler optimization algorithms will stop once the reached the valley.

The function was implemented as an executable, reading the two parameters from the first two lines of the text file `in.txt`, e.g.

```
3.4
4.2
```

and returns the value of $f(x)$ to a `out.txt` file as e.g.

```
234.56
```

Both file are not the standard I/O files for any software tested in this paper. Also no optimization tools was able to use the simple I/O files provided directly, leading to a different amount of pre and post processing necessary.

2.2 Identification

As identification problem (In many fields of science and technology, this process is called calibration. In this paper, the term identification is used to be consistent with [1]) the ARGESIM Comparison 15 from the SNE magazine [1] was chosen (the detailed definition is also available on `http://www.argesim.org/`). The system is a model consisting of two compartments, representing the extracellular space of a kidney. The state variables x_1 and x_2 represent the concentration of a marker in the volumes, with unknown transitions k_{12} and k_{21} between the compartments, an unknown clearance rate (outflow) k_{01} from the first compartment and an unknown volume for the first compartment. With a given input τ in the first compartment and 19 pairs of (x_{1i}, t_i) data, the for unknown variables $(k_{12}, k_{21}, k_{01}, V_1)$ should be should be identified with the standard target function

$$\sum_{i=1}^{19} (x_{1i} - x_1(t_i))^2 \to \min$$

where $x_1(t)$ represents the solution of the differential equations derived from the compartment model. As the differential equation are simple enough to have an analytic solution, a executable was created using the same input and output files as for the Rosenbrock function. This time `in.txt` file contains the four parameters, while `out.txt` returns the 19 values at the times where data is available.

2.3 Testing Procedure

For all test performed, relative and/or absolute accuracy was set to 10^{-4}, depending on the choices available. All the other parameters where left a the defaults provided at installation time.

For the Rosenbrock function, all software that allowed the automation of configuration with respect to the starting values for parameters was tested by running each available algorithm from 25 different starting points of the set $\{(x,y) | x, y \in \{\pm 2, \pm 1, 0\}\}$. This set contains the actual optimum on purpose.

As all programs behave different when giving the number of function calls, the executable file used for the testing writes a empty file with a 16 character file name in a predefined directory, so the number of calls can be determined by just counting those files. The file name is composed randomly of the characters '0'-'9' with a random seed calculated from the current system time, so the probability of missing an iteration because of two identical random number chosen is 10^{-16}. As none of the algorithms needed more then 10^4 iterations, the cumulative probability for this kind of is error is 10^{-12} and therefore negligible.

Due the huge amount of algorithms available, and the large numbers of parameters these algorithms depend on that can be varied, results are only provided for the best algorithm for each task of each of the three types defined in table 1. For the Rosenbrock function, the average function value $\overline{f(x)}$ for all 25 runs is stated, along with the average number of runs $\overline{n}$. The algorithms chosen in each case are stated in the feature list for each software. For the identification, the procedure was directly take from 'Task C' of [1] which calls for a Monte Carlo method using 1000 iterations of the identification process with artificially generated data.

Table 2 gives the algorithm which got results reasonably close to those of the sample solution of the task [2] with the least number of iterations - again using only the default parameters provided - and the respective numerical results for the mean μ and the standard deviation σ.

One software tested (VisualDoc) had no batch-mode capability and was therefore excluded from these tests.

3 Numerical results and features

The features of each piece of software are listed. A table each contains the information about the algorithms implemented in the software, divided into three main categories: gradient based methods (e.g. Newton's method), deterministic heuristic methods (pattern e.g. search) and heuristic methods with stochastic factors (e.g. simulated annealing) with the exact algorithm chosen for table 1.

Furthermore the input/output possibilities, capability of utilising parallel processing techniques and availability for different operating systems are stated. as five of the eight tools tested are not individual tools but rather libraries for other software, which in each case is available for LINUX / UNIX as well as the Windows NT family of operating system, these were classified as independent of the type of operation system.

3.1 DAKOTA[3]

The Design Analysis Kit for optimization and Terascale Applications (DAKOTA) is published under the GNU LPGL by the Sandia National Laboratories. It is mostly command line based, although a rough JAVA based GUI exists, as well as independent output graphics for the Linux version and its port to the Windows NT family (which requires Cygwin to be installed).

It serves as number of different optimization libraries which are also published under the LGPL, and is also prepared to work with several commercial libraries. It has very strict formats for its I/O, but as the C++ libraries are available, it can be directly incorporated into software projects.

3.2 Dymola Design and Optimization library[4]

This commercial Dymola library (in the tables abbreviated as DYDO) feature gradient based algorithms as well as genetic algorithms and direct search. Via a `Modelica.Utilities.Sytem` call, also the external software written for this test can be used. While some of the features do not work this way - especially the calibration procedures - as the library is intended to

Tab. 3 Features of DAKOTA

feature	yes	no
GUI (input)	(✓)	
GUI (output)	(✓)	
batch mode	✓	
parallelisation	✓	
gradient based SQP	✓	
deterministic heuristic pattern search	✓	
stochastic heuristic GA	✓	
OS dependent Windows NT family Linux / Unix	✓ ✓ ✓	
licence	GNU lesser GPL	

Tab. 4 Features of the Dymola Design and optimization library

feature	yes	no
GUI (input)	✓	
GUI (output)		✓
batch mode	✓	
parallelisation		✓
gradient based SQP	✓	
deterministic heuristic pattern search	✓	
stochastic heuristic GA	✓	
OS dependent Windows NT family Linux / Unix		✓
license	commercial	

be used with Dymola models only, users who already own this library can use it for other optimization problems as well, including use of the scripting language. A major drawback is the lack of parallel computing capabilities.

3.3 Generic Optimization Program[5]

The Generic Optimization Program (GenOpt) is developed by the Lawrence Berkley National Laboratory. Released under a slightly modified version of the BSD license, it offers a selection of classical algorithms. Implemented are generalised pattern search with a particle swarm extension, the simplex algorithm and a discrete gradient algorithm. All of those are documented very well including pseudo-code and the most important references.

Instructions for Genopt consist four files - a configurations file for the black-box software, a command file for the algorithm used, an initialisation file for the optimization and a template file linking everything together. Although this sounds rather cumbersome,

Tab. 1 Evaluation of the Rosenbrock function

Software	Gradient		deterministic		probabilistic	
	$\overline{f(x)}$	$\overline{n}.$	$\overline{f(x)}$	$\overline{n}.$	$\overline{f(x)}$	$\overline{n}.$
DAKOTA	0.0046983	70	0.043273	119	6.5102e-005	470
DYDO	0.0042215	69	0.040207	142	5.184e-005	600
GenOpt	0.004711	73	0.05564	100		
MATLAB GOT			0.042236	133	4.6785e-005	635
MATLAB OT	0.005422	71				
MOPS	0.0050156	69	0.037843	127	4.868e-005	507
OpenOpt	0.0058733	63				

Tab. 2 ARGESIM C15 results

Software	algorithm	k_{01}		k_{12}		k_{21}		V_1	
		μ	σ	μ	σ	μ	σ	μ	σ
DAKOTA	NL2SOL	0.00395	0.00018	0.0514	0.00511	0.0495	0.00507	7.04	0.216
DYDO	SQP	0.00418	0.000121	0.0494	0.00497	0.0503	0.00506	7.11	0.405
GenOpt	discrete gradient	0.00408	0.000171	0.0489	0.00512	0.0498	0.00504	7.14	0.299
MATLAB GOT	Simulated Annealing	0.00406	0.00028	0.0492	0.00506	0.0509	0.00511	7.06	0.058
MATLAB OT	Levenberg-Marquardt	0.00398	0.00027	0.05	0.00507	0.0501	0.00502	7.29	0.368
MOPS	SQP	0.00397	0.000273	0.0501	0.00488	0.0494	0.00502	7.19	0.127
OpenOpt	BFGS	0.00408	0.000185	0.048	0.0048	0.048	0.00483	7.36	0.191

Tab. 5 Features of GenOpt

feature	yes	no
GUI (input)		✓
GUI (output)	✓	
batch mode	✓	
parallelisation	✓	
gradient based	✓	
deterministic heuristic discrete gradient	✓	
stochastic heuristic pattern search		✓
OS dependent Windows NT family Linux / Unix		✓
license	modified BSD	

Tab. 6 Features of the MATLAB Global Optimization Toolbox

feature	yes	no
GUI (input)	✓	
GUI (output)	✓	
batch mode	✓	
gradient based		
deterministic heuristic direct search	✓	
stochastic heuristic simulated annealing	✓	
OS dependent Windows NT family Linux / Unix		✓
license	commercial	

all of them can be reused to a great extend, and the software is quite easy to use. The advantage of this setup is that this mechanisms are used for dealing with multiple instances of the same program, if possible and / or necessary. The GUI displaying the choice of parameters and the value of the target function over the iteration was found to be very useful, as it allows for a quick visual check whether the optimization is actually progressing.

3.4 MATLAB Global Optimization Toolbox[6]

The Mathworks offers two optimization libraries for their flagship software MATLAB. This one - in the tables abbreviated as MATLAB GOT - contains the algorithms which do not use any mathematical information about the function, but heuristic strategies in the search for optima. As with all MATLAB toolboxes proviede by the Mathworks, the code is open to view, and scientifically disseminated. Apart form various pattern search algorithms, simulated annealing and a general genetic algorithm are included (this toolbox was formerly sold as MATLAB Direct Search and GA Toolbox). One drawback for the use of these algorithms is the fact that to exploit their potential, the MATLAB parallel processing toolbox has to be bought as well.

Tab. 7 Features of the MATLAB Optimization Toolbox

feature	yes	no
GUI (input)	✓	
GUI (output)	✓	
batch mode	✓	
gradient based BFGS	✓	
deterministic heuristic		
stochastic heuristic		
OS dependent Windows NT family Linux / Unix		✓
license	commercial	

Tab. 8 Features of MOPS

feature	yes	no
GUI (input)	✓	
GUI (output)	✓	
batch mode	✓	
gradient based SQP	✓	
deterministic heuristic Pattern Search	✓	
stochastic heuristic GA	✓	
OS dependent Windows NT family Linux / Unix		✓
license	commercial	

3.5 MATLAB Optimization Toolbox[6]

This MATLAB toolbox (abbreviation: MATLAB OT) contains all the algorithms which are usually summarized under the term mathematical optimization - all methods based on gradients or their approximation. A very broad spectrum of methods for different scales as well as constrained and unconstrained problems is available. Again transparency and documentation are very good. A basic GUI allows the user to specify the parameters for an optimization task slightly easier then in the command line. Again, parallelisation is only possible by buying another toolbox.

3.6 Multi-Objective Parameter Synthesis[7]

A MATLAB toolbox developed by the Institute of Robotics and Mechatronics, System Dynamics and Control of the German Aerospace Center, abbreviated to MOPS - the German term for pug. It is a combination of a GUI and number of 'set' and 'get' style commands that have to be used in the right place in a MATLAB function to communicate with the optimization software. For parallelisation, it relies on the free PM-toolbox. It is shipped as compiled code and a very good documentation, as well as with a small array of annoying dog noises in mp3 format to notify the user when his or her optimization has finished. Notable is the capability for Pareto optimization.

3.7 OpenOpt[8]

This is a free PYTHON packages, distributed under the BSD license. It is optimised to run within in the SciPy PYTHON distribution, which tries to emulated MATLAB by using several other relevant PYTHON packages. OpenOpt consists of a lot of solvers and some wrappers to C++ and FORTRAN solver, which in turn are distributed under various Open Source licenses.

With the integration in the SciPy environment the software has various graphical output capabilities, but is still command line driven. Linking to the black box software also has to be done by a wrapper function, which also has to handle the problems that can arise from OpenOpts multi-threading capability. While quite good at black box optimization, it is better suited for classical curve fitting with known analytic functions due to the automatic differentiation routines built into OpenOpt.

Tab. 9 Features of OpenOpt

feature	yes	no
GUI (input)		✓
GUI (output)		✓
batch mode	✓	
gradient based BFGS	✓	
deterministic heuristic		✓
stochastic heuristic		✓
OS dependent Windows NT family Linux / Unix		✓
license	BSD	

3.8 VisualDoc[9]

A commercial software sold by Vanderplaats Research & Development, Inc. Aimed at engineers without any experience in optimization, it offers a large number of algorithms including Pareto optimization and a extensive graphical user interface.

One of its most prominent features is the 'VisualScript' interfaces, which basically allows a user to do the scripting needed for more complex problems in a visual interface, and then converting it to PYTHON code. Within the interface, a whole flow chart can be modelled, including links to multiple black box programs, and links for MATLAB and Excel are provided. All optimization tasks and their results are saved to a internal database, and can be redone or evaluated afterwards. A huge range of algorithms is provided, but not the ability to define optimization tasks without the GUI.

A professional tool a high price for users who do not want to focus on the optimization, as the individual algorithms are neither discussed in depth nor are there references to the original publications.

Tab. 10 Features of VisualDoc

feature	yes	no
GUI (input)	✓	
GUI (output)	✓	
batch mode		✓
gradient based	✓	
deterministic heuristic	✓	
stochastic heuristic	✓	
OS dependent Windows NT family Linux / Unix	✓ ✓ ✓	
license	commercial	

4 Conclusion

As can be seen in the two tables regarding the numerical results, commercial software did not outperform the open source tools. Also, as most of the Open Source tools were developed within the scientific community, not only the source code but also the documentation live up to scientific standards by providing all the information needed by a prospective user to be sure that the results produced can be properly explained, and if necessary can be reproduced by others without prohibitive licensing costs.

Commercial software on the other hand is more convenient to use, and better suited for situations were results are needed, and there is no need for reproduction or proof.

The two MATLAB toolboxes sold by Mathworks are a compromise. In all three fields - user friendliness, documentation and open sources code - they are somewhere in between open source software and the single purpose tools tested.

5 References

[1] N.Popper W.Estelberger. C15 clearance identification - definition. *Simulation News Europe*, (35/36):65f, 12 2002.

[2] N.Popper. C15 clearance identification - matlab / simulink. *Simulation News Europe*, (35/36):68, 12 2002.

[3] http://www.cs.sandia.gov/dakota/.

[4] http://www.3ds.com/products/catia/portfolio/dymola.

[5] http://simulationresearch.lbl.gov/go/.

[6] http://www.mathworks.com.

[7] http://www.dlr.de/rm/en/desktopdefault.aspx/tabid-3842/6343_read-9099/.

[8] http://www.openopt.org/.

[9] http://www.vrand.com/.