GenOpt® – A Generic Optimization Program

Michael Wetter
Simulation Research Group
Building Technologies Department
Environmental Energy Technologies Division
Lawrence Berkeley National Laboratory
Berkeley, CA 94720 – USA

ABSTRACT

The potential offered by computer simulation is often not realized: Due to the interaction of system variables, simulation users rarely know how to choose input parameter settings that lead to optimal performance of a given system. Thus, a program called GenOpt® that automatically determines optimal parameter settings has been developed.

GenOpt is a generic optimization program. It minimizes an objective function with respect to multiple parameters. The objective function is evaluated by a simulation program that is iteratively called by GenOpt. In thermal building simulation – which is the main target of GenOpt – the simulation program usually has text-based I/O. The paper shows how GenOpt's simulation program interface allows the coupling of any simulation program with text based I/O by simply editing a configuration file, avoiding code modification of the simulation program. By using object-oriented programming, a high-level interface for adding minimization algorithms to GenOpt's library has been developed. We show how the algorithm interface separates the minimization algorithms and GenOpt's kernel, which allows implementing additional algorithms without being familiar with the kernel or having to recompile it. The algorithms can access utility classes that are commonly used for minimization, such as optimality check, line-search, etc.

GenOpt has successfully solved various optimization problems in thermal building simulation. We show an example of minimizing source energy consumption of an office building using EnergyPlus, and of minimizing auxiliary electric energy of a solar domestic hot water system using TRNSYS. For both examples, the time required to set up the optimization was less than one hour, and the energy savings are about 15%, together with better daylighting usage or lower investment costs, respectively.

INTRODUCTION

Why Do Optimization?

Usually, a lot of time is spent in creating the input for a simulation model, but once this is done, the user usually does not determine the parameter values that lead to optimal system performance. This can be because there is no time left to do the tedious process of changing input values, running the simulation, interpreting the new results and guessing how to change the input for the next trial, or because the system being analyzed is so complex that the user is just not capable of understanding the nonlinear interactions of the various parameters. However, using mathematical programming, it is possible to do automatic single- or multi-parameter optimization with search techniques that require only little effort.

What is GenOpt?

GenOpt is a generic optimization program being developed for such system optimizations. It is designed for finding the values of user-selected design parameters that minimize a so-called objective function, such as annual energy use, peak electrical demand, or predicted percentage of dissatisfied people (PPD value), leading to best operation of a given system. The objective function is calculated by an external simulation program, such as SPARK, EnergyPlus, DOE-2, TRNSYS, etc. GenOpt can also identify unknown parameters in a data-fitting process.

GenOpt allows coupling any simulation program with text-based I/O by simply modifying a configuration file, without requiring code modifications. Further, it has an open interface for easily adding custom minimization algorithms to its library. This allows using GenOpt as an environment for the development of optimization algorithms.

How GenOpt Works

To perform the optimization GenOpt automatically generates input files for the simulation program. These files are based on input templates for the particular simulation program. GenOpt then launches the simulation program, reads the function value being minimized from the simulation result file, checks possible simulation errors and then determines a new set of input parameters for the next run. The whole process is repeated iteratively until a minimum of the function is found. If the simulation problem has some underlying constraints, they can be taken into account either by a default implementation or by modifying the function that has to be minimized. GenOpt offers a default scheme for simple constraints on the independent variables (*box constraints*), as well as a formalism that allows adding constraints to the simulation problem by means of so-called *penalty* or *barrier functions*. For example, GenOpt could be used to find the window area on the different facades of a house that minimizes annual energy use subject to the constraint that each area must be within user-specified minimum and maximum values.

Simulation Program Interface

GenOpt has an open interface on both the simulation program side and the minimization algorithm side. It allows the easy coupling of any external program (like EnergyPlus, SPARK, DOE-2, TRNSYS, etc., or any user-written program) by simply modifying a configuration file.

The data exchange between GenOpt and the external program is done with text files only (see Fig. 1). For performing the optimization, GenOpt, based on input template files, automatically generates new input files for the simulation program. To generate such templates, the user accesses the already-defined simulation input files and replaces the numerical values of the parameters to be modified with keywords. GenOpt then replaces these keywords with the corresponding numerical values and writes the simulation input files. This approach makes GenOpt capable of writing text input files for any simulation program. In a configuration file, the user can specify how the simulation program is to be launched and where GenOpt can find the current value of the objective function to be minimized, as well as other values that may be processed by the optimization algorithm.

This makes it possible to couple any external program to GenOpt without modifying and recompiling either program. The only requirement of the external program is that it must read its input from text files and write the function value to be minimized (plus any possible error messages) to text files.

Optimization Algorithm Interface

Users can easily add their own minimization algorithms to GenOpt's library by extending the superclass `Optimizer` that offers access to the GenOpt kernel (see Fig. 2). This superclass is the interface between the GenOpt kernel and the minimization algorithm. It offers methods to retrieve the required settings for the specific algorithm, the initial values of the design parameters, and possible bounds of these values. It also offers several other methods like evaluating a simulation with the current set of design parameters or reporting the results of the optimization run. Utility classes (e.g., for linear algebra, for line-search or for checking the optimality conditions) can be shared by different algorithms. Thus, the user has only to deal with the actual mathematical formulation of the minimization algorithm and not with the data handling, output writing, syntax checking and other tedious work.

Status of GenOpt 1.1

GenOpt 1.1 can be downloaded free of charge from http://SimulationResearch.lbl.gov. It can be run either with a graphical user interface (GUI, see Fig. 3), or as a console application. The GUI features an online chart showing the optimization progress of the objective function and of the design parameters. All values can be added and removed from the chart during runtime. The console application allows using GenOpt from a shell script or a batch job for several sequential optimizations. GenOpt is written entirely in Java so that it is completely platform independent.

The following optimization algorithms are implemented in GenOpt 1.1:

- Pattern Search algorithm from Hooke and Jeeves (1961), with modifications of Smith (1969), Bell and Pike (1966), and De Vogelaere (1968). The Pattern Search algorithm is particularly useful for curve fitting, but is also efficient for other minimization problems. The number of function evaluations increases only linearly with the number of design parameters.

Input Files

initialization:	Specification of file location (input files, output files, log file, etc.)
command:	Specification of parameter names, initial values, bounds, optimization algorithm, etc.
configuration:	Configuration of simulation program (error indicators, start command, etc.)
simulation input template:	Templates of simulation input files

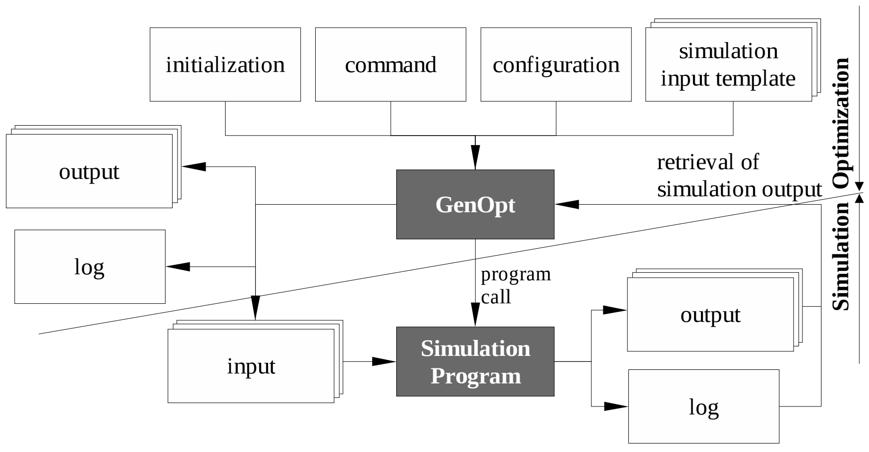
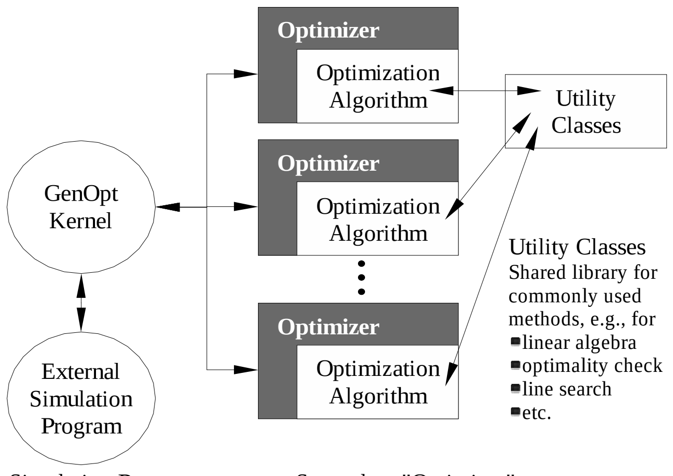


Figure 1: Interface between GenOpt and the simulation program

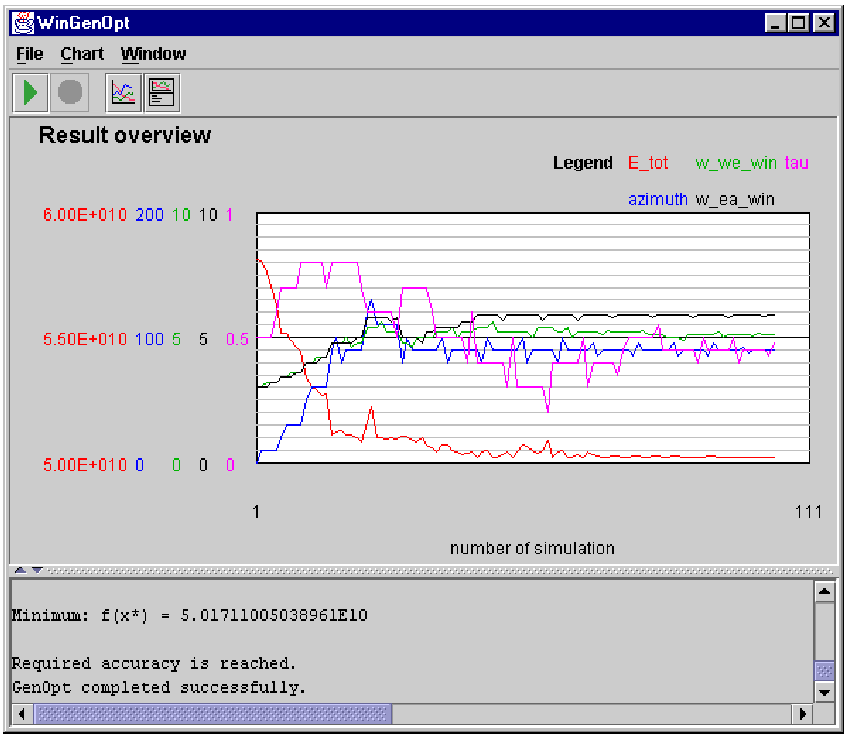


Utility Classes
Shared library for commonly used methods, e.g., for
- linear algebra
- optimality check
- line search
- etc.

Simulation Program
Any simulation program with text- based I/O, e.g.,
- EnergyPlus
- SPARK
- DOE- 2
- TRNSYS
- etc.

Superclass "Optimizer"
Offers methods to easily access GenOpt's kernel, e.g., for
- input retrieval
- objective function evaluation
- results reporting
- error reporting
- etc.

Figure 2: Implementation of minimization algorithms in GenOpt



```
Minimum: f(x*) = 5.01711005038961E10

Required accuracy is reached.
GenOpt completed successfully.
```

Figure 3: Example screen showing the optimization process

- Simplex method of Nelder and Mead (1965) with the extension of O'Neill (1971). The Simplex method is known to be efficient for, but not restricted to, most problems with up to about 10 design parameters.

- One-dimensional line search using Golden Section division.

- One-dimensional line search using Fibonacci division.

- Grid generator for parametric runs on an orthogonal, equidistant grid in a space of arbitrary finite dimension.

EXAMPLES

We will present the minimization of the source energy consumption of an office building, and the minimization of the electricity consumption for a solar domestic hot water system. Both minimization problems can be formally stated as

$$\min_{x \in \mathbf{X}} f(x) \tag{1a}$$

$$\mathbf{X} \triangleq \{x \in \mathbb{R}^n \mid l^i \leq x^i \leq u^i;\ i \in \{1, \ldots, n\}; \quad l^i, u^i \in \mathbb{R} \cup \{\pm\infty\}, \forall\, i\ \}, \tag{1b}$$

where $f\colon \mathbb{R}^n \rightarrow \mathbb{R}$ is the objective function, $x \in \mathbf{X} \subset \mathbb{R}^n$ is the set of design parameters, and $\mathbf{X}$ is the feasible set for x. $l \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$ denote lower and upper bounds, respectively, of the design parameter.

We note that even though both examples that are presented below have only 4 design parameters, GenOpt can be used to solve optimization problems with any number of design parameters.

Office Building

We optimize the yearly source energy consumption for heating, cooling, and lighting of an office building in Chicago, IL. The thermal zone depicted in Fig. 4 is representative of a typical zone in an office building. Floor, ceiling, and north and south walls of the zone are adiabatic. The exterior walls have a U-value of $0.25\,\mathrm{W}/(\mathrm{m}^2\,\mathrm{K})$ and consist of (listed from outside to inside) $1\,\mathrm{cm}$ wood siding, $10\,\mathrm{cm}$ insulation and $20\,\mathrm{cm}$ concrete. The ceiling and floor consist of carpet, $5\,\mathrm{cm}$ concrete, insulation and $18\,\mathrm{cm}$ concrete. Interior walls are $12\,\mathrm{cm}$ brick. Both windows are low emissivity double pane window with Krypton gas fill and exterior shading device. The shading device is actuated only

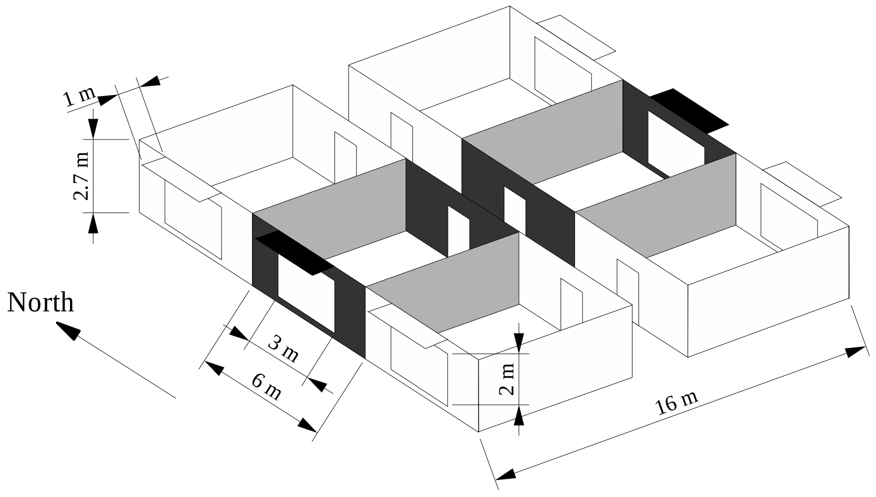


Figure 4: Office zone (shaded) in its initial configuration before the optimization

during summer when the solar irradiation on the window exceeds $200\,\mathrm{W/m^2}$. Both windows have a fixed overhang that projects out $1\,\mathrm{m}$. The zone has daylighting controls with an illuminance setpoint of $500\,\mathrm{lux}$ at a point $3\,\mathrm{m}$ from each window.

The annual source energy consumption is given by

$$\begin{aligned} f(x) &\triangleq E_{tot}(x) \\ &= \frac{Q_{heat}(x)}{\eta_{heat}} + \frac{Q_{cool}(x)}{\eta_{cool}} + 3\,E_{lights}(x), \end{aligned} \quad (2)$$

where $Q_{heat}(x)$ and $Q_{cool}(x)$ are the zone's yearly heating or cooling load, $E_{lights}(x)$ is the zone's electricity consumption for lighting, and the efficiencies $\eta_{heat} = 0.44$ and $\eta_{cool} = 0.77$ are typical plant efficiencies that relate the zone load with the source energy consumption for heating and cooling generation, including electricity consumption for fans and pumps. Lighting electricity (as well as electricity for fans and pumps) is weighted by a factor 3 to convert site electricity to source fuel energy consumption. The efficiency values were obtained from Huang and Franconi (1999) and are typical for large office buildings in Chicago, IL.

There are four design parameters: The building azimuth, α, the width of the west and east windows, w_w and w_e, respectively, and the shading device transmittance, τ.

Given a set of design parameters, x, and an objective function, $f(\cdot)$, how much improvement can be achieved depends on the initial values of x. The initial values in this example are selected to correspond to a "good" energy design. In particular, we think that the window width is a good trade-off between allowing enough daylight usage while keeping the solar gain low.

The optimization is done using the Hooke-Jeeves algorithm. Table 1 shows the initial values and the optimized values of the parameters, together with the change of each term of (2), and Fig. 5 shows the values at each iteration step. To achieve the minimum point 117 iterations were required. Since the algorithm requires the objective function value at some points more than once (GenOpt detects such cases and, hence, does not perform a simulation run), a total of 98 EnergyPlus simulations were required. This took 4 hours on a $200\mathrm{MHz}$ Pentium computer running Windows NT 4. The optimization reduced $E_{tot}(x)$ by 14%. Parametric runs showed that the major reduction of $E_{tot}(x)$ is due to the larger window area and not due to the building rotation.

Domestic Hot Water Solar System

The goal of this example is to minimize the yearly auxiliary electric consumption of the domestic hot water solar system shown in Fig. 6. An electrical heater, located in the tank, reheats the water to $55°\mathrm{C}$ if the collector cannot maintain this temperature. Weather data from Copenhagen, Denmark, are used. There are four design parameters: The azimuth and

	lower bound	initial value	upper bound	optimum value
Annual source energy consumption, E_{tot} [kWh/(m^2 a)]	-	168.1 (100%)	-	145.3 (86%)
Annual heating energy consumption, Q_{heat} [kWh/(m^2 a)]	-	9.86 (100%)	-	7.14 (72%)
Annual cooling energy consumption, Q_{cool} [kWh/(m^2 a)]	-	32.0 (100%)	-	34.5 (108%)
Annual lighting energy consumption, E_{lights} [kWh/(m^2 a)]	-	34.7 (100%)	-	28.0 (81%)
Building azimuth, α [deg]	$-\infty$	0	∞	90
Width of west window, w_w [m]	0.1	3	5.9	5.1
Width of east window, w_e [m]	0.1	3	5.9	5.9
Shading device transmittance, τ [$-$]	0.2	0.5	0.8	0.45

Table 1: Lower bound, initial values, upper bounds and final values of optimization for office zone

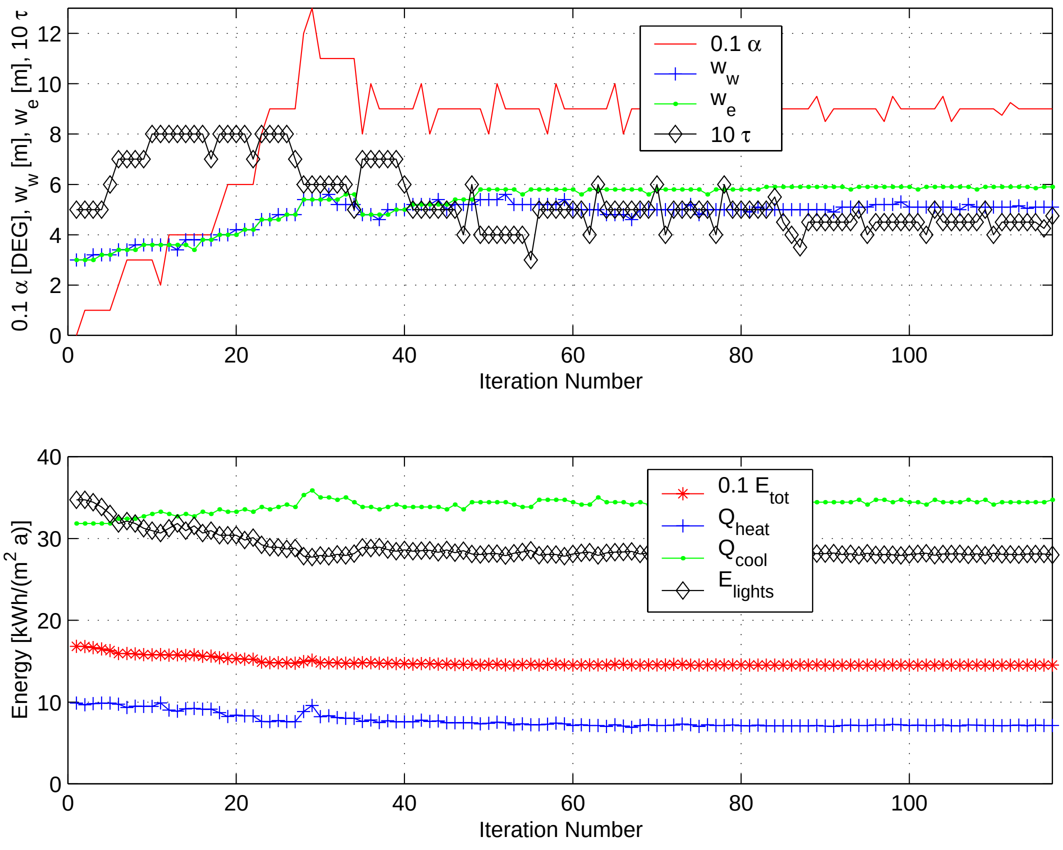


Figure 5: Iteration sequence of the Hooke-Jeeves optimization algorithm

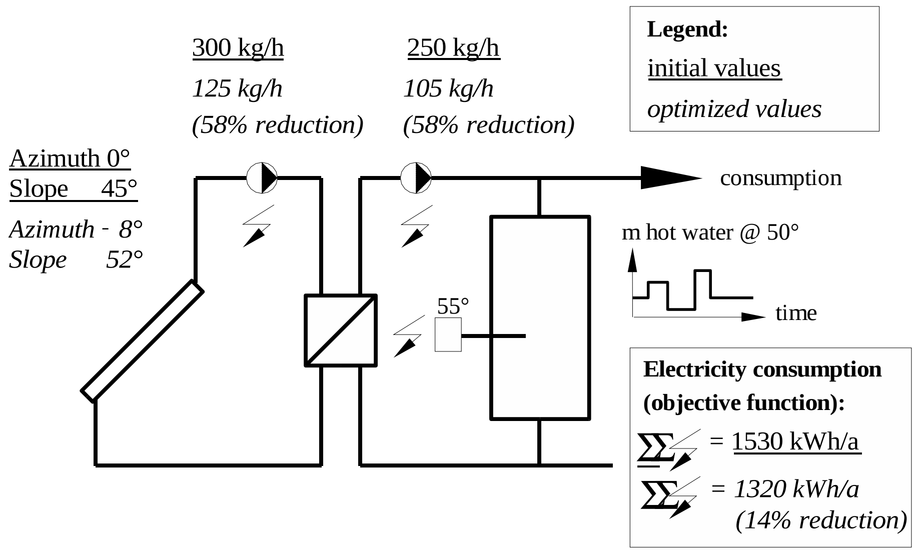


Figure 6: Schematic of the domestic hot water solar system

slope of the collector, and the mass flow of the collector and the tank. A given hot water consumption profile is maintained. The optimization is done using the Simplex algorithm of Nelder and Mead, with initial values as found in the TRNSYS 14.2 example file `soldemo.hgf`.

As shown in Fig. 6, the optimization yields a 14% reduction of the yearly auxiliary electric consumption. Further, it allows reducing the mass flow of the circulation pumps by 58%, which results in lower first costs. To achieve the minimum GenOpt required 250 TRNSYS function evaluations, using about 6 hours of CPU time on a Pentium 200 MHz computer.

CONCLUSION

GenOpt's simulation program interface makes it easy to couple any simulation program to GenOpt by simply modifying a configuration file. No code modifications of either program are required. Minimization algorithms can be added to GenOpt's library by extending a superclass that provides an interface to GenOpt's kernel. This means that new minimization algorithms can be implemented without having to deal with the tedious programming of input retrieval, data management, etc. GenOpt also offers a mathematical library for methods used by different minimization algorithms, which further shortens development time of new algorithms.

GenOpt has been successfully applied to various problems in thermal building simulation. Given a simulation model, the time required to set up an optimization problem is typically less than an hour. Hence, the labor cost for doing the optimization is usually paid back by the lower operating cost of the optimized system.

Research is in progress on how to reduce optimization time by means of *algorithm implementation* or by *consistent approximations* (Polak 1997) and on how to ensure convergence to a stationary point.

ACKNOWLEDGMENTS

This work was supported by the U.S. Department of Energy (DOE), the Swiss Academy of Engineering Sciences (SATW), the Swiss National Energy Fund (NEFF), and the Swiss National Science Foundation (SNF).

REFERENCES

Bell, M. and M.C. Pike. 1966. “Remark on algorithm 178.” *Comm. ACM* 9 (Sep.): 685–686.

De Vogelaere, R. 1968. “Remark on Algorithm 178.” *Comm. ACM* 11 (Jul.): 498.

Hooke, R. and T.A. Jeeves. 1961. “’Direct search’ solution of numerical and statistical problems.” *J. Assoc. Comp. Mach.* 8 (2): 212–229 (Apr.).

Huang, Joe and Ellen Franconi. 1999, Nov. “Commercial Heating and Cooling Loads Component Analysis.” Technical Report LBL-37208, Lawrence Berkeley National Laboratory, EETD.

Nelder, J. A. and R. Mead. 1965. “Simplex Method for Function Minimization.” *The Computer Journal* 7 (4): 308–13 (Jan.).

O’Neill, R. 1971. “Algorithm AS 47–Function Minimization Using a Simplex Procedure.” *Appl. Stat.* 20:338–45.

Polak, Elijah. 1997. *Optimization, Algorithms and Consistent Approximations.* Volume 124 of *Applied Mathematical Sciences.* Springer Verlag.

Smith, Lyle B. 1969. “Remark on algorithm 178.” *Comm. ACM* 12 (Nov.): 638.

NOMENCLATURE

E_{lights}	yearly electricity consumption for lighting
E_{tot}	total yearly source energy consumption
$f(\cdot)$	objective function, $f\colon \mathbb{R}^n \to \mathbb{R}$
l	lower bound of design parameter
n	dimension of design parameter
Q_{cool}	yearly cooling load
Q_{heat}	yearly heating load
u	upper bound of design parameter
w_w	width of west window
w_e	width of east window
x	design parameter
$\mathbf{X}$	feasible set of design parameter
α	building azimuth ($\alpha = 90^\circ$ means that the west facade is now facing north)
η_{cool}	plant efficiency for cooling
η_{heat}	plant efficiency for heating
τ	shading device transmittance for visible, solar, and long-wave radiation
$\mathbb{R}$	set of real numbers
$\triangleq$	equal by definition