Simulation-Based Building Energy Optimization

by

Michael Wetter

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering — Mechanical Engineering

in the

GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Prof. Elijah Polak, Co-chair
Prof. Van P. Carey, Co-chair
Prof. Alice M. Agogino
Prof. Alexandre J. Chorin

Spring 2004

The dissertation of Michael Wetter is approved:

Co-chair Date

Co-chair Date

Date

Date

University of California, Berkeley

Spring 2004

Simulation-Based Building Energy Optimization

Abstract

Simulation-Based Building Energy Optimization

by

Michael Wetter

Doctor of Philosophy in Engineering — Mechanical Engineering

University of California, Berkeley

Prof. Elijah Polak, Co-chair

Prof. Van P. Carey, Co-chair

This dissertation presents computational techniques for simulation-based design optimization of buildings and heating, ventilation, air-conditioning and lighting systems in which the cost function is smooth. In such problems, the evaluation of the cost function involves the numerical solution of systems of differential algebraic equations (DAE). Since the termination criteria of the iterative solvers often depend on the design parameters, a computer code for solving such systems usually defines a numerical approximation to the cost function that is discontinuous in the design parameters. The discontinuities can be large in cost functions that are evaluated by commercial building energy simulation programs, and optimization algorithms that require smoothness frequently fail if used with such programs. Furthermore, controlling the numerical approximation

error is often not possible with commercial building energy simulation programs.

In this dissertation, we present BuildOpt, a new detailed thermal building and daylighting simulation program. BuildOpt's simulation models define a DAE system that is smooth in the state variables, in time and in the design parameters. This allows proving that the DAE system has a unique solution that is smooth in the design parameters, and it is required to compute high precision approximating cost functions that converge to a cost function that is smooth in the design parameters as the DAE solver tolerance is tightened.

For simulation programs that allow such a precision control, we constructed subprocedures for Generalized Pattern Search (GPS) optimization algorithms that adaptively control the precision of the cost function evaluations: coarse precision for the early iterations, with precision progressively increasing as a stationary point is approached. This scheme significantly reduces the computation time, and it allows to prove that the sequence of iterates contains stationary accumulation points.

For optimization problems in which commercial building energy simulation programs are used to evaluate the cost function, we compared by numerical experiment several deterministic and probabilistic optimization algorithms.

Co-chair Date

Co-chair Date

To Maureen.

To my parents.

Contents

List of Figures

List of Tables

Conventions and Symbols

Numbering and Cross-Referencing System

The following system of numbering and cross-referencing is used.

Definitions, assumptions, lemmas, propositions, theorems, corollaries and remarks are numbered in order of occurrence and using a three number system *(a.b.c)*, where a is the chapter number, b is the section number, and c is the item number. This numbering system does not distinguish between definitions, assumptions, lemmas, etc.

Within each section, equations are numbered consecutively, using a single number system, and are referred to by a three number system *(a.b.c)*, where a is the chapter number, b is the section number, and c is the item number.

Conventions

1. $\mathbb{R}^n$ denotes the Euclidean space of n-tuplets of real numbers. Vectors $x \in \mathbb{R}^n$ are always column vectors, and their elements are denoted by superscripts. The inner product in $\mathbb{R}^n$ is denoted by $\langle \cdot, \cdot \rangle$ and for $x, y \in \mathbb{R}^n$ it is defined by $\langle x, y \rangle \triangleq \sum_{i=1}^{n} x^i y^i$. The norm in $\mathbb{R}^n$ is denoted by $\| \cdot \|$ and for $x \in \mathbb{R}^n$ defined by $\|x\| \triangleq \langle x, x \rangle^{1/2}$.

2. We denote by $\mathbb{Z}$ the set of integers, by $\mathbb{Q}$ the set of rational numbers, and by $\mathbb{N} \triangleq \{0, 1, \ldots\}$ the set of natural numbers. The set $\mathbb{N}_+$ is defined as $\mathbb{N}_+ \triangleq \{1, 2, \ldots\}$.

Similarly, vectors in $\mathbb{R}^n$ with strictly positive elements are denoted by $\mathbb{R}^n_+ \triangleq \{x \in \mathbb{R}^n \mid x^i > 0,\ i \in \{1,\ldots,n\}\}$ and the set $\mathbb{Q}_+$ is defined as $\mathbb{Q}_+ \triangleq \{q \in \mathbb{Q} \mid q > 0\}$.

3. For $\varepsilon \in \mathbb{R}^q_+$, by $\varepsilon \leq \varepsilon_{\mathbf{S}}$, we mean that $0 < \varepsilon^i \leq \varepsilon^i_{\mathbf{S}}$, for all $i \in \{1, \ldots, q\}$.

4. $f(\cdot)$ denotes a function where $(\cdot)$ stands for the undesignated variables. $f(x)$ denotes the value of $f(\cdot)$ for the argument x. $f\colon A \to B$ indicates that the domain of $f(\cdot)$ is in the space A, and that the image of $f(\cdot)$ is in the space B.

5. We say that a function $f : \mathbb{R}^n \to \mathbb{R}^m$ is Lipschitz continuous on a set $\mathbf{S} \subset \mathbb{R}^n$, with respect to (w.r.t.) $x \in \mathbf{S}$, if $f(\cdot)$ is defined on $\mathbf{S}$, and if there exists a constant $L \in (0,\infty)$ such that $\|f(x') - f(x'')\| \leq L\|x' - x''\|$, for all $x', x'' \in \mathbf{S}$.

6. We say that a function $f : \mathbb{R}^n \to \mathbb{R}$ is k-times (Lipschitz) continuously differentiable on a set $\mathbf{S} \subset \mathbb{R}^n$, with respect to (w.r.t.) $x \in \mathbf{S}$, if $f(\cdot)$ is defined on $\mathbf{S}$, and if $f(\cdot)$ has k (Lipschitz) continuous derivatives on $\mathbf{S}$.

7. If a subsequence $\{x_i\}_{i\in\mathbf{K}} \subset \{x_i\}_{i=0}^{\infty}$ converges to some point x, we write $x_i \to^{\mathbf{K}} x$.

8. Let $\mathbb{W}$ be a set containing a sequence $\{w_i\}_{i=0}^k$. Then, we denote by $\underline{w}_k$ the sequence $\{w_i\}_{i=0}^k$ and by $\underline{\mathbf{W}}_k$ the set of all $k+1$ element sequences in $\mathbb{W}$.

9. We denote by $\{e_i\}_{i=1}^n$ the unit vectors in $\mathbb{R}^n$.

10. If $\mathbf{X}$ is a set, we denote by $\partial\mathbf{X}$ its boundary.

11. If $\mathbf{S}$ is a set, we denote by $2^{\mathbf{S}}$ the set of all nonempty subsets of $\mathbf{S}$.

12. If $D \in \mathbb{Q}^{n\times q}$ is a matrix, we will use the notation $d \in D$ to denote the fact that $d \in \mathbb{Q}^n$ is a column vector of the matrix D.

13. For $s \in \mathbb{R}$, we define $\lfloor s \rfloor \triangleq \max\{k \in \mathbb{Z} \mid k \leq s\}$ and $\lceil s \rceil \triangleq \min\{k \in \mathbb{Z} \mid k \geq s\}$.

14. We denote by $H\colon \mathbb{R} \to \mathbb{R}$ the Heaviside function, defined by

$$H(x) \triangleq \begin{cases} 0, & \text{for } x < 0, \\ 1, & \text{for } x \geq 0. \end{cases}$$

15. For $s,t \in \mathbb{R}$ and $f\colon \mathbb{R} \to \mathbb{R}$, by $\lim_{s\downarrow t} f(s)$, we mean $\lim_{s\to t} f(s)$ with $s > t$.

Symbols

Sets

$2^{\mathbf{S}}$	set of all non-empty subsets of $\mathbf{S}$
$\mathbf{L}_\alpha(f)$	level set of $f(\cdot)$
$\mathbb{N}$	$\{0,1,2,\ldots\}$
$\mathbb{Q}$	set of rational numbers
$\mathbb{Q}_+$	$\{q \in \mathbb{Q} \mid q > 0\}$
$\mathbb{R}$	set of real numbers
$\mathbb{R}^q_+$	$\{x \in \mathbb{R}^q \mid x^i > 0,\ i \in \{1,\ldots,q\}\}$
$\mathbb{Z}$	$\{\ldots,-2,-1,0,1,2,\ldots\}$

Functions

$f(\cdot)$	cost function
$f^*(\cdot,\cdot)$	approximating cost function
$\lfloor s \rfloor$	$\max\{k \in \mathbb{Z} \mid k \leq s\}$
$\lceil s \rceil$	$\min\{k \in \mathbb{Z} \mid k \geq s\}$
$\mathrm{card}(\cdot)$	cardinality of a set
$df(x;h)$	directional derivative
$d^0 f(x;h)$	Clarke's generalized directional derivative

Sequences

$\underline{w}_k$	$\{w_i\}_{i=0}^k$
$x_i \rightarrow^{\mathbf{K}} x$	$\{x_i\}_{i\in\mathbf{K}} \subset \{x_i\}_{i=0}^\infty$ converges to x

Miscellaneous

$\triangleq$	equal by definition
$\square$	end of proof, example, assumption, etc.

Acknowledgments

I would like to extend special thanks to Professor Elijah Polak for his guidance and patience over the last five years. His kind support has been key to my academic development, and his research style has had a profound influence on my work.

I am also grateful to Professor Van P. Carey and Professor Alice M. Agogino for their advice, and to Professor Alexandre J. Chorin for his guidance and advice regarding the numerical solutions of systems of differential equations.

I am indebted to Frederick Winkelmann; his support and trust were instrumental in completing this work. My thanks also go to Dimitri Curtil for his advice in C++, which was of great help in developing BuildOpt, to Ender Erdem for fixing everything that has to do with computers, whether it was related to software or hardware, to Kathy Ellington for her assistance, from getting me settled in Berkeley to editing the countless manuscripts, to Fred Buhl for looking at several EnergyPlus problems that we encountered in running our numerical experiments and to Bill Carroll for his assistance in daylighting modeling.

Thanks to Jonathan Wright for implementing the Genetic Algorithm that is used in this work and for his assistance in comparing deterministic and probabilistic optimization algorithms.

This research was supported by the Swiss Academy of Engineering Sciences (SATW), the Swiss National Energy Fund (NEFF), the Swiss National Science Foundation (SNF) and by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of

Building Technology Programs of the U.S. Department of Energy, under Contract No. DE-AC03-76SF00098. I would like to thank these institutions for their generous support.

I am thankful to my family and friends for their support and encouragement, in particular to my parents who always supported me in my plans. Special thanks go to Maureen O'Sullivan for providing me with strength and continuous support through the ups and downs of writing a dissertation.

Chapter 1

Introduction

1.1 Problem Discussion

1.1.1 Optimization Problem

In designing building envelope and heating, ventilation and air-conditioning (HVAC) systems, one generally attempts to find the values of a vector of design parameters that yield optimal system performance, subject to some architectural and comfort constraints. The system performance can be measured by a so-called *cost function*. Examples of cost functions are the annual energy consumption, the annual energy cost and the life cycle cost of a building.

In this dissertation, we will consider design optimization problems in which the components of the vector of design parameters can take on any real number, possibly bounded by a finite lower and a finite upper bound. Also, we assume that architectural constraints can be specified by lower and upper bounds on the design parameters (such as a minimal and maximal window area) and that comfort constraints can be met by selecting an appropriate control law for the HVAC and the lighting system. In this situation, the problem of finding an optimal building and HVAC design can be described formally by the optimization problem

$$\mathbf{P} \qquad \min_{x \in \mathbf{X}} f(x), \tag{1.1.1a}$$

where $\mathbf{X} \subset \mathbb{R}^n$ is the constraint set, defined as

$$\mathbf{X} \triangleq \big\{x \in \mathbb{R}^n \mid l^i \leq x^i \leq u^i,\ i \in \{1,\ldots,n\}\big\}, \tag{1.1.1b}$$

with $-\infty \leq l^i < u^i \leq \infty$ for all $i \in \{1,\ldots,n\}$, and the cost function is

$$f(x) \triangleq F(z(x,1)), \tag{1.1.2}$$

where $F\colon \mathbb{R}^m \to \mathbb{R}$ is once continuously differentiable[1] and $z(x,1) \in \mathbb{R}^m$ is a vector of state variables. The components of $z(\cdot,\cdot)$ can be the energy consumption for lighting, cooling and heating, the building's room air temperatures and construction temperatures at specified locations in the walls, floors and ceilings. As we will show in Chapter 2, the vector of state variables $z(x,1)$ can be expressed as the solution of a semi-explicit nonlinear DAE system with index one (Brenan et al., 1989) of the form

$$\frac{dz(x,t)}{dt} = h\big(x,z(x,t),\mu\big), \qquad t \in [0,\ 1], \tag{1.1.3a}$$

$$z(x,0) = z_0(x), \tag{1.1.3b}$$

$$\gamma\big(x,z(x,t),\mu\big) = 0, \tag{1.1.3c}$$

where $h\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^m$, $z_0\colon \mathbb{R}^n \to \mathbb{R}^m$ and $\gamma\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^l$. We discuss

[1] In case of minimizing the annual peak electrical demand, $F\colon \mathbb{R}^m \to \mathbb{R}$ is not continuously differentiable. However, in this dissertation we consider only problems in which $F(\cdot)$ is once continuously differentiable.

this DAE system in more detail in Chapter 2, in which we show that under appropriate assumptions, equation (1.1.3) has for all $x \in \mathbf{X}$ a unique solution $z(x,1)$ that is once continuously differentiable in x, from which follows that $f(\cdot)$ is once continuously differentiable.

1.1.2 Approximating Optimization Problems

In building design optimization, as well as in many other multidisciplinary optimization problems, the solution $z(\cdot,1)$ can only be approximated numerically using time-intensive computer simulations that are done using computer simulation programs that consist of many thousand lines of code. Those computer simulation programs generally contain adaptive solvers. Examples of adaptive solvers are variable time step integration routines, Newton-Raphson solvers for non-linear systems of equations, Gauss-Seidel solvers for large linear systems of equations, and mesh generators that discretize the domain on which partial differential equations are defined. Those adaptive solvers cause the smooth cost function $f(\cdot)$ to be replaced with an approximating cost function $f^*(\varepsilon,\cdot) \triangleq F(z^*(\varepsilon,\cdot,1))$, where $\varepsilon \in \mathbb{R}^q_+$, for some fixed $q \in \mathbb{N}$, denotes the vector that contains the tolerance settings of the adaptive solvers, and $z^*(\varepsilon,\cdot,1)$ is the numerical approximation to $z(\cdot,1)$. Because the adaptive solvers can cause the sequence of code executions to change if the design parameter x is perturbed, the approximate solutions $\{z^*(\varepsilon,\cdot,1)\}_{\varepsilon\in\mathbb{R}^q_+}$, and hence also the approximating cost functions $\{f^*(\varepsilon,\cdot)\}_{\varepsilon\in\mathbb{R}^q_+}$, are discontinuous functions of the design parameter. Thus, the adaptive solvers cause the

problem **P**, defined in (1.1.1), to be replaced by an approximating optimization problem, that is parametrized by the tolerance settings of the adaptive solvers $\varepsilon \in \mathbb{R}^q_+$, and which is of the form

$$\mathbf{P}_\varepsilon \qquad \min_{x \in \mathbf{X}} f^*(\varepsilon, x), \tag{1.1.4}$$

where $f^* : \mathbb{R}^q_+ \times \mathbb{R}^n \to \mathbb{R}$.[2]

When moderate precision is used in evaluating the cost function, it is not uncommon that optimization algorithms which require smoothness of the cost function jam at discontinuities of $f^*(\varepsilon, \cdot)$. Hence, in solving $\mathbf{P}_\varepsilon$, using moderate precision does not appear to be appropriate, while using high precision does not appear to be practical because high precision cost function evaluations are computationally expensive.

1.1.3 Commercial Building Energy Simulation Programs

Many commercial building energy simulation programs do not allow the adjustment of all components of $\varepsilon \in \mathbb{R}^q_+$, and the discontinuities in $f^*(\varepsilon, \cdot)$ can be large. For example, in our experiments, the commercial whole building energy analysis program EnergyPlus (Crawley et al., 2001) became unstable as we decreased ε, and it did not seem possible to construct a rule for decreasing ε so that the numerical error, which was in the order of 2% of the cost function value, could be made small. To illustrate why it did not seem

[2]Because $f^*(\varepsilon, \cdot)$ is discontinuous, it may not attain a minimum, even on compact sets. Thus, to be correct, $\min_{x \in \mathbf{X}} f^*(\varepsilon, x)$ should be replaced by $\inf_{x \in \mathbf{X}} f^*(\varepsilon, x)$. For simplicity, we will not make this distinction.

possible to control the approximation error with EnergyPlus, we note that EnergyPlus is built on nonsmooth models which makes it hard, if not impossible, for numerical solvers to converge to a solution. The source code consists of $200,000$ lines of Fortran code and hence reformulating the models in order to make them smooth is impractical. There are 10 to 20 numerical solvers which were implemented ad-hoc, whose tolerance are in most cases fixed at compile time, and which are in some cases coupled to each other using heuristic rules that were deduced by numerical experiments.[3] By the time of this writing, about 70 man years were invested in the development of EnergyPlus, which is built on code of the two simulation programs DOE-2 (Winkelmann et al., 1993) and BLAST (BLAST, 1999).

1.2 Objective of the Dissertation

The objective of this dissertation is to develop a technique for writing building energy simulation programs that can approximate $f(\cdot)$ with adaptive precision, and to develop optimization algorithms with adaptive precision cost function evaluations, so that it is possible to prove that the optimization converges to a stationary point of $f(\cdot)$.

In addition, because considerable investment has been done to develop building energy simulation programs, another objective is to test which optimization algorithms are

[3]For example, in EnergyPlus, heuristic rules are used in the computation of the heat conduction in solids. The algorithm for computing heat conduction in solids is so sensitive to numerical instability that it is implemented in IP units rather than in SI units, which has been shown experimentally to be numerically less prone to instabilities.

likely to obtain good results in optimizing $\mathbf{P}_\varepsilon$ in situations where the simulation program does not allow controlling the error $|f^*(\varepsilon, x) - f(x)|$.

1.3 Market for Building and HVAC Design Optimization

In the literature (Al-Homoud, 1997; Wetter and Wright, 2003a,b), savings of 5% to 30% in annual energy consumption for lighting, cooling and heating due to optimized building and HVAC design has been reported.

To obtain an estimate of the return on investment for an office building with $10,000\,\mathrm{m}^2$ floor area, assume that the average cost savings due to optimized building and HVAC design are 15%, that the average energy cost is \$0.10 per kWh, and that the annual energy consumption is $200\,\mathrm{kWh}/(\mathrm{m}^2\,\mathrm{a})$. Then, the savings due to optimized building and HVAC design are \$30,000 per year. As large buildings are often designed using energy simulations, and hence a computer simulation model exists for those buildings, the additional effort to do an optimization is only a few man hours. Thus, the return on investment is achieved within the first year of building operation.

The market demand for simulation-based building and HVAC design optimization is also attested by the fact that more than 1,000 users registered for a license for the GenOpt$^{(R)}$ program, which is an optimization program that was conceived and devel-

oped by the author at LBNL to solve building and HVAC design optimization problems (Wetter, 2001, 2004).

1.4 Review of State-of-the-Art

Because the applicability of optimization algorithms for solving problem **P** depends on the properties of the cost function $f(\cdot)$ and its numerical approximations $\{f^*(\varepsilon,\cdot)\}_{\varepsilon\in\mathbb{R}^q_+}$, we will present a review of the state-of-the-art in the following order: First, we discuss the applicability of building energy simulation programs for use with optimization algorithms that require the cost function to be smooth. Next, we present optimization techniques for problems where the approximation error $|f^*(\varepsilon,x)-f(x)|$ can be controlled in such a way that $f^*(\varepsilon,\cdot)$ converges to a smooth function $f(\cdot)$, as $\varepsilon\to 0$. Next, we present optimization algorithms that do not control ε, but which are frequently applied to problem $\mathbf{P}_\varepsilon$ in a heuristic context. The last point addresses situations in which existing multi-disciplinary simulation programs need to be used to evaluate the cost function. Finally, we report applications of building and HVAC design optimizations.

1.4.1 Building Energy Simulation Programs

Existing building energy simulation programs, such as EnergyPlus (Crawley et al., 2001), TRNSYS (Klein et al., 1976), ESP-r (Clarke, 2001), DOE-2 (Winkelmann et al., 1993) and IDA-ICE (Björsell et al., 1999; Sahlin and Bring, 1991) have been developed

in such a way that it is not possible to prove that their numerical approximations $z^*(\varepsilon, \cdot, 1)$ converge to a function $z(\cdot, 1)$ that is once continuously differentiable as the tolerance of the numerical solvers is tightened. Convergence to a smooth function cannot be proven because those simulation programs are built on models that define the time rate of change $dz(\cdot,t)/dt$ of equation (1.1.3) by functions that are non-smooth in the state variables and in the building and HVAC design parameters. Because $z(\cdot, 1)$ is not differentiable, $\nabla f(x) = 0$ is not defined, which in turn makes it impossible to construct optimization algorithms for which convergence to a stationary point of $f(\cdot)$ can be proven.

Furthermore, in many existing building energy simulation programs, those solvers for which ε can be adjusted frequently fail to compute high precision approximating solutions. Lack of convergence of those solvers may be attributed to the fact that those solvers are typically designed based on Taylor expansions (such as a Newton solver), and hence the algorithms in those solvers were built on the assumption that they are used to solve differentiable equations. Thus, if the equations are not differentiable, the solvers can fail, particularly if the solver tolerance is tight (cf. Tab. 2.1 on page 40).

It is worth a mention that a promising simulation program for use with optimization algorithms with adaptive precision cost function evaluations is the IDA-ICE program. IDA-ICE generates from an equation-based modeling language, the so-called *Neutral Model Format* (Sahlin and Sowell, 1989), computer code that defines the residual equations of a DAE system which it solves by computing simultaneous solutions for all equations. However, because IDA-ICE is also built on non-smooth models, it can

only be assessed by doing numerical experiments how IDA-ICE performs in conjunction with optimizations algorithms that adaptively control the precision of the cost function evaluations.

As there is no detailed building energy simulation program available that allows computing high precision approximate state variables $z^*(\varepsilon, \cdot, 1)$ so that they converge to a function $z(\cdot, 1)$ that is once continuously differentiable, we developed BuildOpt, a new thermal building and daylighting simulation program, that we present in Chapter 2.

1.4.2 Optimization with Adaptive Precision Cost Function Evaluations

Polak (1997) presents several algorithm models that adaptively control the precision of the approximating cost function in the course of the optimization. Most algorithm models in Polak (1997) are not applicable if $f^*(\varepsilon, \cdot)$ is discontinuous. The only algorithm in Polak (1997) that is applicable if $f^*(\varepsilon, \cdot)$ is discontinuous is Master Algorithm Model 1.2.36. The Master Algorithm Model 1.2.36 states a general framework of how precision can be controlled so that the sequence of iterates converges to a stationary point of $f(\cdot)$.

1.4.3 Optimization with Fixed Precision Cost Function Evaluations

We will now discuss a few optimization algorithms that are frequently cited in the literature to solve heuristically problem $\mathbf{P}_\varepsilon$, with fixed ε.

A family of optimization algorithms that is frequently applied to $\mathbf{P}_\varepsilon$ is the family of Generalized Pattern Search (GPS) optimization algorithms. Examples of pattern search algorithms are the coordinate search algorithm (Polak, 1971), the pattern search algorithm of Hooke and Jeeves (1961), and the multidirectional search algorithm of Dennis and Torczon (1991). Torczon (1997) proved for problem **P** that $\nabla f(\cdot)$ vanishes at accumulation points of sequences constructed by GPS algorithms. Audet and Dennis (2003) present a simpler abstraction of GPS algorithms and a convergence analysis that is based on Clarke's generalized directional derivative (Clarke, 1990). Audet and Dennis (2003) regain the results of Torczon (1997). Audet and Dennis (2000a) extended GPS algorithms for mixed variable programming. A filter method for GPS algorithms for the solution of problem **P** with nonlinear inequality constraints $g(x) \leq 0$ was presented by Audet and Dennis (2000b). Abramson (2002) combines the work of Audet and Dennis to construct a GPS algorithm for mixed variable programming with nonlinear inequality constraints. Kolda et al. (2003) present a review of pattern search algorithms.

The Implicit Filtering algorithm (Choi and Kelley, 2000; Kelley, 1999b) has been developed to solve optimization problems in which only discontinuous approximating cost functions $f^*(\varepsilon, \cdot)$ are available. In its simplest form, the Implicit Filtering algorithm

is an implementation of the Steepest Descent algorithm with finite difference approximation to the gradient and Armijo line search. The Implicit Filtering algorithm defines a rule for reducing the finite difference increment in the course of the optimization. It has been successfully applied to solve various engineering optimization problems (for a list of problems, see for example Choi and Kelley (2000)). However, the error of the cost function evaluations is not controlled in Implicit Filtering. If the error of the cost function evaluations decays faster to zero than the step size used in the finite difference approximation to $\nabla f(\cdot)$, then convergence to a stationary point of $f(\cdot)$ can be proven.

A framework for managing models in nonlinear optimization of computationally expensive cost functions is presented in Serafini (1998) and in Booker et al. (1999). It defines a rule for adaptively refining in region of interest a computationally cheap model that is constructed based on sample points of $f(\cdot)$ during the optimization. The optimization is done on the computationally cheap model, and if no further cost reduction can be found on the model, then additional function values of $f(\cdot)$ are sampled and used to check the progress of the optimization of $f(\cdot)$ and to update the model. Under the assumption that $f(\cdot)$ is smooth and can be evaluated exactly, the model management framework guarantees that the search on the model convergences to a stationary point of $f(\cdot)$. The model management is typically used with GPS algorithms and has been applied successfully to solve engineering optimization problems (see for example Booker et al. (1998) or, for an application with nonlinear inequality constraints $g(x) \leq 0$, Marsden et al. (2004)).

The DIRECT (DIviding RECTangles) optimization algorithm is a global sampling method that divides the search space into rectangles in an effort to move toward an optimum (Finkel, 2003; Gablonsky and Kelley, 2001; Perttunen et al., 1993). It has been developed for bound constrained optimization of nondifferentiable cost functions. Gablonsky and Kelley (2001) report that there is little convergence theory for the DIRECT algorithm beyond the observation from Perttunen et al. (1993) that the search will eventually sample arbitrarily near every point in the search space. Gablonsky and Kelley (2001) also report that the method has been applied to the optimal design of gas pipelines and aerospace engineering, and that it seems to perform well, especially in the early stages of the optimization.

The Simplex algorithm from Nelder and Mead is frequently applied to problems of the form $\mathbf{P}_\varepsilon$. In its original form, it can fail to find a minimum even for problem **P** with smooth cost function (see for example Kelley (1999b), Torczon (1989), Kelley (1999a), Wright (1996), McKinnon (1998) and Lagarias et al. (1998)), both in practice and in theory, particularly if the dimension of the vector of design parameters is large, say bigger than 10 (Torczon, 1989). Several improvements to the Simplex algorithm or algorithms that were motivated by the Simplex algorithm exist, see for example Kelley (1999a,b), Torczon (1989) and Tseng (1999).

1.4.4 Building and HVAC Design Optimization

We will now discuss simulation-based building and HVAC design optimization problems in which the cost function evaluation requires a complex building energy simulation that is done by a detailed simulation program, such as EnergyPlus. The reason for focusing on those problems is that they require cost function evaluations that are computationally expensive and that in detailed simulations, many adaptive solvers and mesh generators are used which introduce discontinuities in $f^*(\varepsilon, \cdot)$ that can be large. Thus, optimization algorithms that perform well if the cost function is evaluated by a simple simulation model may not perform well in situations where the cost function is evaluated by a detailed simulation model and hence are not discussed here.

Most annual building energy optimizations that use a detailed simulation model are solved using a Genetic Algorithm (GA). GAs seem to be popular because they are easy to implement, they do not require smoothness of the approximating cost function, they can take into account discontinuous design parameters and their population-based search makes it easy to use them for multi-criteria optimization problems. However, to achieve convergence to a minimizer with high probability, GAs require a large number of cost function evaluations and despite their frequent use, they are not necessarily the best choice if $f^*(\varepsilon, \cdot)$ is defined on $\mathbb{R}^n$, as our experiments in Chapter 4 show.

Wright and Loosemore (2001) and Wright et al. (2002) used GAs for the minimization of annual energy consumption. However, because a huge number of cost function

evaluations was required, they simulated only a few typical design days to reduce the computation time.

Caldas and Norford (2002) developed a building design tool that is based on a GA. To reduce the number of cost function evaluations, they use a micro-GA (Krishnakumar, 1989).

In Choudhary et al. (2003) and Choudhary (2004), a hierarchical optimization framework for simulation-based architectural design is proposed. The method is based on Analytical Target Cascading (Kim et al., 2003), which is a system design approach enabling top level design targets to be cascaded down to lower levels of the modeling hierarchy. In the lower level optimization, the computationally expensive cost function, which is in the cited literature defined by an EnergyPlus simulation model, is approximated by a computationally cheap surrogate function. At the higher system level, sequential quadratic programming is used to solve an optimization problem with smooth cost function.

1.5 Proposed New Approach

We will first propose an optimization technique that uses adaptive precision cost function evaluations, and then address the situation where existing simulation programs which do not allow controlling $|f^*(\varepsilon, x) - f(x)|$ need to be used to evaluate the cost function.

1.5.1 Optimization with Adaptive Precision Cost Function Evaluations

1.5.1.1 Smoothness of Cost Function

As we will show in Chapter 2 and in Appendix A, it is possible to implement the DAE system (1.1.3) using models that are smooth in the state variables, in time and in the design parameter. Using smooth models allows proving for the DAE system defined in (1.1.3) that a solution $z(\cdot,1)$ exists and furthermore that $z(\cdot,1)$ is unique and once continuously differentiable. The use of smooth models is also required for the DAE solver to converge during the computation of high precision approximations for the once continuously differentiable cost function $f(\cdot)$ (cf. Tab. 2.1 on page 40).

However, computing high precision approximating cost functions $f^*(\varepsilon,\cdot)$ is computationally expensive. For example, for the numerical experiments that we present in Chapter 2 and in which we used BuildOpt to evaluate the cost function, computing $f^*(10^{-1},\cdot)$ was one hundred times faster than computing $f^*(10^{-5},\cdot)$. The normalized computation time for an annual building energy simulation as a function of the DAE solver tolerance $\varepsilon \in \mathbb{R}_+$ is shown in Fig. 1.1, which was generated using BuildOpt and the simulation model presented in Chapter 2.

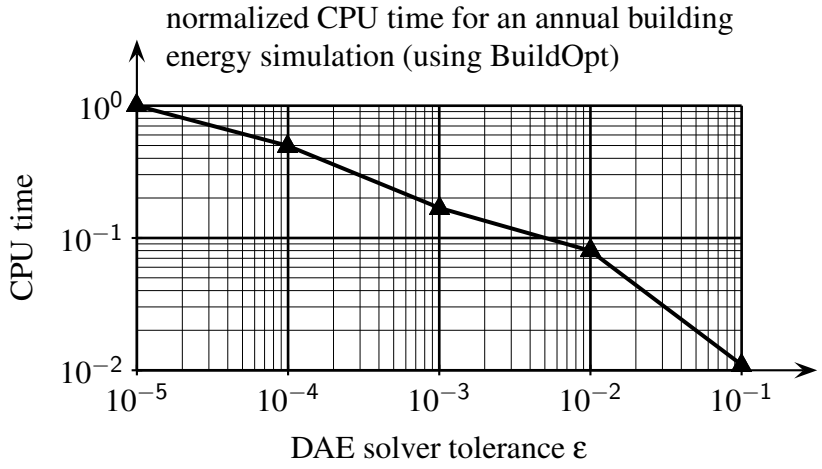


Figure 1.1: Normalized computation time for an annual building energy simulation as a function of the DAE solver tolerance ε.

1.5.1.2 Diagonalization Scheme

In view of Fig. 1.1, it is natural to use a loose DAE solver tolerance ε while the iterates are far from a stationary point, and progressively tighten the DAE solver tolerance ε as the sequence of iterates converges to a stationary point. We developed such a scheme which, given an initial DAE solver tolerance $\varepsilon_0 \in \mathbb{R}^q_+$, approximately solves the approximating optimization problem $\mathbf{P}_{\varepsilon_0}$ until a test in the optimization algorithm fails. If this test fails, a higher precision approximating optimization problem $\mathbf{P}_{\varepsilon_1}$ is constructed by replacing ε_0 with ε_1 according to a prescribed rule. The new problem $\mathbf{P}_{\varepsilon_1}$ is initialized with the iterate that yield lowest cost for problem $\mathbf{P}_{\varepsilon_0}$. In implementing our adaptive simulation precision optimization scheme, the construction of approximating optimization problems $\{\mathbf{P}_{\varepsilon_N}\}_{N\in\mathbb{N}}$ is done iteratively until a final precision ε_M is achieved. This

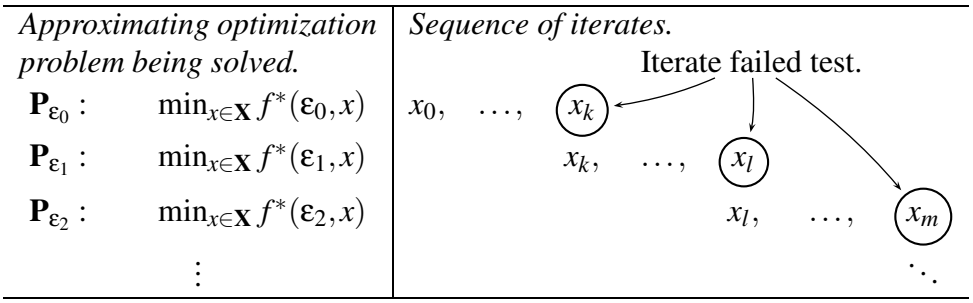


Figure 1.2: Diagonalization scheme (schematic). The circled iterates failed to satisfy a test in the optimization algorithm. This caused the construction of the next approximating optimization problem which is of higher precision.

gives the diagonal sequence of iterates which is schematically shown in Fig. 1.2, and which gives the scheme the name *diagonalization scheme*. The values ε_0 and ε_M are problem dependent. They can be determined by doing, prior to the optimization, a few cost function evaluations in order to check how coarse an ε_0 gives accurate enough approximations to $f(\cdot)$ when far from a stationary point, and how tight an ε_M need to be selected to compute smooth enough approximations to $f(\cdot)$.

1.5.1.3 Optimization Algorithms

Because the approximating cost functions $\{f^*(\varepsilon,\cdot)\}_{\varepsilon\in\mathbb{R}^q_+}$ are discontinuous, we selected the derivative-free Generalized Pattern Search (GPS) optimization algorithms to search for a decrease in $f^*(\varepsilon,\cdot)$. Our main research result for the diagonalization scheme with GPS algorithms is that we developed tests when to construct $\mathbf{P}_{\varepsilon_{N+1}}$ and a rule how to construct $\mathbf{P}_{\varepsilon_{N+1}}$ for which we proved that any GPS algorithm constructs sequences of iterates with stationary accumulation points, i.e., $x_k \to x^*$ where $\nabla f(x^*) = 0$. The fact that the approximations to the cost function are discontinuous does not affect the

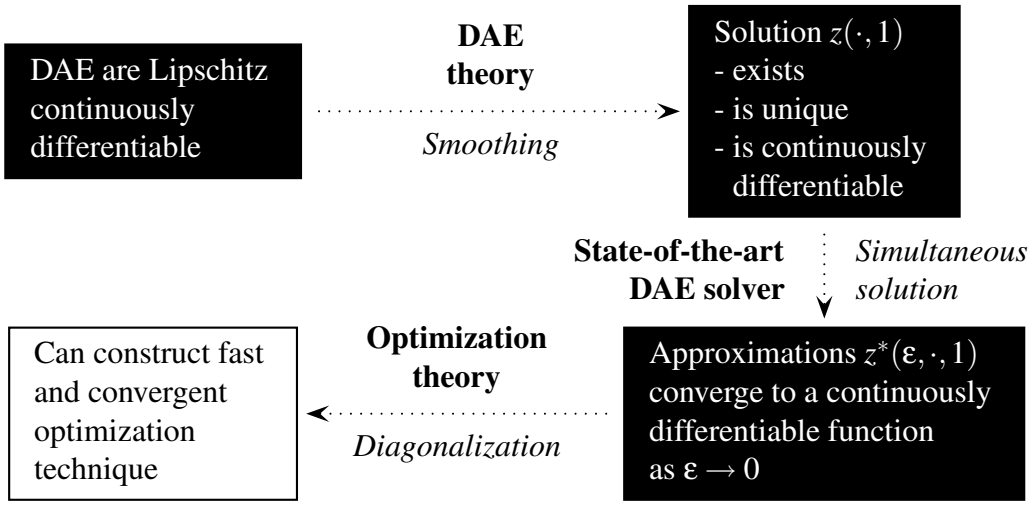


Figure 1.3: Approach for developing a fast convergent optimization technique.

convergence of our optimization technique.

In our numerical experiments, the use of adaptive precision cost function evaluations reduced the computation time for a building design optimization from five days to about one day.

1.5.1.4 Summary of Approach

Fig. 1.3 summarizes our approach for constructing a fast optimization technique to solve optimization problems in which the cost function is defined on the solution to a DAE system. Firstly, because we constructed the models that define the DAE system (1.1.3) using functions that are Lipschitz continuously differentiable in the state variables, in time and in the building design parameters, we were able to use standard DAE theory to prove that our DAE system has a solution $z(\cdot,1)$ that is unique and once

continuously differentiable. To have a computer code available that generates a building energy simulation model that represent such a DAE system, we developed $30,000$ lines of C/C++ code that contains smoothing methods which we used to convert non-differentiable models into smooth models. Because our simulation models are defined by smooth equations, we were able to solve all equations of the DAE system simultaneously using a state-of-the-art DAE solver. Finally, once we established convergence of $z^*(\varepsilon,\cdot,1)$ to a smooth function $z(\cdot,1)$, we could construct a diagonalization scheme that progressively increases the precision of the approximating optimization problems $\mathbf{P}_\varepsilon$. The diagonalization scheme significantly reduced the computation time for the optimization, and it allowed the use optimization theory to prove that the sequence of iterates contains stationary accumulation points.

1.5.2 Optimization with Fixed Precision Cost Function Evaluations

In many situations, one needs to use existing simulation programs that do not allow controlling the approximation error $|f^*(\varepsilon,x)-f(x)|$ as they have not been designed for use with optimization algorithms that require smoothness of the cost function. Approximating cost functions computed by such existing simulation programs may have large discontinuities, and it may not be practical to rewrite such code. We address this situation in Chapter 4, in which we compare the performance of probabilistic and deterministic optimization algorithms in minimizing cost functions that were computed by EnergyPlus. This comparison is meant as a guideline that shows which existing optimization

algorithms tend to work well on such problems.

Chapter 2

BuildOpt – A New Building Energy Simulation Program that is Built on Smooth Models

2.1 Introduction

In this chapter we present BuildOpt, a new multi-zone thermal and daylighting building energy simulation program. BuildOpt is different from existing building energy simulation programs, such as EnergyPlus (Crawley et al., 2001), TRNSYS (Klein et al., 1976), ESP-r (Clarke, 2001), and DOE-2 (Winkelmann et al., 1993), since it is built on models that are defined by differential algebraic equations (DAE system) that are once Lipschitz continuously differentiable in the building design parameters, in the state variables and in time, and since all partial differential equations, ordinary differential equations and algebraic equations are solved simultaneously. The use of smooth models not only allows proving that the DAE system has a unique solution that is once continuously differentiable in the building design parameters, but it is in fact required to achieve convergence of the DAE solver if the solver tolerances are tight. This is a significant observation because today's building energy simulation programs are built on non-smooth models, and their solvers frequently fail to obtain a numerical solution if the solver tolerances are tight.

The use of a DAE solver, as opposed to ad-hoc implemented solvers that are spread throughout the code (which is common in most building energy simulation programs), allows controlling the precision of the numerical approximations to the solution of the DAE system and hence it allows obtaining a function that bounds the approximation error as a function of the solver tolerance. This is required in order for the simula-

tion program to be used with Generalized Pattern Search (GPS) optimization algorithms with adaptive precision cost function evaluations that we present in Chapter 3, or by algorithms that are based on the Master Algorithm Model 1.2.36 in Polak (1997). Those optimization algorithms use coarse precision simulations for the early iterations and progressively increase the precision of the simulations. This significantly reduces computation time and allows proving that the optimization algorithm constructs sequences of iterates with stationary accumulation points. To the best of our knowledge, BuildOpt is the first building energy simulation program that can be used to do building design optimizations that provably converge to a stationary point.

Numerical experiments with EnergyPlus and analysis of its source code revealed that it does not seem possible to prove that EnergyPlus computes an approximate solution that converges to a function that is once continuously differentiable in the building design parameters as the solver tolerances are tightened. In fact, in numerical experiments in which we modeled a building's heating and cooling load and daylighting control, there were about ten solvers that controlled subsystems of the simulation model (such as the heat conduction in the solids, the variable time-step integration of the room air temperature and the initialization of the state variables). We were not able to analyze how the approximation errors of the different solvers were propagated from one model to another, and from one time step to the next, and the code became unstable as we increased the solver tolerances. In Section 4.5.2 on page 110, as well as in Wetter and Wright (2003a) and Wetter and Polak (2003), it is shown that a building's annual energy con-

sumption computed by EnergyPlus is discontinuous in the building design parameters, and that the discontinuities are in some cases on the order of 2% of the cost function value. This caused in some numerical experiments the optimization algorithms to fail far from a minimum (see for example Fig. 4.2 on page 109). In order to have a building energy simulation program that can be used to perform building design optimizations that provably find a stationary point of the cost function, we had to develop BuildOpt.

One may ask why we developed our own simulation program rather than having used the IDA-ICE program (Björsell et al., 1999; Sahlin and Bring, 1991), which is an equation-based building energy analysis program that has a large library of simulation models. IDA-ICE generates from equation-based models a DAE system which it solves simultaneously. Discussions with its developer Per Sahlin showed that IDA-ICE might indeed be a promising tool for use with our optimization algorithms. However, without extensive numerical experiments and code analysis, it is not possible to conclude that IDA-ICE satisfies our requirements. Furthermore, in case of bad performance of our optimization algorithms, it would have been hard if not impossible to detect why our algorithms did not work as expected. Therefore, for the initial experiments of our optimization algorithms, we preferred to develop our own code.

This chapter is structured as follows. First, we present the optimization problem for which we developed BuildOpt to compute numerical approximations to the cost function. Then, we present the assumptions that the simulation program needs to satisfy in order to be used with our GPS algorithms with adaptive precision cost function evalu-

ations. Next, we characterize BuildOpt's physical models, explain some of the models that are implemented in BuildOpt, and explain some of the smoothing techniques that are used in implementing the models. Then, we present numerical experiments that compare how the smoothing techniques affect the convergence of the DAE solver and that verify that the numerical approximations to the state variables converge to a function that is once continuously differentiable in the building design parameters as the precision of the DAE solver is increased.

A detailed discussion of all models and of the smoothing techniques can be found in Appendix A. A validation of BuildOpt can be found in Appendix B.

2.2 Properties of Optimization Problem

2.2.1 Statement of the Optimization Problem

We will now state the optimization problem in which we will use BuildOpt to compute numerical approximations to the cost function. The optimization problem is of the form

$$\min_{x \in \mathbf{X}} f(x), \tag{2.2.1}$$

where $\mathbf{X} \triangleq \left\{ x \in \mathbb{R}^n \mid l^i \leq x^i \leq u^i,\ i \in \{1,\ldots,n\} \right\}$ is the constraint set, with $-\infty \leq l^i < u^i \leq \infty$ for all $i \in \{1,\ldots,n\}$.

We assume that the cost function is once continuously differentiable and defined as

$$f(x) \triangleq F(z(x,1)), \tag{2.2.2}$$

where $F\colon \mathbb{R}^m \to \mathbb{R}$ is once continuously differentiable and $z(x,1) \in \mathbb{R}^m$ is the solution of a semi-explicit nonlinear DAE system with index one (Brenan et al., 1989) of the form

$$\begin{aligned} \dot{z}(x,t) &= h\big(x, z(x,t), \mu\big), \qquad t \in [0,\ 1], && \text{(2.2.3a)} \\ z(x,0) &= z_0(x), && \text{(2.2.3b)} \\ \gamma\big(x, z(x,t), \mu\big) &= 0, && \text{(2.2.3c)} \end{aligned}$$

where $h\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^m$, $z_0\colon \mathbb{R}^n \to \mathbb{R}^m$ and $\gamma\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^l$ are once Lipschitz continuously differentiable in all arguments and equation (2.2.3c) has, for all $x \in \mathbb{R}^n$ and for all $z(\cdot,\cdot) \in \mathbb{R}^m$, a unique solution $\mu^*(x,z) \in \mathbb{R}^l$ and the matrix with partial derivatives $\partial\gamma(x, z(x,t), \mu^*(x,z))/\partial\mu \in \mathbb{R}^{l\times l}$ is non-singular. The notation $\dot{z}(x,t)$ denotes differentiation with respect to time.

Equation (2.2.3) is a DAE system that describes a building energy simulation model after the spatial domains of wall, floor and ceiling constructions have been discretized in a finite number of nodal points. For example, the components of the vector $z(\cdot,\cdot)$ can be the room air temperature, the solid temperature at the nodal points, and the building energy consumption for cooling, heating and lighting, and $\gamma(\cdot,\cdot,\cdot)$ can be a system of

nonlinear equations that is used to describe the temperature of elements with negligible thermal capacity, such as window glass.

2.2.2 Existence of a Unique Smooth Solution of the DAE System

We will now state the assumptions that we use to establish existence, uniqueness and differentiability of the solution $z(\cdot,1)$ of (2.2.3).

Assumption 2.2.1 *With* γ: $\mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^l$ *as in* (2.2.3c)*, we assume that* $\gamma(\cdot,\cdot,\cdot)$ *is once continuously differentiable, and we assume that for all* $x \in \mathbb{R}^n$ *and for all* $z(\cdot,\cdot) \in \mathbb{R}^m$*, equation* (2.2.3c) *has a unique solution* $\mu^*(x,z) \in \mathbb{R}^l$ *and that the matrix with partial derivatives* $\partial\gamma(x,z(x,t),\mu^*(x,z))/\partial\mu \in \mathbb{R}^{l\times l}$ *is non-singular.* □

By using the Implicit Function Theorem (Polak, 1997), one can show that Assumption 2.2.1 implies that the solution of (2.2.3c), i.e., the $\mu^*(x,z)$ that satisfies the algebraic equation $\gamma\big(x,z(x,t),\mu^*(x,z)\big) = 0$, is unique and once continuously differentiable in x and z. Therefore, to establish existence, uniqueness and differentiability of $z(\cdot,1)$, we can reduce the DAE system (2.2.3) to an ordinary differential equation, which will allow us to use standard results from the theory of ordinary differential equations. To do so, we define for $x \in \mathbb{R}^n$, for $t \in [0,1]$ and for $z(x,t) \in \mathbb{R}^m$ the function

$$\widetilde{h}(x,z(x,t)) \triangleq h\big(x,z(x,t),\mu^*(x,z)\big), \tag{2.2.4}$$

and write the DAE system (2.2.3) in the form

$$\dot{z}(x,t) = \widetilde{h}\big(x, z(x,t)\big), \qquad t \in [0,\ 1], \tag{2.2.5a}$$

$$z(x,0) = z_0(x). \tag{2.2.5b}$$

We will use the notation $\widetilde{h}_x(x,z(x,t))$ and $\widetilde{h}_z(x,z(x,t))$ to denote the partial derivatives $(\partial/\partial x)(\widetilde{h}(x,z(x,t)))$ and $(\partial/\partial z)(\widetilde{h}(x,z(x,t)))$, respectively. We will make the following assumption.

Assumption 2.2.2 *With $\widetilde{h}\colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$ and $z_0\colon \mathbb{R}^n \to \mathbb{R}^m$ as in* (2.2.5)*, we assume that*

1. *the initial condition $z_0(\cdot)$ is continuously differentiable,*

2. *there exists a constant $K \in [1,\infty)$ such that for all $x', x'' \in \mathbb{R}^n$ and for all $z', z'' \in \mathbb{R}^m$, the following relations hold:*

$$\|\widetilde{h}(x',z') - \widetilde{h}(x'',z'')\| \leq K\left(\|x'-x''\| + \|z'-z''\|\right), \tag{2.2.6a}$$

$$\|\widetilde{h}_x(x',z') - \widetilde{h}_x(x'',z'')\| \leq K\left(\|x'-x''\| + \|z'-z''\|\right), \tag{2.2.6b}$$

and

$$\|\widetilde{h}_z(x',z') - \widetilde{h}_z(x'',z'')\| \leq K\left(\|x'-x''\| + \|z'-z''\|\right). \tag{2.2.6c}$$

□

Now we can use the following theorem, which is a special case of Corollary 5.6.9 in Polak (1997), to show that $f(\cdot) \triangleq F(z(\cdot,1))$ is once continuously differentiable.

Theorem 2.2.3 *Suppose that $F\colon \mathbb{R}^m \to \mathbb{R}$ is once continuously differentiable on bounded sets, that Assumptions 2.2.1 and 2.2.2 are satisfied and that $f\colon \mathbb{R}^n \to \mathbb{R}$ is defined by $f(x) \triangleq F(z(x,1))$. Then, $f(\cdot)$ is once continuously differentiable on bounded sets.* □

2.2.3 Numerical Solutions of the DAE System

We assume that $z(x,t)$ cannot be evaluated exactly, but that it can be approximated numerically by functions $z^*(\varepsilon,x,t)$, where $z^*\colon \mathbb{R}^q_+ \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^m$ and $\varepsilon \in \mathbb{R}^q_+$ is a vector that contains the precision parameters of the DAE solvers. Hence, we denote by $z^*(\varepsilon,x,t)$ the numerical approximation for the solution $z(x,t)$ of (2.2.3), as computed by a simulation program with solver precision parameters ε. Thus, for $\varepsilon \in \mathbb{R}^q_+$ and $x \in \mathbb{R}^n$, we define approximating cost functions $f^*(\varepsilon,x) \triangleq F(z^*(\varepsilon,x,1))$ which are, in general, discontinuous in x due to adaptive algorithms in the DAE solver, such as variable time step integration algorithms or Newton-based solvers.

2.2.4 Mathematical Requirements on the Solutions of the DAE System

BuildOpt has been developed to evaluate the cost function in optimization algorithms that adaptively control the precision of the cost function evaluations during the course of

the optimization. Example of such algorithms are the Generalized Pattern Search (GPS) algorithms with adaptive precision cost function evaluations (Polak and Wetter, 2003), which are discussed in Chapter 3. For those algorithms, we need to make the following assumptions on the solution $z(\cdot,1)$ and its numerical approximations $\{z^*(\varepsilon,\cdot,1)\}_{\varepsilon\in\mathbb{R}^q_+}$.

Assumption 2.2.4

1. *There exists an error bound function* $\varphi\colon \mathbb{R}^q_+ \to \mathbb{R}_+$ *such that for any bounded set* $\mathbf{S} \subset \mathbf{X}$*, there exists an* $\varepsilon_{\mathbf{S}} \in \mathbb{R}^q_+$ *and a scalar* $K_{\mathbf{S}} \in (0,\infty)$ *such that for all* $x \in \mathbf{S}$ *and for all* $\varepsilon \in \mathbb{R}^q_+$*, with* $\varepsilon \le \varepsilon_{\mathbf{S}}$*,*

$$|z^*(\varepsilon,x,1) - z(x,1)| \le K_{\mathbf{S}}\,\varphi(\varepsilon). \tag{2.2.7}$$

Furthermore,

$$\lim_{\|\varepsilon\|\to 0} \varphi(\varepsilon) = 0. \tag{2.2.8}$$

2. *The function* $z\colon \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ *is once continuously differentiable.* □

Note that we allow the functions $\{z^*(\varepsilon,\cdot,1)\}_{\varepsilon\in\mathbb{R}^q_+}$ to be discontinuous. Examples of error bound functions $\varphi(\cdot)$ can be found in Section 3.5 on page 66 and in Polak and Wetter (2003).

2.3 BuildOpt Simulation Program

Many if not all of today's detailed building simulation programs are built on models that do not satisfy Assumptions 2.2.1 and 2.2.2. Furthermore, the numerical experiments presented in Section 4.5.2 on page 110, as well as in Wetter and Wright (2003a) and Wetter and Polak (2003), show that approximating cost functions computed by EnergyPlus can have discontinuities on the order of 2% of the cost function value. In some numerical experiments, this caused optimization algorithms to jam.

To ensure that the simulation program used in the optimizations is built on models that are once Lipschitz continuously differentiable in the input data and to ensure that controlling the approximation error is possible, we developed BuildOpt, a new building energy simulation program.

BuildOpt consists of two parts. The first part, which we will call the *simulation model generator*, parses a text input file with the detailed description of the building geometry, the building materials and the expected occupancy behavior and then generates a simulation model for the particular building.

The second part of BuildOpt, to which our simulation model generator was linked, is the commercial DAE solver DASPK (Brenan et al., 1989; Brown et al., 1994, 1998).

The total size of BuildOpt is $38,000$ of C/C++ and Fortran code, of which $30,000$ lines (1.2 MB) of C/C++ code represent the simulation model generator that we wrote and $8,000$ lines (0.3 MB) of Fortran 77 code represent the commercial solver DASPK.

2.3.1 Simulation Model Generator

The simulation model generator constructs a detailed simulation model for the building that is defined in the simulation input file. To give an impression of the model complexity, we will here present a brief overview of some of the component models that are used to construct the building simulation model. A detailed description of the component models can be found in Appendix A.

The diffuse solar irradiation is computed using the model of Perez et al. (1990, 1987) and the radiation temperature of a cloudy sky is computed using the model developed by Martin and Berdahl (1984). To compute the heat conduction in opaque materials, with possibly composite layers, the Galerkin method (Evans, 1998; Strang and Fix, 1973) is used for the spatial discretization. The spatial discretization results in systems of ordinary differential equations. The systems are coupled to other constructions via long-wave radiative heat exchange, and are coupled to the room air temperatures via convective heat transfer. The short-wave radiation through multi-pane windows is computed using a model similar to the one used in the WINDOW 4 program (Arasteh et al., 1989; Finlayson et al., 1993). The daylight illuminance is computed with a model based on view-factors that is similar to the model in the DeLight program of Vartiainen (2000). All equations are solved simultaneously, as explained in Section 2.3.3 on page 36.

2.3.2 Smoothing Techniques

BuildOpt differs from other building simulation programs in that it uses various smoothing techniques to make all model equations as well as the table look-ups (used in Perez' model), hourly schedules of internal heat gains and weather data once Lipschitz continuously differentiable in the state variables, in the model parameters and in time. Smoothing is required to satisfy Assumption 2.2.2, it significantly reduces the computation time and it is required to achieve convergence of the DAE solver if the solver tolerance is tight.

The building blocks used to formulate once Lipschitz continuously differentiable models are as follows. For $s \in \mathbb{R}$ and for some $\delta > 0$, we defined a once Lipschitz continuously differentiable approximation for the Heaviside function as

$$\widetilde{\mathrm{H}}(s;\delta) \triangleq \begin{cases} 0, & \text{for } s < -\delta, \\ \frac{1}{2}\left(\sin\left(\frac{s}{\delta}\frac{\pi}{2}\right)+1\right), & \text{for } -\delta \leq s < \delta, \\ 1, & \text{for } \delta \leq s. \end{cases} \tag{2.3.1}$$

We parametrized (2.3.1) by $\delta > 0$ to be able to take the scaling of s into account. Equation (2.3.1) is used to define a once Lipschitz continuously differentiable approximation for the max function as

$$\widetilde{\max}(a,b;\delta) \triangleq a + (b-a)\,\widetilde{\mathrm{H}}(b-a;\delta). \tag{2.3.2}$$

This function is then, for example, used to smooth the convective heat transfer coefficient between a wall surface and the room air as we will now explain. A commonly used equation for the convective heat transfer coefficient due to natural convection between a wall surface at temperature T and the room air, which we assume for this explanation to have zero temperature, is $h(T) = 1.310|T|^{1/3}$. Hence, the convective heat transfer per unit area is $q(T) = 1.310|T|^{1/3}T$, which has a derivative that fails to be Lipschitz continuous near zero. Consequently, we used for the convective heat transfer coefficient the once Lipschitz continuously differentiable approximation $\widetilde{h}(T) = 1.310\ \widetilde{\max}(1,\ |T|^{1/3}; 0.1)$.

To interpolate the weather data for time instants that do not coincide with time stamps in the weather data file, we used cubic splines. To interpolate values of internal loads for people, lighting and electric equipment, which are specified by hourly schedules, we used the smooth Heaviside function (2.3.1) with $\delta = 1/2$ hour.

Thus, BuildOpt's simulation models are written in such a way that the functions $h\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^m$, $z_0\colon \mathbb{R}^n \to \mathbb{R}^m$ and $\gamma\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^l$, defined in (2.2.3), are implemented so that the Assumptions 2.2.1 and 2.2.2 are satisfied. We do not believe that there is any other building energy simulation program that is built on models that are as detailed as the models in BuildOpt and that satisfies Assumptions 2.2.1 and 2.2.2.

2.3.3 Solving the Equations

BuildOpt's models are linked with the DAE solver DASPK (Brenan et al., 1989; Brown et al., 1994). The DASPK solver uses a variable time-step, variable order Backward-Differentiation Formula.

To solve the DAE system (2.2.3), the DASPK solver requires the simulation model to be written in the residual form

$$G(t, v(x,t), \dot{v}(x,t)) = \begin{pmatrix} \dot{z}(x,t) - h(x, z(x,t), \mu^*(x,z)) \\ \\ \gamma(x, z(x,t), \mu^*(x,z)) \end{pmatrix} = 0, \qquad (2.3.3)$$

where $v(x,t) \triangleq (z(x,t), \mu^*(x,z))^T \in \mathbb{R}^{m+l}$ is the vector of differential variables $z(x,t)$ and of algebraic variables $\mu^*(x,z)$, which are the solution of (2.2.3c). Given initial values of the differential variables $z(x,0)$, DASPK computes consistent initial conditions $\dot{z}(x,0)$ and $\mu^*(x,z(x,0))$, or conversely, given $\dot{v}(x,0)$, it computes consistent values for $v(x,0)$ (see Brown et al. (1998)).[1] At each time step $t \in [0,1]$, DASPK passes to BuildOpt a $\widehat{t} > t$, a $\widehat{v}(x,\widehat{t})$ and a $\widehat{\dot{v}}(x,\widehat{t})$, where $\widehat{\dot{v}}(x,\widehat{t})$ is approximated by backward differences[2] and BuildOpt returns to DASPK the residual vector $G\left(\widehat{t}, \widehat{v}(x,\widehat{t}), \widehat{\dot{v}}(x,\widehat{t})\right) \in \mathbb{R}^{m+l}$. This process is repeated iteratively until all convergence tests in DASPK are satisfied. Our simulation model is too big to obtain an analytical expression for the iteration matrices $G_v(\cdot,\cdot,\cdot)$ and $G_{\dot{v}}(\cdot,\cdot,\cdot)$ used by DASPK. Hence, we configured DASPK so that it ap-

[1] We say that initial conditions $v(x,0)$ and $\dot{v}(x,0)$ are consistent if $G(0, v(x,0), \dot{v}(x,0)) = 0$.

[2] E.g., if the Implicit Euler method is used, then $\widehat{\dot{v}}(x,\widehat{t})$ is replaced by $(v(x,\widehat{t}) - v(x,\widehat{t} - \delta))/\delta$, where $\delta \in \mathbb{R}$ is the integration time step.

proximates the iteration matrices using finite differences. The linear system of equations that arises in the Newton iterations is solved using a direct method. We note that more efficient solution strategies could be implemented in our code, but we have not yet pursued such improvements. For example, the linear system could be solved using a sparse matrix solver or Krylov iterations. Furthermore, we currently check only in the computationally most expensive models if the input data are the same as in the last model evaluation, in which case no model evaluation is required.[3]

2.3.4 Model Validation

The thermal simulation model was validated using the ANSI/ASHRAE Standard test procedure 140-2001 (ASHRAE, 2001), and the daylighting simulation was validated using benchmark tests from Laforgue (1997) and Fontoynont et al. (1999), which were produced in the Task 21 of the International Energy Agency (IEA) Solar Heating & Cooling Program. The validations can be found in Appendix B. The results of BuildOpt show good agreement with the results of the other validated programs.

2.4 Numerical Experiments

We will now describe how the smoothing techniques affect the convergence of the DASPK solver and consequently reduce the computation time. For the numerical ex-

[3]When DASPK approximates the elements of the iteration matrices $G_v(\cdot,\cdot,\cdot)$ and $G_{\dot{v}}(\cdot,\cdot,\cdot)$, many models are repetitively evaluated with no change in input data.

periments, we did computations of the annual source energy consumption for heating, cooling and lighting of an office building in Houston, TX. We simulated three representative spaces: a north and a south facing room and a hallway between the rooms. We used the same simulation model as the one described in Section 3.5.1 on page 67. In Tab. 2.1 we show for different precision parameters ε the normalized number of evaluations of the residual function $G(\cdot,\cdot,\cdot)$, defined in (2.3.3), in an annual simulation. For BuildOpt, the number of residual evaluations is proportional to the computation time. A normalized computation time of 1.0 corresponds to 33.4 minutes on one 2.2 GHz AMD processor using the Linux $2.4.18-3$ kernel. The first three columns in Tab. 2.1 are defined as follows: In the column labeled "model equations", "smooth" means that the smoothing of the model equations is enabled (i.e., all model equations, but not necessarily the hourly schedules, are once Lipschitz continuously differentiable in the state and in time), and "non-smooth" means that the smoothing is disabled. In the column labeled "internal loads", "smooth" means that internal loads due to people, lights, and electric equipment, which are all specified by hourly schedules, are interpolated using the function $\widetilde{H}(\cdot;\delta)$, as defined in (2.3.1), with $\delta = 1/2$ hour, "linear" means that we used linear interpolation, where the change from one value to the next occurs over one hour, and "step" means that the hourly schedules are implemented as step functions. In the column labeled "weather data", "cubic" means that we used cubic splines to interpolate the weather data, and "linear" means that we used linear interpolation. For all combinations of these smoothing techniques, we did five annual simulations with solver

tolerance settings $\varepsilon \in \{10^{-m}\}_{m=1}^{5}$.

We observed that we were only able to compute high precision approximations when we used once Lipschitz continuously differentiable models, which will hardly surprise any numerical analyst. The reason is that the DASPK solver uses Taylor expansions to approximate solutions of nonlinear equations and to replace derivatives by finite difference approximations, which is common to any Newton-based solver. However, if the equations being solved are not differentiable, then the Taylor expansions are inaccurate or even completely wrong, which can cause the Newton search direction to point away from the solution of the equation. However, in practice we observe that building simulation programs are built on non-smooth models and contain Newton-based solvers that frequently fail to find a solution if the solver tolerances are tight. This is what we also observed in BuildOpt when we disabled the smoothing techniques. Furthermore, when the solver tolerance was tight, BuildOpt's computation time increased by a factor of two when we changed from cubic splines to linear interpolation of the weather data. This is interesting because many if not all of today's building simulation programs use linear interpolation rather than cubic splines. Thus, we believe that the convergence properties of today's building energy simulation programs could be significantly improved if they were built on once Lipschitz continuously differentiable models and if they used weather data interpolations that are once Lipschitz continuously differentiable in time. Numerical solvers that detect state events, such as a change in model equations for some domain of the model input data, are likely to be more robust than DASPK if non-smooth models

Smoothing			Solver tolerance ε				
Model equations	Internal loads	Weather data	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
smooth	smooth	cubic	0.011	0.080	0.169	0.497	1
smooth	linear	cubic	0.011	0.078	0.170	0.503	1.048
smooth	step	cubic	0.012	0.091	*	*	*
smooth	smooth	linear	0.005	0.056	0.299	0.879	2.012
smooth	linear	linear	0.006	0.054	0.300	0.889	2.044
smooth	step	linear	0.007	0.069	0.454	*	*
non-smooth	smooth	cubic	0.011	0.078	0.233	*	*
non-smooth	linear	cubic	0.011	0.078	0.238	*	*
non-smooth	step	cubic	0.012	0.093	*	*	*
non-smooth	smooth	linear	0.009	0.093	0.511	*	*
non-smooth	linear	linear	0.009	0.093	0.521	*	*
non-smooth	step	linear	0.010	0.110	*	*	*

Table 2.1: Normalized number of calls to $G(\cdot,\cdot,\cdot)$ in an annual simulation with different solver tolerances and different smoothing. An asterisk "*" means that the DAE solver failed to converge in 25 time steps, in which case the simulation stopped.

are used, but the detection of state events is computationally expensive and hence it may be better to prevent state events where possible.

We will now show how the approximate numerical solutions converge to a once continuously differentiable function as the tolerance of the DASPK solver is tightened.

Let $z^*(\varepsilon, x, 1) \in \mathbb{R}$ denote the numerical approximation of the annual source energy consumption for heating, cooling and lighting of the office building used in the above numerical experiments, computed by BuildOpt with solver tolerance $\varepsilon \in \mathbb{R}_+$. Here, $x \in \mathbb{R}$ denotes the normalized setpoint for the shading device of the south facing window. Let

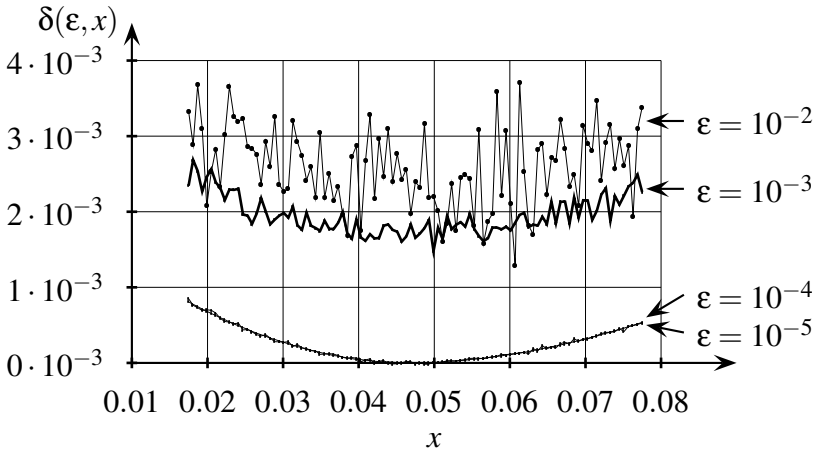


Figure 2.1: Relative change of the annual source energy consumption $\delta(\varepsilon,x)$, defined in (2.4.1), for different precision parameters ε. For better visibility of the data series, the support points are connected by lines.

$\delta(\varepsilon,x)$ denote the relative change in the source energy consumption, defined as

$$\delta(\varepsilon,x) \triangleq \frac{z^*(\varepsilon,x,1) - z^*(10^{-5},0.048,1)}{z^*(10^{-5},0.048,1)}. \tag{2.4.1}$$

In (2.4.1), the argument 0.048 corresponds to the normalized shading control setpoint that yields lowest annual source energy consumption. In Fig. 2.1, we plot $\delta(\varepsilon,x)$ for different values of ε and x. The figure shows how $z^*(\varepsilon,\cdot,1)$ converges to a once continuously differentiable function as $\varepsilon \to 0$. The difference between $\delta(10^{-4},\cdot)$ and $\delta(10^{-5},\cdot)$ is almost invisible.

2.5 Conclusion

Building energy simulation programs can be written so that they compute approximate solutions to a DAE system that converge to a function that is once continuously differentiable in the building design parameters as the solver tolerance is tightened. This is required if the simulation program is used with GPS algorithms with adaptive precision cost function evaluations which provably construct sequences of iterates with stationary accumulation points.

To obtain convergence of the DAE solver at tight solver tolerance, and to reduce the computation time, it is essential that the model equations, the hourly schedules and the weather data are once Lipschitz continuously differentiable in the state variables and in time. This observation is significant because today's building energy simulation programs are built on non-smooth models and their solvers are known to frequently fail to converge to a solution if the solver tolerances are tight.

Chapter 3

Optimization with Adaptive Precision Cost Function Evalutions

3.1 Introduction

Generalized pattern search (GPS) algorithms are derivative free methods for the minimization of smooth functions, possibly with linear inequality constraints. Examples of pattern search algorithms are the coordinate search algorithm (Polak, 1971), the pattern search algorithm of Hooke and Jeeves (1961), and the multidirectional search algorithm of Dennis and Torczon (1991). What they all have in common is that they define the construction of a mesh, which is then explored according to some rule, and if no decrease in cost is obtained on mesh points around the current iterate, then the mesh is refined and the process is repeated.

In 1997, Torczon was the first to show that all the existing pattern search algorithms are specific implementations of an abstract pattern search scheme and to establish that for unconstrained problems with smooth cost functions, the gradient of the cost function vanishes at accumulation points of sequences constructed by this scheme. Lewis and Torczon extended her theory to address bound constrained problems (Lewis and Torczon, 1999) and problems with linear inequality constraints (Lewis and Torczon, 2000). In both cases, convergence to a feasible point x^* satisfying $\langle \nabla f(x^*), x - x^* \rangle \geq 0$ for all feasible x is proven under the condition that $f(\cdot)$ is once continuously differentiable. Audet and Dennis (2003) present a simpler abstraction of GPS algorithms, and, in addition to reestablishing the Torczon and the Lewis and Torczon results, they relax the assumption that the cost function is smooth to that it is locally Lipschitz continuous.

However, their characterization of accumulation points of sequences constructed by a GPS algorithm, on a locally Lipschitz continuous cost function, while not without merit, falls short of showing that the accumulation points are stationary in the Clarke sense (Clarke, 1990), i.e., $0 \in \partial^0 f(x^*)$. It does not seem possible to improve their result.

In principle, a natural area for the application of GPS algorithms is engineering optimization, where the cost functions are defined on the solution of complex systems of equations including implicit equations, ordinary differential equations, and partial differential equations. However, in such cases, obtaining an accurate approximation to the cost function often takes many hours, and there is no straightforward way of approximating gradients. Furthermore, it is not uncommon that the termination criteria of the numerical solvers introduce discontinuities in the approximations to the cost function. Hence, standard GPS algorithms can only be used heuristically in this context.

In this chapter we present a modified class of GPS algorithms which adjust the precision of the cost function evaluations adaptively: low precision in the early iterations, with precision progressively increasing as a solution is approached. The modified GPS algorithms converge to stationary points of the cost function even though the cost function is approximated by a family of discontinuous functions.

We assume that the cost function $f(\cdot)$ is at least locally Lipschitz continuous and that it can be approximated by a family of functions, say $\{f^*(\varepsilon,\cdot)\}_{\varepsilon\in\mathbb{R}^q_+}$, with fixed $q \in \mathbb{N}$, where $\varepsilon \in \mathbb{R}^q_+$ denotes the tolerance settings of the PDE, ODE and algebraic equation solvers, and each $f^*(\varepsilon,\cdot)$ may be discontinuous but converges to $f(\cdot)$ uniformly

on compact sets. A test in the algorithm determines when precision must be increased. This test includes parameters that can be used to control the speed with which precision is increased. We will show by numerical experiments that this flexibility can be exploited to obtain a significant reduction in computation times, as compared to using high precision throughout the computation.

Under the assumption that $f(\cdot)$ is once continuously differentiable, our GPS algorithms converge to stationary points of $f(\cdot)$, while under the assumption that $f(\cdot)$ is only locally Lipschitz continuous, our algorithms converge to points at which the Clarke generalized directional derivatives of $f(\cdot)$ are nonnegative in predefined directions. Thus, we regain the results of Audet and Dennis (2003).

Contrary to the model management framework with GPS algorithms (Booker et al., 1999; Dennis and Torczon, 1997; Serafini, 1998; Torczon and Trosset, 1998), we do not assume that function values of $f(\cdot)$ are available, and consequently, we do not construct surrogate models of increasing accuracy that are based on support points at which $f(\cdot)$ has been evaluated. Our algorithms construct an infinite sequence of approximating cost functions $\{f^*(\varepsilon,\cdot)\}_{\varepsilon\in\mathbb{R}^q_+}$ so that $f^*(\varepsilon,\cdot)$ converges to $f(\cdot)$ fast enough near stationary points. However, since our GPS algorithms include global search and local search stages, as is typical in GPS algorithms, our GPS algorithms allow the use of surrogate models of $f^*(\varepsilon,\cdot)$ to obtain points for the global search.

In implicit filtering, Kelley (1999b) accounts for the situation where $f(\cdot)$ is approximated numerically using a computer program that contains adaptive solvers. In implicit

filtering, however, one does not adaptively control the error of the cost function evaluations, but rather assumes in proving convergence to a stationary point of $f(\cdot)$ that the error of the approximating cost function decays faster to zero than the step size used in the finite difference approximation of the gradient of the cost function. Because in our convergence analysis we establish a lower bound for Clarke's generalized directional derivative, which we bound by a sequence of finite difference approximations, we need to assume the same rate of error decay as is assumed in implicit filtering. However, in contrast to implicit filtering, our algorithms adaptively control the approximation error. This allows us to construct a simulation precision control algorithm that causes the optimization to use computationally cheap coarse precision approximations to the cost function in the early iterations, and progressively use higher precision cost function evaluations as needed when the algorithm approaches a stationary point.

3.2 Optimization Problem

To best explain our precision control algorithm without having to discuss technical details of constructing search directions that conform, in the sense of Audet and Dennis (2003), to the feasible set of design parameters, we restrict our discussion to box-constrained problems rather than to problems with general linear constraints. The construction of search directions for linearly constrained problems is discussed in Kolda et al. (2003), Audet and Dennis (2003) and Polak and Wetter (2003).

We will consider box-constrained problems

$$\min_{x \in \mathbf{X}} f(x), \tag{3.2.1a}$$

$$\mathbf{X} \triangleq \left\{ x \in \mathbb{R}^n \mid l^i \leq x^i \leq u^i,\ i \in \{1, \ldots, n\} \right\}, \tag{3.2.1b}$$

with $-\infty \leq l^i < u^i \leq \infty$ for $i \in \{1, \ldots, n\}$, where the cost function $f \colon \mathbb{R}^n \to \mathbb{R}$ is (at least) Lipschitz continuous.

We assume that the function $f(\cdot)$ cannot be evaluated exactly, but that it can be approximated by functions $f^* \colon \mathbb{R}^q_+ \times \mathbb{R}^n \to \mathbb{R}$, and that $\varepsilon \in \mathbb{R}^q_+$ is a vector of fixed dimension $q \in \mathbb{N}$ that contains the tolerance settings of the PDE, ODE and algebraic equation solvers. We will assume that $f(\cdot)$ and its approximating functions $\{f^*(\varepsilon, \cdot)\}_{\varepsilon \in \mathbb{R}^q_+}$ have the following properties.

Assumption 3.2.1

1. *There exists an error bound function* $\varphi\colon \mathbb{R}^q_+ \to \mathbb{R}_+$ *such that for any compact set* $\mathbf{S} \subset \mathbf{X}$*, there exists an* $\varepsilon_{\mathbf{S}} \in \mathbb{R}^q_+$ *and a scalar* $K_{\mathbf{S}} \in (0,\infty)$ *such that for all* $x \in \mathbf{S}$ *and for all* $\varepsilon \in \mathbb{R}^q_+$*, with* $\varepsilon \leq \varepsilon_{\mathbf{S}}$*,*

$$|f^*(\varepsilon,x) - f(x)| \leq K_{\mathbf{S}}\,\varphi(\varepsilon). \tag{3.2.2a}$$

Furthermore,

$$\lim_{\|\varepsilon\|\to 0} \varphi(\varepsilon) = 0. \tag{3.2.2b}$$

2. *The function* $f\colon \mathbb{R}^n \to \mathbb{R}$ *is at least locally Lipschitz continuous.* □

Remark 3.2.2 The functions $\{f^*(\varepsilon,\cdot)\}_{\varepsilon\in\mathbb{R}^q_+}$ may be discontinuous. □

Examples of error bound functions $\varphi(\cdot)$ can be found in Section 3.5 and in Polak and Wetter (2003).

Next, we state an assumption on the level sets of the family of approximating cost functions. To do so, we first define the notion of a level set.

Definition 3.2.3 (Level Set) *Given a function* $f\colon \mathbb{R}^n \to \mathbb{R}$ *and an* $\alpha \in \mathbb{R}$*, such that* $\alpha > \inf_{x\in\mathbb{R}^n} f(x)$*, we will say that the set* $\mathbf{L}_\alpha(f) \subset \mathbb{R}^n$*, defined as*

$$\mathbf{L}_\alpha(f) \triangleq \{x \in \mathbb{R}^n \mid f(x) \leq \alpha\}, \tag{3.2.3}$$

is a level set *of* $f(\cdot)$*, parametrized by* α*.* □

Assumption 3.2.4 (Compactness of Level Sets) *Let the functions* $\{f^*(\varepsilon,\cdot)\}_{\varepsilon\in\mathbb{R}^q_+}$ *be as in Assumption 3.2.1 and let* $\mathbf{X}\subset\mathbb{R}^n$ *be the constraint set. Let* $x_0\in\mathbf{X}$ *be the initial iterate and* $\varepsilon_0\in\mathbb{R}^q_+$ *be the initial solver tolerance. Then, we assume that there exists a compact set* $\mathbf{C}\subset\mathbb{R}^n$ *such that for all* $\varepsilon\in\mathbb{R}^q_+$*, with* $\varepsilon\leq\varepsilon_0$*,*

$$\mathbf{L}_{f^*(\varepsilon_0,x_0)}\big(f^*(\varepsilon,\cdot)\big)\cap\mathbf{X}\subset\mathbf{C}. \tag{3.2.4}$$

□

3.3 Precision Control for Generalized Pattern Search Algorithms

3.3.1 Characterization of Generalized Pattern Search Algorithms

There exist different geometrical characterizations for pattern search algorithms, and a general framework is presented in the review by Kolda et al. (2003). To focus on the explanation of our precision control algorithms without having to repeat the excellent discussions by Kolda et al. (2003), we will use a simple implementation of pattern search algorithms to explain our precision control algorithms. In particular, we will assume that

the search directions are the columns of the matrix

$$D \triangleq [-e_1, +e_1, \ldots, -e_n, +e_n] \in \mathbb{Z}^{n\times 2n}, \tag{3.3.1}$$

which suffices for box-constrained problems. Furthermore, we assume that the sequence of mesh size parameters, which parametrize the minimum distance between iterates, is constructed as follows.

Assumption 3.3.5 (k-th Mesh Size Parameter) *Let $r, s_0, k \in \mathbb{N}$, with $r > 1$, and $\{t_i\}_{i=0}^{k-1} \subset \mathbb{N}$. We assume that the sequence of mesh size parameters $\{\Delta_k\}_{k=0}^{\infty}$ satisfies*

$$\Delta_k \triangleq \frac{1}{r^{s_k}}, \tag{3.3.2a}$$

where for $k > 0$

$$s_k \triangleq s_0 + \sum_{i=0}^{k-1} t_i. \tag{3.3.2b}$$

□

With this construction, all iterates lie on nested rational meshes of the form

$$\mathbb{M}_k \triangleq \{x_0 + \Delta_k D m \mid m \in \mathbb{N}^{2n}\}. \tag{3.3.3}$$

We will now characterize the set-valued maps that determine the mesh points for the "global" and "local" searches.

Definition 3.3.6 *Let $\underline{\mathbf{X}}_k \subset \mathbb{R}^n$ and $\underline{\Delta}_k \subset \mathbb{Q}_+$ be the sets of all sequences containing $k+1$ elements, let $\mathbb{M}_k$ be the current mesh as defined in* (3.3.3)*, and let $\varepsilon \in \mathbb{R}^q_+$ be the solver tolerance.*

1. *We define the* global search set map *to be any set-valued map*

$$\gamma_k \colon \underline{\mathbf{X}}_k \times \underline{\Delta}_k \times \mathbb{R}^q_+ \to \left(2^{\mathbb{M}_k} \cap \mathbf{X}\right) \cup \emptyset \tag{3.3.4a}$$

   whose image $\gamma_k(\underline{x}_k, \underline{\Delta}_k, \varepsilon)$ contains only a finite number of mesh points.

2. *We will call $\mathcal{G}_k \triangleq \gamma_k(\underline{x}_k, \underline{\Delta}_k, \varepsilon)$ the* global search set.

3. *We define the directions for the local search as $D \triangleq [-e_1, +e_1, \ldots, -e_n, +e_n]$.*

4. *We will call*

$$\mathcal{L}_k \triangleq \left\{x_k + \Delta_k D e_i \mid i \in \{1, \ldots, 2n\}\right\} \cap \mathbf{X} \tag{3.3.4b}$$

   the local search set. □

Remark 3.3.7

1. As we shall see, the global search affects only the efficiency of the algorithm but not its convergence properties. Any heuristic procedure that leads to a finite number of function evaluations can be used for $\gamma_k(\cdot,\cdot,\cdot)$. Thus, the elements in $\mathcal{G}_k$ can be determined using a search procedure on surrogate cost functions, as in Dennis and Torczon (1997), Torczon and Trosset (1998), Serafini (1998) and Booker et al. (1999), if the search procedure is a finite process.

2. The empty set is included in the range of $\gamma_k(\cdot,\cdot,\cdot)$ to allow omitting the global search.

□

3.3.2 Adaptive Precision GPS Algorithm Models

We will now present our GPS algorithm models with adaptive precision cost function evaluations. We will first present an algorithm that simultaneously decreases Δ_k and ε.

Algorithm 3.3.8 (GPS Algorithm Model with Simultaneous Decrease of ε and Δ_k)

Data: Parameter $\zeta \geq 0$;
Initial iterate $x_0 \in \mathbf{X}$;
Mesh size divider $r \in \mathbb{N}$, with $r > 1$;
Initial mesh size exponent $s_0 \in \mathbb{N}$.

Maps: Global search set map $\gamma_k \colon \underline{\mathbf{X}_k} \times \underline{\Delta}_k \times \mathbb{R}^q_+ \to \left(2^{\mathbb{M}_k} \cap \mathbf{X}\right) \cup \emptyset$;
$\varphi \colon \mathbb{R}^q_+ \to \mathbb{R}_+$ as in Assumption 3.2.1;
Function $\rho \colon \mathbb{R}_+ \to \mathbb{R}^q_+$ (to assign ε), such that the composition
$\varphi \circ \rho \colon \mathbb{R}_+ \to \mathbb{R}_+$ is strictly monotone increasing and satisfies
$\varphi(\rho(\Delta))/\Delta \to 0$, as $\Delta \to 0$.

Step 0: Initialize $k = 0$, $\Delta_0 = 1/r^{s_0}$, and $\varepsilon = \rho(1)$.

Step 1: Global Search
Construct the global search set $\mathcal{G}_k = \gamma_k(\underline{x}_k, \underline{\Delta}_k, \varepsilon)$.
If $f^*(\varepsilon, x') - f^*(\varepsilon, x_k) < -\zeta\varphi(\varepsilon)$ for any $x' \in \mathcal{G}_k$, go to Step 3;
else, go to Step 2.

Step 2: Local Search
Evaluate $f^*(\varepsilon, \cdot)$ for any $x' \in \mathcal{L}_k$ until some $x' \in \mathcal{L}_k$
satisfying $f^*(\varepsilon, x') - f^*(\varepsilon, x_k) < -\zeta\varphi(\varepsilon)$ is obtained, or until all points
in $\mathcal{L}_k$ are evaluated.

Step 3: Parameter Update
If there exists an $x' \in \mathcal{G}_k \cup \mathcal{L}_k$ satisfying $f^*(\varepsilon, x') - f^*(\varepsilon, x_k) < -\zeta\varphi(\varepsilon)$,
set $x_{k+1} = x'$, $s_{k+1} = s_k$, $\Delta_{k+1} = \Delta_k$, and do not change ε;
else, set $x_{k+1} = x_k$, $s_{k+1} = s_k + t_k$, with $t_k \in \mathbb{N}_+$ arbitrary,
$\Delta_{k+1} = 1/r^{s_{k+1}}$, and $\varepsilon = \rho(\Delta_{k+1}/\Delta_0)$.

Step 4: Replace k by $k+1$, and go to Step 1.

Remark 3.3.9

1. We allow setting $\zeta = 0$ to obtain a GPS algorithm without imposing a sufficient decrease condition. In proving that $\liminf_{k\to\infty} \Delta_k = 0$, we will make use of the fact that the iterates x_k are contained in a compact set and lie on a rational lattice

in which the spacing of the elements depends on Δ_k, and hence simple decrease suffices to accept an iterate (Kolda et al., 2003).

2. To ensure that ε does not depend on the scaling of Δ_0, we normalize the argument of $\rho(\cdot)$. In particular, we want to decouple ε from the user's choice of the initial mesh size parameter.

3. In Step 2, once a (sufficient) decrease in cost is obtained, one can proceed to Step 3. However, one is allowed to evaluate the approximating cost function at more points in $\mathcal{L}_k$ in an attempt to obtain a larger decrease in cost. However, one is allowed to proceed to Step 3 only after either a (sufficient) decrease in cost has been obtained, or after *all* points in $\mathcal{L}_k$ were tested.

4. In Step 3, one is not restricted to accept the point $x' \in \mathcal{G}_k \cup \mathcal{L}_k$ that gives lowest cost. But the mesh size parameter Δ_k is reduced *only* if there exists no $x' \in \mathcal{G}_k \cup \mathcal{L}_k$ satisfying $f^*(\varepsilon, x') - f^*(\varepsilon, x_k) < -\zeta\varphi(\varepsilon)$.

5. To simplify the explanation of our precision control algorithms, we do not increase the mesh size parameter if the cost has been reduced. However, our global search allows searching on a coarser mesh $\widehat{\mathbb{M}} \subset \mathbb{M}_k$, and hence, our algorithm can easily be extended to include a rule for increasing Δ_k for a finite number of iterations.

6. Audet and Dennis (2003) update the mesh size parameter using the formula $\Delta_{k+1} = \tau^m \Delta_k$, where $\tau \in \mathbb{Q}$, $\tau > 1$, and m is any element of $\mathbb{Z}$. Thus, our update rule for

Δ_k is a special case of Audet's and Dennis' construction since we set $\tau = 1/r$, with $r \in \mathbb{N}_+$, $r \geq 2$ (so that $\tau < 1$) and $m \in \mathbb{N}$. We prefer our construction because we do not think it negatively affects the computational performance, but it leads to simpler convergence proofs. □

In Polak and Wetter (2003), we use the GPS Algorithm Model 3.3.8 with $\zeta = 0$ to extend the Hooke-Jeeves algorithm for use with adaptive precision cost function evaluations.

We will now present a modification of the GPS Algorithm Model 3.3.8 in which we decrease Δ_k only after $\varphi(\varepsilon)$ has been sufficiently decreased.

Algorithm 3.3.10 (GPS Algorithm Model)

Data: Parameters $\alpha \in (0, 1)$ and $\zeta \geq 0$;
Initial iterate $x_0 \in \mathbf{X}$;
Mesh size divider $r \in \mathbb{N}$, with $r > 1$;
Initial mesh size exponent $s_0 \in \mathbb{N}$.

Maps: Global search set map $\gamma_k : \underline{\mathbf{X}_k} \times \underline{\Delta}_k \times \mathbb{R}^q_+ \to \left(2^{\mathbb{M}_k} \cap \mathbf{X}\right) \cup \emptyset$;
$\varphi : \mathbb{R}^q_+ \to \mathbb{R}_+$ as in Assumption 3.2.1;
$\rho : \mathbb{N} \to \mathbb{R}^q_+$ (to assign ε) such that the
composition $\varphi \circ \rho : \mathbb{N} \to \mathbb{R}_+$ is strictly monotone decreasing and
satisfies $\varphi(\rho(N)) \to 0$, as $N \to \infty$.

Step 0: Initialize $k = 0$, $\Delta_0 = 1/r^{s_0}$, $N = 1$ and $\varepsilon = \rho(1)$.

Step 1: Global Search
Construct the global search set $\mathcal{G}_k = \gamma_k(\underline{x}_k, \underline{\Delta}_k, \varepsilon)$.
If $f^*(\varepsilon, x') - f^*(\varepsilon, x_k) < -\zeta\varphi(\varepsilon)$ for any $x' \in \mathcal{G}_k$, go to Step 3;
else, go to Step 2.

Step 2: Local Search
Evaluate $f^*(\varepsilon, \cdot)$ for any $x' \in \mathcal{L}_k$ until some $x' \in \mathcal{L}_k$
satisfying $f^*(\varepsilon, x') - f^*(\varepsilon, x_k) < -\zeta\varphi(\varepsilon)$ is obtained, or until all points
in $\mathcal{L}_k$ are evaluated.

Step 3: Parameter Update
If there exists an $x' \in \mathcal{G}_k \cup \mathcal{L}_k$ satisfying $f^*(\varepsilon, x') - f^*(\varepsilon, x_k) < -\zeta\varphi(\varepsilon)$,
    set $x_{k+1} = x'$, $s_{k+1} = s_k$, $\Delta_{k+1} = \Delta_k$, do not change N,
    and go to Step 4;
else,
    replace N by $N + 1$ and set $\varepsilon = \rho(N)$.
    If $\varphi(\varepsilon)^\alpha / \Delta_k < \Delta_k$,
        set $s_{k+1} = s_k + t_k$, with $t_k \in \mathbb{N}_+$ large enough
        such that $\varphi(\varepsilon)^\alpha / \Delta_{k+1} \geq \Delta_{k+1}$ (with $\Delta_{k+1} = 1/r^{s_{k+1}}$), and
        set $\Delta_{k+1} = 1/r^{s_{k+1}}$;
    else,
        set $s_{k+1} = s_k$ and $\Delta_{k+1} = \Delta_k$.
    Set $x_{k+1} = x_k$, and go to Step 4.

Step 4: Replace k by $k + 1$, and go to Step 1.

Remark 3.3.11 1. In Step 3, the mesh refinement is well defined because there always exists a $t_k \in \mathbb{N}_+$ for which $\varphi(\varepsilon)^\alpha \geq \Delta_{k+1}^2$, namely any $t_k \in \mathbb{N}_+$ that satisfies

$$t_k \geq \frac{2\log\Delta_k - \alpha\log\varphi(\varepsilon)}{2\log r}. \tag{3.3.5}$$

2. In Step 3, the test $\varphi(\varepsilon)^\alpha/\Delta_k < \Delta_k$, with $\alpha \in (0, 1)$, ensures that $\varphi(\varepsilon)/\Delta_k \to 0$, as $\Delta_k \to 0$, which is essential for proving convergence.

3. The algorithm parameter α can be used to control how fast Δ_k is decreased. The smaller $\alpha \in (0, 1)$, the later Δ_k is decreased. □

3.4 Convergence Analysis

3.4.1 Unconstrained Minimization

We will now establish the convergence properties of the GPS Algorithm Models 3.3.8 and 3.3.10 on unconstrained minimization problems, i.e., for $\mathbf{X} = \mathbb{R}^n$.

The following obvious result will be used to show that $\Delta_k \to 0$ as $k \to \infty$.

Proposition 3.4.12 *Any bounded subset of a mesh $\mathbb{M}_k$ contains only a finite number of mesh points.* □

Proposition 3.4.13 *Suppose that Assumption 3.2.4 is satisfied and let $\{\Delta_k\}_{k=0}^\infty \subset \mathbb{Q}_+$*

be the sequence of mesh size parameters constructed by GPS Algorithm Model 3.3.8 or 3.3.10. Then, $\liminf_{k\to\infty}\Delta_k = 0$.

Proof. We first prove the proposition for the GPS Algorithm Model 3.3.8. By (3.3.2a), $\Delta_k = 1/r^{s_k}$, where $r \in \mathbb{N}$ with $r > 1$, and $\underline{s_k} \subset \mathbb{N}$ is a nondecreasing sequence. For the sake of contradiction, suppose that there exists a $\Delta_{k^*} \in \mathbb{Q}_+$, such that $\Delta_k \geq \Delta_{k^*}$ for all $k \in \mathbb{N}$. Then there exists a corresponding $s_{k^*} = \max_{k\in\mathbb{N}} s_k$, and the finest possible mesh is $\mathbb{M}_{k^*} \triangleq \{x_0 + (1/r^{s_{k^*}})\, D\, m \,|\, m \in \mathbb{N}^{2n}\}$.

Next, by Assumption 3.2.4, there exists a compact set $\mathbf{C}$, such that $\mathbf{L}_{f^*(\varepsilon_0,x_0)}\big(f^*(\varepsilon,\cdot)\big) \subset \mathbf{C}$ for all $\varepsilon \in \mathbb{R}^q_+$, with $\varepsilon \leq \varepsilon_0 = \rho(1)$. Therefore, it follows from Proposition 3.4.12 that $\mathbb{M}_{k^*} \cap \mathbf{L}_{f^*(\varepsilon_0,x_0)}\big(f^*(\varepsilon,\cdot)\big)$ contains only a finite number of mesh points for any $\varepsilon \in \mathbb{R}^q_+$, with $\varepsilon \leq \rho(1)$. Thus, at least one point in $\mathbb{M}_{k^*}$ must belong to the sequence $\{x_k\}_{k=0}^{\infty}$ infinitely many times. Furthermore, because $\{s_k\}_{k=0}^{\infty} \subset \mathbb{N}$ is nondecreasing with s_{k^*} being its maximal element, it follows that $\varepsilon = \varepsilon^* = \rho(\Delta_{k^*}/\Delta_0)$ for all iterations $k \geq k^*$. Hence the sequence $\{f^*(\varepsilon^*,x_k)\}_{k=0}^{\infty}$ cannot satisfy $f^*(\varepsilon^*,x_{k+1}) - f^*(\varepsilon^*,x_k) < -\zeta\varphi(\varepsilon^*)$ for all $k \geq k^*$, which contradicts the constructions in Algorithm 3.3.8.

We now prove the proposition for the GPS Algorithm Model 3.3.10. Suppose that $\liminf_{k\to\infty}\Delta_k \neq 0$. Then, there exists only a finite number of iterations in which there exists no $x' \in \mathcal{G}_k \cup \mathcal{L}_k$ that satisfies $f^*(\varepsilon,x') - f^*(\varepsilon,x_k) < -\zeta\varphi(\varepsilon)$ because otherwise, N is replaced by $N+1$ an infinite number of times in Step 3, from which follows that $\varphi(\rho(N))^{\alpha} \to 0$, as $N \to \infty$, and hence $\Delta_k \to 0$, as $k \to \infty$. Thus, there exists an $N^* \in \mathbb{N}$ and a corresponding $k^* \in \mathbb{N}$ such that $N \leq N^*$, $\Delta_k = \Delta_{k^*}$ and $\varepsilon^* = \rho(N^*)$ for all $k \geq k^*$,

and the finest possible mesh is $\mathbb{M}_{k^*} \triangleq \left\{x_0 + \Delta_{k^*} Dm \mid m \in \mathbb{N}^{2n}\right\}$.

By Assumption 3.2.4, there exists a compact set $\mathbf{C}$, such that $\mathbf{L}_{f^*(\varepsilon_0,x_0)}\left(f^*(\varepsilon,\cdot)\right) \subset \mathbf{C}$, for all $\varepsilon \in \mathbb{R}^q_+$, with $\varepsilon \leq \varepsilon_0 = \rho(1)$. Hence, it follows from Proposition 3.4.12 that $\mathbb{M}_{k^*} \cap \mathbf{L}_{f^*(\varepsilon_0,x_0)}\left(f^*(\varepsilon,\cdot)\right)$ contains only a finite number of mesh points for all $\varepsilon \leq \varepsilon_0$. Thus, at least one point in $\mathbb{M}_{k^*}$ must belong to the sequence $\{x_k\}_{k=0}^{\infty}$ infinitely many times. Hence, the sequence $\{f^*(\varepsilon^*,x_k)\}_{k=k^*}^{\infty}$ cannot satisfy $f^*(\varepsilon^*,x_{k+1}) - f^*(\varepsilon^*,x_k) < -\zeta\varphi(\varepsilon^*)$ for all $k \geq k^*$, which contradicts the constructions in Algorithm 3.3.10. □

Having shown that $\liminf_{k\to\infty} \Delta_k = 0$, we can use the notion of a refining subsequence as introduced by Audet and Dennis (2003).

Definition 3.4.14 (Refining Subsequence) *Consider a sequence $\{x_k\}_{k=0}^{\infty}$ constructed by GPS Algorithm Model 3.3.8 or by GPS Algorithm Model 3.3.10. We will say that the subsequence $\{x_k\}_{k\in\mathbf{K}}$ is the* refining subsequence*, if $\Delta_{k+1} < \Delta_k$ for all $k \in \mathbf{K}$, and $\Delta_{k+1} = \Delta_k$ for all $k \notin \mathbf{K}$.* □

When the cost function $f(\cdot)$ is only locally Lipschitz continuous, we, as well as Audet and Dennis (2003), only get a weak characterization of limit points of the refining subsequence, as we will now see.

We recall the definition of Clarke's generalized directional derivative (Clarke, 1990):

Definition 3.4.15 (Clarke's Generalized Directional Derivative) *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz continuous at the point $x^* \in \mathbb{R}^n$. Then,* Clarke's generalized directional

derivative *of $f(\cdot)$ at x^* in the direction $h \in \mathbb{R}^n$ is defined by*

$$d^0 f(x^*; h) \triangleq \limsup_{\substack{x \to x^* \\ t \downarrow 0}} \frac{f(x + t\,h) - f(x)}{t}. \tag{3.4.1}$$

□

Theorem 3.4.16 *Suppose that Assumptions 3.2.1 and 3.2.4 are satisfied, let D be as in Definition 3.3.6, and let $x^* \in \mathbb{R}^n$ be an accumulation point of the refining subsequence $\{x_k\}_{k \in \mathbf{K}}$, constructed by GPS Algorithm Model 3.3.8 or by GPS Algorithm Model 3.3.10. Then, for all $d \in D$,*

$$d^0 f(x^*; d) \geq 0. \tag{3.4.2}$$

Proof. The proof is identical for both algorithms. Let $\{x_k\}_{k \in \mathbf{K}}$ be the refining subsequence and, without loss of generality, suppose that $x_k \to^{\mathbf{K}} x^*$. By Assumption 3.2.4, there exists a compact set $\mathbf{C}$ such that $\mathbf{L}_{f^*(\varepsilon_0, x_0)}\big(f^*(\varepsilon, \cdot)\big) \subset \mathbf{C}$ for all $\varepsilon \in \mathbb{R}^q_+$, with $\varepsilon \leq \varepsilon_0 = \rho(1)$. Therefore, by Assumption 3.2.1, there exists an $\varepsilon_{\mathbf{L}} \in \mathbb{R}^q_+$ and a scalar $K_{\mathbf{L}} \in (0, \infty)$ such that, for all $x \in \mathbf{C}$ and for all $\varepsilon \in \mathbb{R}^q_+$, with $\varepsilon \leq \varepsilon_{\mathbf{L}}$, we have $|f^*(\varepsilon, x) - f(x)| \leq K_{\mathbf{L}}\, \varphi(\varepsilon)$. Because $f(\cdot)$ is locally Lipschitz continuous, its directional derivative $d^0 f(\cdot; \cdot)$ exists. The precision control schemes of Algorithm 3.3.8 and Algorithm 3.3.10 both imply that in all iterations $k \in \mathbf{K}$, ε is decreased, and furthermore that $f^*(\varepsilon, x_k +$

$\Delta_k d) - f^*(\varepsilon, x_k) \geq -\zeta\,\varphi(\varepsilon)$ for all $d \in D$ and for all $k \in \mathbf{K}$. Hence, for any $d \in D$,

$$\begin{aligned} d^0 f(x^*; d) &\triangleq \limsup_{\substack{x \to x^* \\ t \downarrow 0}} \frac{f(x + t\,d) - f(x)}{t} \\ &\geq \limsup_{k \in \mathbf{K}} \frac{f(x_k + \Delta_k d) - f(x_k)}{\Delta_k} \\ &\geq \limsup_{k \in \mathbf{K}} \frac{f^*(\varepsilon, x_k + \Delta_k d) - f^*(\varepsilon, x_k) - 2\,K_{\mathbf{L}}\,\varphi(\varepsilon)}{\Delta_k} \\ &\geq \limsup_{k \in \mathbf{K}} \frac{f^*(\varepsilon, x_k + \Delta_k d) - f^*(\varepsilon, x_k)}{\Delta_k} - \limsup_{k \in \mathbf{K}} 2\,K_{\mathbf{L}}\,\frac{\varphi(\varepsilon)}{\Delta_k} \\ &\geq -\limsup_{k \in \mathbf{K}} \zeta\,\frac{\varphi(\varepsilon)}{\Delta_k} - \limsup_{k \in \mathbf{K}} 2\,K_{\mathbf{L}}\,\frac{\varphi(\varepsilon)}{\Delta_k}. \end{aligned} \tag{3.4.3}$$

The last inequality holds because $\{x_k\}_{k \in \mathbf{K}}$ is the refining subsequence. Since by Proposition 3.4.13, $\Delta_k \to^{\mathbf{K}} 0$, it follows from the constructions in GPS Algorithm Model 3.3.8 and 3.3.10 that $\varphi(\varepsilon)/\Delta_k \to^{\mathbf{K}} 0$. □

We now state that pattern search algorithms with adaptive precision cost function evaluations converge to stationary points.

Theorem 3.4.17 (Convergence to a Stationary Point) *Suppose that Assumptions 3.2.1 and 3.2.4 are satisfied and, in addition, that $f(\cdot)$ is once continuously differentiable. Let $x^* \in \mathbb{R}^n$ be an accumulation point of the refining subsequence $\{x_k\}_{k \in \mathbf{K}}$, constructed by GPS Algorithm Model 3.3.8 or by GPS Algorithm Model 3.3.10. Then,*

$$\nabla f(x^*) = 0. \tag{3.4.4}$$

Proof. Since $f(\cdot)$ is once continuously differentiable, we have $d^0 f(x^*; h) = df(x^*; h) = \langle \nabla f(x^*), h \rangle$ for all $h \in \mathbb{R}^n$. It follows from Theorem 3.4.16 that $0 \leq \langle \nabla f(x^*), d \rangle$ for all $d \in D$, with D as in Definition 3.3.6. We can express any $h \in \mathbb{R}^n$ as

$$h = \sum_{i=1}^{2n} \alpha_i d_i, \quad d_i \in D, \quad \alpha_i \geq 0, \quad \forall i \in \{1, \ldots, 2n\}. \tag{3.4.5a}$$

Hence, $0 \leq \langle \nabla f(x^*), h \rangle$. Similarly, we can express the vector $-h$, as follows,

$$-h = \sum_{i=1}^{2n} \beta_i d_i, \quad d_i \in D, \quad \beta_i \geq 0, \quad \forall i \in \{1, \ldots, 2n\}. \tag{3.4.5b}$$

Hence, $0 \geq \langle \nabla f(x^*), h \rangle$, which implies that $0 = \langle \nabla f(x^*), h \rangle$, and, since h is arbitrary, that $\nabla f(x^*) = 0$. □

3.4.2 Constrained Minimization

We now extend our convergence proofs to the box-constrained problem (3.2.1).

First, we introduce the notion of a tangent cone and a normal cone, which are defined as follows:

Definition 3.4.18 (Tangent and Normal Cone)

1. *Let* $\mathbf{X} \subset \mathbb{R}^n$ *be defined as in* (3.2.1b). *Then, we define the* tangent cone *to* $\mathbf{X}$ *at a point* $x^* \in \mathbf{X}$ *by*

$$\mathbf{T}_{\mathbf{X}}(x^*) \triangleq \{\mu(x - x^*) \mid \mu \geq 0, x \in \mathbf{X}\}. \tag{3.4.6a}$$

2. *Let* $\mathbf{T}_{\mathbf{X}}(x^*)$ *be as above. Then, we define the* normal cone *to* $\mathbf{X}$ *at* $x^* \in \mathbf{X}$ *by*

$$\mathbf{N}_{\mathbf{X}}(x^*) \triangleq \{v \in \mathbb{R}^n \mid \forall t \in \mathbf{T}_{\mathbf{X}}(x^*), \langle v, t \rangle \leq 0\}. \tag{3.4.6b}$$

□

Next, for $x^* \in \mathbf{X}$ we will introduce a function whose range is defined by those column vectors of the search direction matrix $D \in \mathbb{Z}^{n \times 2n}$ that are required to generate the tangent cone $\mathbf{T}_{\mathbf{X}}(x^*)$. This will facilitate the extension of Theorem 3.4.16 to box-constrained problems.

Definition 3.4.19

1. *For* $D = [-e_1, +e_1, \ldots, -e_n, +e_n]$, *we denote by* $\bar{D} \subset D$ *any matrix that is constructed by deleting columns of D, and we define* $\mathbb{D} \triangleq \{\bar{D} \mid \bar{D} \subset D\}$ *to be the set of all matrices constructed by deleting columns of D.*

2. *Let* $\mathbf{X} \subset \mathbb{R}^n$ *and* $\mathbf{T}_{\mathbf{X}}(\cdot)$ *be as in Definition 3.4.18. We define* $\delta\colon \mathbb{R}^n \to \mathbb{D}$ *to be any function such that (i) for any* $x \in \mathbf{X}$, *all columns of* $\delta(x)$ *belong to* $\mathbf{T}_{\mathbf{X}}(x)$, *and (ii)* $\mathbf{T}_{\mathbf{X}}(x)$ *can be generated by nonnegative linear combinations of the columns of* $\delta(x)$.

□

Since we defined $D \triangleq [-e_1, +e_1, \ldots, -e_n, +e_n]$ and since $\mathbf{X}$ is a box-constrained set, the function $\delta(\cdot)$ is unique.

We can now state that the GPS Algorithm Models 3.3.8 and 3.3.10 generate sequences of iterates which contain accumulation points that are feasible stationary points of problem (3.2.1).

Theorem 3.4.20 (Convergence to a Feasible Stationary Point)

Suppose that Assumptions 3.2.1 and 3.2.4 are satisfied and that $f(\cdot)$ is once continuously differentiable. Let $x^* \in \mathbf{X}$ be an accumulation point of the refining subsequence $\{x_k\}_{k\in\mathbf{K}}$ constructed by GPS Algorithm Model 3.3.8 or by GPS Algorithm Model 3.3.10 in solving problem (3.2.1)*. Then,*

$$\langle \nabla f(x^*), t \rangle \geq 0, \qquad \forall t \in \mathbf{T}_{\mathbf{X}}(x^*), \tag{3.4.7a}$$

and

$$-\nabla f(x^*) \in \mathbf{N}_{\mathbf{X}}(x^*). \tag{3.4.7b}$$

Proof. If x^* is in the interior of $\mathbf{X}$, then the result reduces to Theorem 3.4.17.

Let $x^* \in \partial\mathbf{X}$ and let $\delta(\cdot)$ be as in Definition 3.4.19. Since the number of constraints is finite, there exists for all $x^* \in \partial\mathbf{X}$ a corresponding $\rho_{x^*} > 0$ such that for all $x_k \in \mathbf{X}$,

with $\|x_k - x^*\| < \rho_{x^*}$, $\mathbf{T_X}(x^*) \subset \mathbf{T_X}(x_k)$ and hence $\delta(x^*) \subset \delta(x_k)$. Thus, there exists an infinite subset $\mathbf{K'} \subset \mathbf{K}$ in which $x_k \to^{\mathbf{K'}} x^*$ and $\delta(x^*) \subset \delta(x_k)$. By Theorem 3.4.16, we have $\langle \nabla f(x^*), d \rangle \geq 0$ for all $d \in \delta(x^*)$. By the definition of $\delta(x^*)$, every $t \in \mathbf{T_X}(x^*)$ can be expressed as a nonnegative linear combination of columns of $\delta(x^*)$. Therefore, $\langle \nabla f(x^*), t \rangle \geq 0$. It follows directly that $\langle -\nabla f(x^*), t \rangle \leq 0$, which shows that $-\nabla f(x^*) \in \mathbf{N_X}(x^*)$. □

When the function $f(\cdot)$ is only locally Lipschitz continuous, we obtain following corollary which follows directly from Theorem 3.4.16 and equation (3.4.7a).

Corollary 3.4.21 *Let $\delta\colon \mathbb{R}^n \to \mathbb{D}$ be as in Definition 3.4.19. Suppose that the assumptions of Theorem 3.4.20 are satisfied, but $f(\cdot)$ were only locally Lipschitz continuous. Then,*

$$d^0 f(x^*; d) \geq 0, \qquad \forall d \in \delta(x^*). \tag{3.4.8}$$

□

3.5 Numerical Experiments

In all numerical experiments, in the GPS Algorithm Models 3.3.8 and 3.3.10 we set the mesh size divider $r = 2$ and the initial mesh size exponent $s_0 = 0$. If no (sufficient) decrease in cost has been obtained, then we divide the mesh size parameter Δ_k by a factor of two. Hence, if $\mathbf{K}$ denotes the set that contains the iteration indices of the refin-

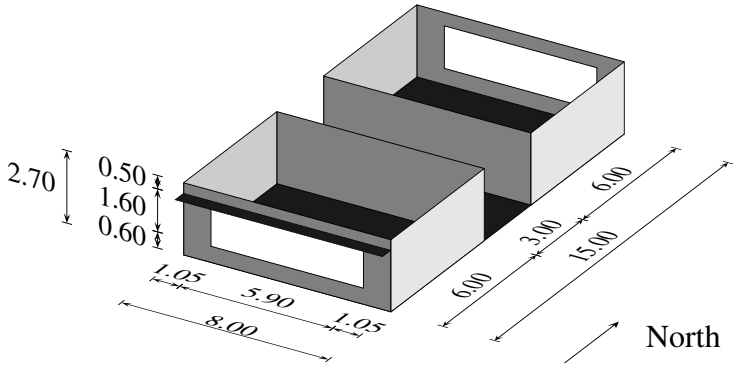


Figure 3.1: Thermal zones used for computing the building's annual source energy consumption.

ing subsequence, as defined in Definition 3.4.14, then in GPS Algorithm Models 3.3.8 and 3.3.10 $t_k = 1$ for $k \in \mathbf{K}$ and $t_k = 0$ for $k \notin \mathbf{K}$.

3.5.1 Cost Function defined on the Solutions of a DAE System

In this numerical experiment, we minimized the annual source energy consumption of the office rooms shown in Figure 3.1. Three thermal zones were simulated: A north facing room, a south facing room and a hallway between the two rooms. We assumed that all rooms that are adjacent to the three rooms have the same temperatures and radiative heat gains as the simulated rooms.

The building has a high thermal mass. The walls are made of concrete and have 20 cm exterior insulation. The windows are double-pane windows and have an exterior shading device with a solar and visible transmittance of 30% and a reflectance of 50%. The exterior shading device is activated if the total solar radiation on the window exceeds a setpoint. The south window has a shading overhang. The north and south zones

have daylighting controls with an illuminance setpoint of 500 lux three meters from the window. We used TMY2 weather data for Houston Intercontinental, TX.

The components of the design parameter $x \in \mathbb{R}^n$ are the window sizes for the south and north facing windows, the depth of an overhang placed above the south facing window, and two control setpoints that activate shading devices outside the north and south facing windows, hence $n = 5$.

3.5.1.1 Exact Cost Function

The cost function is once continuously differentiable and defined as

$$f(x) \triangleq F\big(z(x,1)\big), \tag{3.5.1}$$

where $z(x,1)$ is the solution of a semi-explicit nonlinear DAE system with index one (Brenan et al., 1989) of the form

$$\begin{aligned} \frac{dz(x,t)}{dt} &= h\big(x, z(x,t), \mu\big), \qquad t \in [0,\ 1], && \text{(3.5.2a)} \\ z(x,0) &= z_0(x), && \text{(3.5.2b)} \\ \gamma\big(x, z(x,t), \mu\big) &= 0, && \text{(3.5.2c)} \end{aligned}$$

where $h\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^m$, $z_0\colon \mathbb{R}^n \to \mathbb{R}^m$ and $\gamma\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^l$ are once Lipschitz continuously differentiable in all arguments. For all $x \in \mathbb{R}^n$ and for all $z(\cdot,\cdot) \in \mathbb{R}^m$,

equation (3.5.2c) has a unique solution $\mu^*(x,z) \in \mathbb{R}^l$ and the matrix with partial derivatives $\partial\gamma(x,z(x,t),\mu^*(x,z))/\partial\mu \in \mathbb{R}^{l\times l}$ is non-singular. Thus, by using the Implicit Function Theorem and standard theory of ordinary differential equations (Polak, 1997) one can show that there exists a unique once continuously differentiable function $z(\cdot,1)$ and hence $f(\cdot)$ is once continuously differentiable. In this experiment, $m = 104$ and $l = 4$, and we defined the function $F(\cdot)$ in (3.5.1) as

$$F\big(z(x,1)\big) \;\triangleq\; \frac{z^1(x,1)}{\eta_h} + \frac{z^2(x,1)}{\eta_c} + 3\,z^3(x,1), \tag{3.5.3}$$

where $z^1(x,1)$ and $z^2(x,1)$ are the annual heating and cooling loads of the rooms, $z^3(x,1)$ is the electricity consumption for lighting the rooms, and $\eta_h = 0.44$ and $\eta_c = 0.77$ are plant efficiencies that relate the annual room load to the source energy consumption for heating and cooling generation, including electricity consumption for fans and pumps (Huang and Franconi, 1999). The electricity consumption is multiplied by a factor of three to convert site electricity to source fuel energy consumption.

3.5.1.2 Approximating Cost Functions

To compute approximations to the cost function $f(\cdot)$, we had to write a thermal building and daylighting simulation program, called *BuildOpt*, that we present in Chapter 2, because existing thermal building and daylighting simulation programs are built on models that do not satisfy the smoothness assumptions required to prove existence,

uniqueness and differentiability of $z(\cdot, 1)$. The program BuildOpt is described in detail in Chapter 2 and in Appendix A. We will here only present a brief overview of BuildOpt to show the complexity of the simulation model, and refer to Chapter 2 and Appendix A for a more detailed description.

BuildOpt is a complex program that consists of two parts. The first part, which we will call the *simulation model generator*, parses a text input file with the detailed description of the building geometry, the building materials and the expected occupancy behavior – which, for our problem, was $1,700$ lines long – and then generates a simulation model for the particular building, i.e., the functions $h(\cdot,\cdot,\cdot)$, $z_0(\cdot)$ and $\gamma(\cdot,\cdot,\cdot)$ of the DAE system (3.5.2). These functions are representations of various detailed models for the heat and daylighting transfer processes and for the building control systems. For example, the heat conduction in walls and ceilings are modeled using the Galerkin finite element method (Evans, 1998; Strang and Fix, 1973), and the transmittances of solar radiation and daylight through the windows are modeled using state-of-the-art optical calculations similar to those in commercial programs (Finlayson et al., 1993; Winkelmann, 2001). There is also a detailed daylighting model that computes the available daylight at various locations in the building for different building and window configurations, and there are models for various control systems such as for the room lighting system and for the heating and cooling system.

The second part of BuildOpt, to which our simulation model generator was linked, is the commercial solver DASPK (Brown et al., 1994, 1998).

The total size of BuildOpt is $38,000$ of C/C++ and Fortran code, of which $30,000$ lines (1.2 MB) of C/C++ code represent the simulation model generator and $8,000$ lines (0.3 MB) of Fortran code represent the commercial solver DASPK.

We constructed the models in such a way that the functions $h(\cdot,\cdot,\cdot)$, $z_0(\cdot)$, and $\gamma(\cdot,\cdot,\cdot)$ are once Lipschitz continuously differentiable in all arguments, which has not been done before for thermal building and daylighting simulation programs. This required various smoothing techniques to replace conditional statements, which was in fact required to achieve convergence of the DASPK solver when the solver tolerance was tight.

In Appendix B, we present validation results of BuildOpt. The validation results of BuildOpt show good agreement with the results of the other validated programs.

On a 2.2 GHz AMD processor running Linux with the $2.4.18-3$ kernel, the computation time for one cost function evaluation was 24 sec for a solver tolerance of $\varepsilon = 10^{-1}$, 2 min 22 sec for $\varepsilon = 10^{-2}$, 5 min 42 sec for $\varepsilon = 10^{-3}$, 16 min 30 sec for $\varepsilon = 10^{-4}$, and 33 min 23 sec for $\varepsilon = 10^{-5}$.

3.5.1.3 Optimization Algorithm

We solved the optimization problem with adaptive precision cost function evaluations using the Hooke-Jeeves algorithm of the GenOpt$^{(R)}$ 2.0.0 optimization program (Wetter, 2001, 2004) with the precision controlled as in GPS Algorithm Model 3.3.10. For comparison, we also solved the problem using the Hooke-Jeeves optimization algorithm with fixed precision cost function evaluations and $\zeta = 0$. In the optimization with

	$\zeta = 0$	$\zeta = 10^{-8}$	$\zeta = 10^{-6}$	$\zeta = 10^{-4}$	$\zeta = 10^{-2}$	Δ_{k^*}
$\alpha = 1/7$	0.27	0.27	0.27	0.28	0.55	$1/2$
$\alpha = 1/6$	0.33	0.33	0.33	0.31	0.61	$1/4$
$\alpha = 1/4$	0.35	0.35	0.35	0.31	0.74	$1/4$
$\alpha = 1/3$	0.55	0.55	0.55	0.60	1.21	$1/8$

Table 3.1: Normalized computation times required to solve the building energy optimization problem with Algorithm 3.3.10. For each α, the last column shows the smallest Δ_k used in the search.

fixed precision cost function evaluations, we set $\varepsilon = 10^{-5}$ and we allowed the mesh size to be decreased four times before the optimization stopped.

3.5.1.4 Precision Control

Present day DAE solvers, such as DASPK, typically control the local error at each time step and do not even attempt to control the global error directly. We assumed that the global error of the approximate solutions $z^*(\varepsilon, x, 1)$ is one order of magnitude greater than the local error. Hence, we set $\varphi(\varepsilon) = 10\varepsilon$. (Alternatively, we could have absorbed the factor 10 in the constant K_S in (3.2.2a).)

We defined $\rho\colon \mathbb{N} \to \mathbb{R}_+$ as $\rho(N) = 10^{-N}$ and increased precision four times. Thus, $\varepsilon = \rho(1) = 10^{-1}$ for the first iterations, and $\varepsilon = \rho(5) = 10^{-5}$ for the last iterations, which is equal to the precision used in the optimization with fixed precision cost function evaluations.

3.5.1.5 Numerical Results

All optimization runs were done on a 2.2 GHz AMD processor running Linux with the $2.4.18-3$ kernel. In Tab. 3.1, we show the values that we selected for the algorithm parameters $\alpha \in (0, 1)$ and $\zeta \geq 0$, the corresponding normalized computation times and in the last column the smallest mesh size parameter Δ_{k^*}. A computation time of "1" corresponds to 5.5 days of computing, which was the time required to solve the optimization problem with the Hooke-Jeeves algorithm with fixed precision cost function evaluations and $\zeta = 0$.

Note that in Algorithm 3.3.10, the parameter $\alpha \in (0, 1)$ is only used to adjust the mesh size parameter Δ_k so that $\varphi(\varepsilon)^\alpha \geq \Delta_k^2$. Since $\varphi(\cdot)$ depends only on N, it is possible to compute for each $N \in \mathbb{N}$ the corresponding mesh size parameter. Such a computation shows that the sequence of mesh size parameters Δ_k, and hence the sequence of iterates x_k, are identical for all $\alpha \leq 1/7$, with $\alpha > 0$, and fixed ζ. Thus, a further reduction of α does not reduce the computation time.

For $\alpha \leq 1/4$, with $\zeta \in \{0, 10^{-8}, 10^{-6}, 10^{-4}\}$, our precision control algorithm reduces the computation time by a factor of three to four. For $\alpha = 1/7$ and $\zeta \leq 10^{-4}$, our precision control subprocedure reduced the computation time from about five days to one day, making the optimization fast enough to be applicable in building design processes. For our optimization problem, $\alpha = 1/3$ and $\zeta \geq 10^{-2}$ turn out to be too big, and imposing a sufficient decrease condition by setting $\zeta > 0$ does not reduce the computation time. All optimization runs converged to $x^* = (1, 1, 1, 0.19, 0.048)^T$ and reduced the energy

consumption for lighting, cooling and heating by 4.6 % or $9.4\,\mathrm{kWh}/(\mathrm{m}^2\,\mathrm{a})$. The 4.6 % reduction is small, but not representative for average savings, because in the literature (Al-Homoud, 1997; Wetter and Wright, 2003a,b), savings of 5% to 30% in energy consumption for lighting, cooling and heating due to optimized building design have been reported. A reduction of 15%, which is more representative for average savings than the 4.6% that we achieved in our experiment, would correspond to a reduction in energy consumption for lighting, cooling and heating of $30\,\mathrm{kWh}/(\mathrm{m}^2\,\mathrm{a})$. For an average current energy cost of \$0.10 per kWh, this corresponds to annual savings of \$3 per square meter floor area, or to annual savings of $\$30,000$ for a large $10,000\,\mathrm{m}^2$ office building. As large buildings are often designed using energy simulations, and hence a computer simulation model exists for those buildings, the additional effort to do an optimization is only a few man hours. Thus, the return-of-investment is achieved within the first year of the building operation time.

We will now describe how the optimization runs with fixed and adaptive precision cost function evaluations, with $\zeta = 10^{-4}$ and $\alpha = 1/6$, converged to a minimum. Let the normalized distance of the k-th iterate $x_k \in \mathbb{R}^n$ to the minimizer $x^* \triangleq \arg\min_{x \in \mathbf{X}} f(x)$ be defined as

$$d(x_k) \triangleq \frac{\|x_k - x^*\|}{\|x_0 - x^*\|}, \tag{3.5.4}$$

where $x_0 \in \mathbb{R}^n$ is the initial iterate. Fig. 3.2 shows the cost function value and the

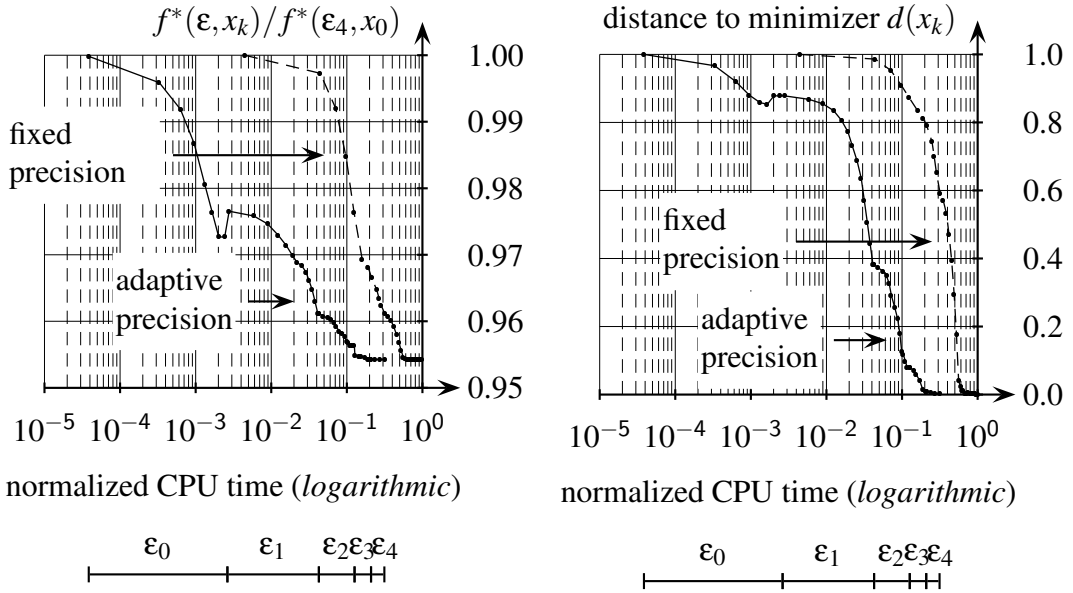


Figure 3.2: Normalized cost function value (left graph) and distance to the minimizer (right graph) as a function of the normalized CPU time in logarithmic scale. Below the graphs we show the intervals for which the precision ε has been kept constant. For the adaptive precision optimization, we used $\zeta = 10^{-4}$ and $\alpha = 1/6$. For better display of the early iterations, the time axis is in logarithmic scale.

distance to the minimizer as a function of the computation time. Below the axis we show when precision was increased. The different precision values are indicated by ε_m, $m \in \{0, 1, 2, 3, 4\}$, where $\varepsilon_m = 10^{-(m+1)}$. In the left graph, we can see that even for such coarse a precision as $\varepsilon = 10^{-1}$, the approximating cost function $f^*(10^{-1}, \cdot)$ allowed a substantial decrease in cost during the first 0.2% of the computation time.

3.5.2 Cost Function defined on the Solutions of a Nonlinear System of Equations

We will now present the computational performance of our precision control algorithms in minimizing a cost function that is defined on the approximate solutions of a nonlinear system of equations with 373 unknowns.

The objective is to fit four parameters of a detailed air-to-water cooling coil computer simulation model in such a way that the difference between simulated and measured coil air outlet temperature is minimized for a prescribed number of measurement points. The measurement data are the air and water inlet temperature, the air humidity ratio, the air and water mass flow, and the valve position of the throttle valve in the water circuit. There are 401 measurement data, equally spaced in time.

The simulation model (Xu and Haves, 2001) consists of a coupled system of nonlinear equations that is solved for 373 variables using Newton iterations. The model is static and was simulated in SPARK 1.0.3 (SPARK, 2003). For the range of measurement data, all model equations are once continuously differentiable, and the Jacobian matrix is non-singular in a neighborhood of the solution. Therefore, it follows from the Implicit Function Theorem (Polak, 1997) that the exact solution, and hence the cost function, is once continuously differentiable.

The design parameters are the air and water side heat transfer coefficients and two

parameters that define the valve characteristics. We controlled two precision parameters: the precision parameter for the Newton solver and the number of data points that were used in the data fit.

3.5.2.1 Exact Cost Function

We defined the exact cost function as follows. Let $\tau \triangleq [0,\ 1]$ denote the normalized time interval over which the measurement took place. Let $T_m\colon [0,\ 1] \to \mathbb{R}$ be the linear interpolation of the measured coil air outlet temperatures. For $x \in \mathbb{R}^n$ and $t \in \tau$, let $T_s(x,t) \in \mathbb{R}$ denote the exact solution of the system of equations that defines the coil air outlet temperature, obtained by using linearly interpolated measurement data. Then, for

$$e(x,t) \triangleq (T_m(t) - T_s(x,t))^2, \tag{3.5.5}$$

we defined the exact cost function

$$f(x) \triangleq \int_0^1 e(x,t)\,dt. \tag{3.5.6}$$

3.5.2.2 Approximating Cost Functions

The integral (3.5.6) cannot be evaluated because $T_s(x,t)$ can only be numerically approximated by an approximate solution $T_s^*(\varepsilon^1,x,t) \in \mathbb{R}$ with precision parameter $\varepsilon^1 \in \mathbb{R}_+$, and the integral can only be approximated by a quadrature formula. Thus, in com-

puting the approximating cost functions, we have two sets of approximations.

We approximated $f(\cdot)$ as follows. For some $\varepsilon_0 \in \mathbb{R}^2_+$, with $\varepsilon_0^2 \leq 1/2$, let $\varepsilon^1 \in (0, \varepsilon_0^1]$ denote the precision parameter of the Newton solver, and let $\varepsilon^2 \in (0, \varepsilon_0^2]$ denote the time interval for the quadrature formula. For $t \in \tau$, we approximated equation (3.5.5) by

$$e^*(\varepsilon^1, x, t) \triangleq (T_m(t) - T_s^*(\varepsilon^1, x, t))^2, \tag{3.5.7a}$$

using Newton iterations. The Newton solver in the SPARK program is set up in such a way that for any compact set $\mathbf{S} \subset \mathbf{X}$, there exists an $\varepsilon_{\mathbf{S}} \in \mathbb{R}_+$ and a $K'_{\mathbf{S}} \in (0, \infty)$ such that for all $x \in \mathbf{S}$, for all $t \in [0, 1]$ and for all $\varepsilon \in \mathbb{R}_+$, with $\varepsilon \leq \varepsilon_{\mathbf{S}}$,

$$|e^*(\varepsilon, x, t) - e(x, t)| \leq K'_{\mathbf{S}} \varepsilon. \tag{3.5.7b}$$

We approximated the integral (3.5.6) by

$$f^*(\varepsilon, x) \triangleq \sum_{i=0}^{N(\varepsilon^2)-1} \frac{e^*(\varepsilon^1, x, i/N(\varepsilon^2)) + e^*(\varepsilon^1, x, (i+1)/N(\varepsilon^2))}{2N(\varepsilon^2)}, \tag{3.5.8}$$

where $N(\varepsilon^2) \triangleq \lfloor 1/\varepsilon^2 \rfloor$.

It can be shown that for any compact set $\mathbf{S} \subset \mathbb{R}^n$, there exist a $K_{\mathbf{S}} \in (0, \infty)$ and an $\varepsilon_{\mathbf{S}} \in \mathbb{R}^2_+$ such that

$$|f^*(\varepsilon, x) - f(x)| \leq K_{\mathbf{S}} \|\varepsilon\|, \tag{3.5.9}$$

for all $x \in \mathbf{S}$ and for all $\varepsilon \in \mathbb{R}^2_+$, with $\varepsilon \leq \varepsilon_{\mathbf{S}}$. Therefore, $\varphi(\cdot)$ in Assumption 3.2.1 is $\varphi(\varepsilon) = \|\varepsilon\|$.

3.5.2.3 Optimization Algorithm

Numerical experiments showed that $f(\cdot)$ has several local minima, and some local minima have a cost function value that is three times larger than the best found solution. Therefore, we used the multi-start Hooke-Jeeves optimization algorithm from the GenOpt$^{(R)}$ 2.0.0 optimization program (Wetter, 2001, 2004) with four randomly selected initial iterates. We controlled the precision of the approximating cost functions using the precision control algorithm from GPS Algorithm Model 3.3.8. For comparison, we also solved the problem with the multi-start Hooke-Jeeves algorithm with fixed precision $\varepsilon = (10^{-10}, 1/400)^T$, $\zeta = 0$ and three step reductions.

3.5.2.4 Precision Control

To control $\varepsilon \in \mathbb{R}^2_+$, with $\varepsilon \leq \varepsilon_0$, as a function of the mesh size factor $\Delta \in \mathbb{Q}_+$, we defined $\rho : \mathbb{R}_+ \to \mathbb{R}^2_+$ as

$$\rho^i(\Delta) \triangleq \varepsilon^i_{\min} \left(\frac{\Delta}{\Delta_{min}} \right)^{\alpha^i}, \qquad \alpha^i > 1, \qquad i \in \{1, 2\}, \tag{3.5.10}$$

where $\Delta_{min} \triangleq \min_{k \in \mathbb{N}} \{\Delta_k\} = 1/8$ denotes the smallest mesh size parameter, and $\varepsilon_{min} = (10^{-10}, 1/400)^T$ is the precision parameter for the last iterations. Since $\alpha^i > 1$ for $i \in$

ε_0^1	ε_0^2	α^1	α^2	CPU time	$f^*(\varepsilon, x^*)$
10^{-10}	1/400	0	0	1	0.0236
10^{-1}	0.025	9.97	1.11	0.13	0.0242
10^{-1}	0.05	9.97	1.44	0.15	0.0225
10^{-1}	0.1	9.97	1.77	0.72	0.0216
10^{-1}	0.5	9.97	2.55	0.34	0.0217
10^{-1}	1	9.97	2.88	0.51	0.0221
10^{-2}	0.025	8.86	1.11	0.12	0.0242
10^{-2}	0.05	8.86	1.44	0.15	0.0225
10^{-2}	0.1	8.86	1.77	0.69	0.0216
10^{-2}	0.5	8.86	2.55	0.18	0.0217
10^{-2}	1	8.86	2.88	0.54	0.0221
10^{-3}	0.025	7.75	1.11	0.13	0.0242
10^{-3}	0.05	7.75	1.44	0.15	0.0225
10^{-3}	0.1	7.75	1.77	0.69	0.0216
10^{-3}	0.5	7.75	2.55	0.18	0.0217
10^{-3}	1	7.75	2.88	0.53	0.0221
average for adaptive precision				0.35	

Table 3.2: Initial precisions ε_0 used for approximating the cost functions, corresponding α, normalized computation time and best obtained local minima for all optimizations.

$\{1,2\}$, we have $\varphi(\rho(\Delta))/\Delta \to 0$, as $\Delta \to 0$.

To determine $\alpha > 1$, we selected different initial precisions $\varepsilon_0 \in \mathbb{R}^2_+$ and then computed α by solving (3.5.10) with $\Delta = \Delta_0 = 1$ and $\rho^i(1) = \varepsilon_0^i$ for $i \in \{1,2\}$. In particular, we set

$$\alpha^i = \frac{\log\left(\varepsilon_0^i/\varepsilon_{min}^i\right)}{\log\left(\Delta_0/\Delta_{min}\right)} = \frac{\log\left(\varepsilon_0^i/\varepsilon_{min}^i\right)}{\log 8}, \qquad i \in \{1,2\}. \tag{3.5.11}$$

Thus, for $\alpha^i > 1$, we need $\varepsilon_0^i > 8\,\varepsilon_{min}^i$ for $i \in \{1,2\}$.

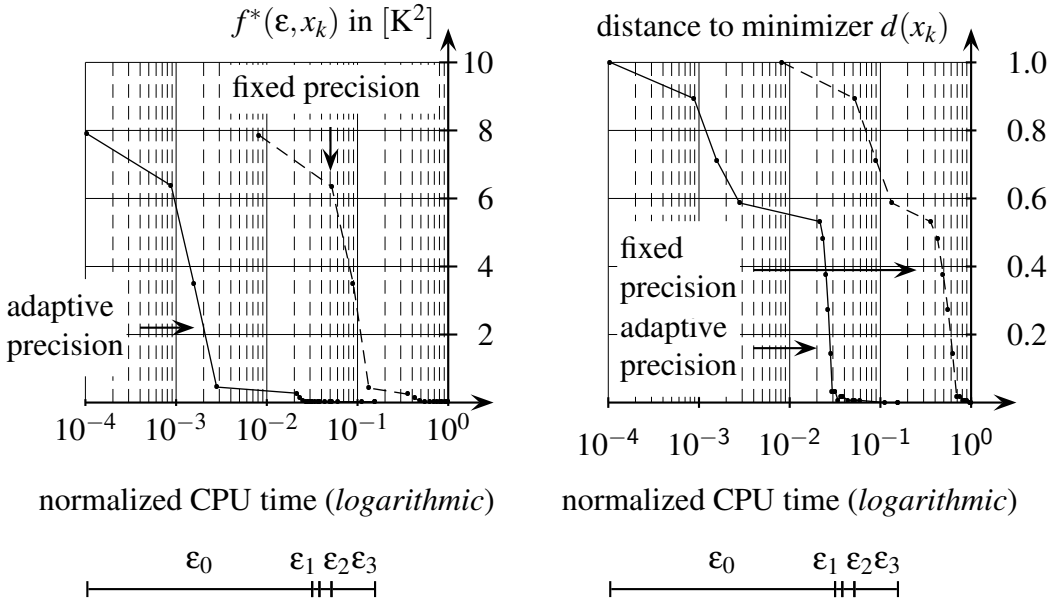


Figure 3.3: Normalized cost function value (left graph) and distance to the minimizer (right graph) as a function of the normalized computation time in logarithmic scale. Below the graphs we show the intervals for which the precision ε has been kept constant. For the adaptive precision optimization, we used $\zeta = 0$ and $\alpha = (9.97, 1.11)^T$. For better display of the early iterations, the time axis is in logarithmic scale.

3.5.2.5 Numerical Results

We did all computations on Linux computers with 2.2 GHz AMD processors and the $2.4.18 - 3$ kernel. The optimization with fixed precision cost function evaluations took 45 minutes.

Tab. 3.2 shows the different settings for ε_0, the corresponding α, the normalized computation time and the cost function values for the best obtained local minima. A normalized computation time of one corresponds to 45 minutes of computation time. All optimization runs obtained a similar reduction in cost. On average, our precision control scheme reduced the computation time by a factor of three.

We will now describe how the optimizations with fixed and adaptive precision cost function evaluations, with $\zeta = 0$ and $\alpha = (9.97, 1.11)^T$, converged to a local minimum. Let the normalized distance to the best local minimizer of the optimization that used fixed precision cost function evaluations be defined as in equation (3.5.4). Fig. 3.3 shows, for the optimizations with adaptive and fixed precision cost function evaluations, the cost function value and the normalized distance to the minimizer as a function of the computation time. For the initial iterate and the precision control parameter α used in the optimizations shown in Fig. 3.3, both algorithms converged to the same local minimum. Below the axis we show when precision was increased (the different precisions are indicated by ε_m, $m \in \{0, 1, 2, 3\}$). For this example, the precision control algorithm set $\varepsilon_0 = (10^{-1}, 0.025)^T$, $\varepsilon_1 = (10^{-4}, 0.012)^T$, $\varepsilon_2 = (10^{-7}, 0.0054)^T$, and $\varepsilon_3 = (10^{-10}, 0.0025)^T$. After 3% of the computation time that was required to solve the fixed precision optimization problem, the cost function values and the iterates of the adaptive precision optimization problem were already close to the minimum.

3.6 Conclusion

We have extended the family of GPS algorithms to a form that converges to a stationary point of a smooth cost function that cannot be evaluated exactly, but that can be approximated by a family of possibly discontinuous functions $\{f^*(\varepsilon, \cdot)\}_{\varepsilon \in \mathbb{R}^q_+}$. An important feature of our algorithms is that they use low-cost, coarse precision approxi-

mations to the cost function when far from a solution, with the precision progressively increased as a solution is approached. We have shown by numerical experiments that our precision control algorithms lead to considerable time savings over using high precision approximations to the cost function in all iterations.

Chapter 4

Optimization with Fixed Precision Cost Function Evaluations

4.1 Introduction

In this Chapter, we compare several deterministic and probabilistic optimization algorithms.

Detailed building simulation programs, such as EnergyPlus (Crawley et al., 2001) and TRNSYS (Klein et al., 1976), are increasingly being used to evaluate the cost function in optimization problems. Annual simulations with these programs are typically computationally expensive. Also, these simulation programs contain code features – such as adaptive integration meshes, iterative solvers that iterate until a convergence criterion is met (e.g., Newton solvers or bisection algorithms) and `if-then-else` logic – that can cause optimization algorithms that require smoothness of the cost function to fail, possibly far from a solution. In many of these building simulation programs the solvers are implemented in such a way that does not allow controlling the numerical error of the approximations to the state variables, and the solver tolerances are fixed at compile time, in some cases at coarse precision settings (Wetter and Polak, 2003; Wetter and Wright, 2003a). Thus, such computer code defines a numerical approximation to the cost function that is discontinuous with respect to the design parameter, and the discontinuities can be large.

It is, however, generally accepted in the simulation-based optimization community that the tolerances of such adaptive solvers must be tight if used in conjunction with optimization algorithms that require the cost function to be smooth (see for example

Bertsekas (1999), Gill et al. (1981) and Polak (1997)). If nonlinear programming algorithms are used to solve such optimization problems, then convergence to a stationary point can be established if the approximating cost functions, defined on the numerical approximations to the state variables, converge to a function that is once continuously differentiable in the design parameters, as the precision of the simulation is increased (see Chapter 3 and Polak (1997)), or if the approximation error goes to zero sufficiently fast as the optimization algorithm approaches a solution (Choi and Kelley, 2000; Kelley, 1999b). However, many building simulation programs do not satisfy these requirements and we observed (Wetter and Polak, 2003; Wetter and Wright, 2003a) that the solver tolerances are so coarse that optimization algorithms that require smoothness of the cost function can indeed fail far from a minimum.

Probabilistic optimization algorithms that do not require smoothness of the cost function have frequently been used to solve building optimization problems with a small number of simulations (see for example Wright and Farmani (2001), Caldas and Norford (2002) and Wetter and Wright (2003a)). However, these algorithms are stochastic in nature, and to achieve convergence with a high confidence, a large number of simulations is required (see for example Greenhalgh and Marshall (2000), Parsopoulos and Vrahatis (2002) and van den Bergh and Engelbrecht (2001)), which is impractical if the computation time required to evaluate the cost function is large.

Hence, it is not clear whether optimization algorithms that require smoothness of the cost function or stochastic algorithms, used with a low number of cost function evalua-

tions, perform better on building optimization problems in which the simulation program does not allow controlling the numerical error of the approximation to the state variables, and in which the design parameter is in $\mathbb{R}^n$ and has box-constraints. This is the question that we address in this chapter.

We compare the performance of nine optimization algorithms using numerical experiments. We compare direct search algorithms (the Coordinate Search, the Hooke-Jeeves, and two versions of the Nelder-Mead Simplex algorithm), stochastic population-based algorithms (a simple Genetic Algorithm and two Particle Swarm Optimization algorithms), a hybrid Particle Swarm Hooke-Jeeves algorithm and a gradient-based algorithm (the Discrete Armijo Gradient algorithm). Other promising methods that have successfully been used in simulation-based optimization, such as methods that use computationally cheap surrogate models (Booker et al., 1999; Dennis and Torczon, 1997; Serafini, 1998; Torczon and Trosset, 1998), are not covered here.

In the numerical experiments, we solved six optimization problems using the EnergyPlus (Crawley et al., 2001) whole building energy analysis program to evaluate the cost function. We used two simulation models, each with three different weather data. One simulation model is so that the cost function is rather smooth and the other is so that the cost function has discontinuities in the order of 2% of the cost function value. By selecting cost functions with different smoothness and identifying what code features can

cause such large discontinuities, we believe that our conclusions will also be applicable if other simulation programs are used to evaluate the cost function.

In the first section, we give a formal definition of the optimization problem, which is used to identify the terms that cause difficulties in solving the optimization problems. Next, we discuss the two simulation models. Then we discuss the main features of the optimization algorithms. Finally, we compare the performance of all optimization algorithms and discuss the causes of the observed discontinuities in the cost functions.

4.2 Optimization Problem

We consider problems of the form

$$\min_{x \in \mathbf{X}} f(x), \tag{4.2.1a}$$

where $x \in \mathbf{X}$ is the vector of independent variables, $f\colon \mathbf{X} \to \mathbb{R}$ is the cost function, and $\mathbf{X} \subset \mathbb{R}^n$ is the constraint set, defined as

$$\mathbf{X} \triangleq \left\{ x \in \mathbb{R}^n \mid l^i \leq x^i \leq u^i, i \in \{1, \ldots, n\} \right\}, \tag{4.2.1b}$$

with $-\infty \leq l^i < u^i \leq \infty$, for all $i \in \{1,\ldots,n\}$. The cost function $f(\cdot)$ is defined as

$$f(x) \triangleq F\big(z(x,1)\big), \tag{4.2.2}$$

where $F\colon \mathbb{R}^m \to \mathbb{R}$ is once continuously differentiable but defined on the solution of a coupled system of differential algebraic equations of the form

$$\frac{dz(x,t)}{dt} = h\big(x,\mu,p(x)\big), \qquad t \in [0,\ 1], \tag{4.2.3a}$$

$$z(x,0) = z_0(x), \tag{4.2.3b}$$

$$\gamma\big(x,z(x,t),\mu\big) = 0, \tag{4.2.3c}$$

where $h\colon \mathbb{R}^n \times \mathbb{R}^l \times \mathbb{R}^q \to \mathbb{R}^m$, $z_0\colon \mathbb{R}^n \to \mathbb{R}^m$ and $\gamma\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^l$ are once Lipschitz continuously differentiable in all arguments and the matrix with partial derivatives $\gamma_\mu(\cdot,\cdot,\cdot)$ is non-singular. The function $h(\cdot,\cdot,\cdot)$ describes the system dynamics, $z(\cdot,\cdot)$ is the vector of state variables whose components are the room air and construction temperatures (after the spatial discretization of the heat equation) and the heating, cooling and lighting power. The algebraic variable μ is the solution of (4.2.3c) and its components are state variables whose thermal capacities are assumed to be negligible, such as the window glass temperatures. The elements of the vector $p(\cdot) \in \mathbb{R}^q$ are the size of the cooling coil, heating coil, and supply and return fan.[1] Thus, the system (4.2.3)

[1]Clearly, the dependence of $h(\cdot,\cdot,\cdot)$ on $p(\cdot)$ can be eliminated by defining $h(x,\mu,p(x)) \triangleq \widetilde{h}(x,\mu)$, but we find it convenient for our discussion to show explicitly the dependence on $p(\cdot)$.

is a mathematical model of a thermal building energy calculation. Under appropriate assumptions, which are discussed in Chapter 2, one can show that (4.2.3) has a unique once continuously differentiable solution. Several optimization algorithms that use approximate solutions of (4.2.3), and progressively decrease the approximation error as the optimization approaches a solution, exist to solve (4.2.1), see Chapter 3.

However, in this Chapter, we are interested in the situation where EnergyPlus is used to compute approximate numerical solutions of (4.2.3). EnergyPlus contains several adaptive spatial and temporal grid generators, `if-then-else` logic and iterative solvers that iterate until a convergence criterion is met. Due to these code features a change in the independent variable x can cause a change in the sequence of code executions, which causes the approximate numerical solution of (4.2.3) to be discontinuous in x.[2] It is generally accepted in the simulation-based optimization community (Bertsekas, 1999; Gill et al., 1981; Polak, 1997) that in situations where (4.2.1) is solved using an optimization algorithm that requires the cost function to be once continuously differentiable, one needs to compute high-precision approximate solutions of (4.2.3). However, in EnergyPlus the numerical solvers and grid generators are spread throughout the code and most solver tolerance settings are fixed at compile time, in some cases at coarse precision. The implementation of the solvers is such that it does not seem possible to control the numerical error. Thus, optimization algorithms that require smoothness may fail far

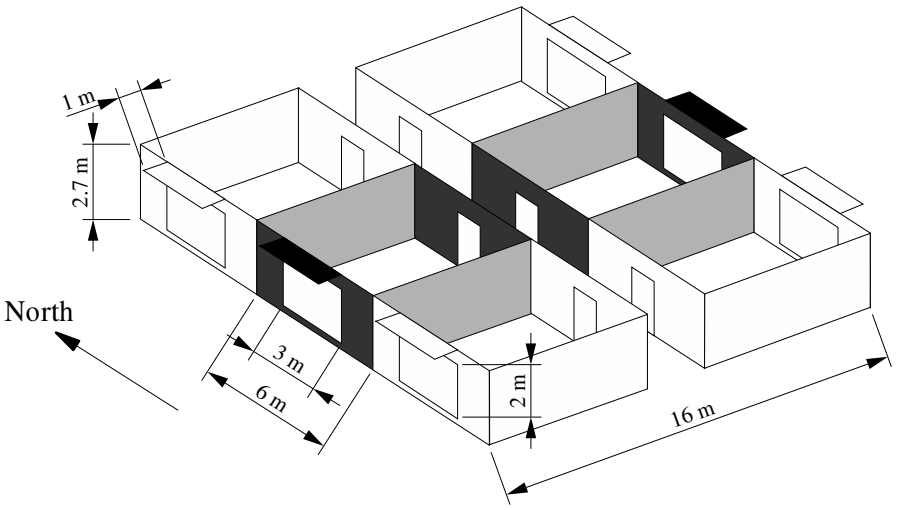
[2]Because $z(\cdot, 1)$ and hence $f(\cdot)$ is discontinuous, $f(\cdot)$ may only have an infimum (i.e., a greatest lower bound) but no minimum even if $\mathbf{X}$ is compact. Thus, to be correct, (4.2.1a) should be replaced by $\inf_{x\in\mathbf{X}} f(x)$. For simplicity, we will not make this distinction.

from a solution or may at best converge much slower. Consequently, we are interested in how they perform compared to probabilistic population-based optimization algorithms on rather smooth cost functions and on cost functions with large discontinuities.

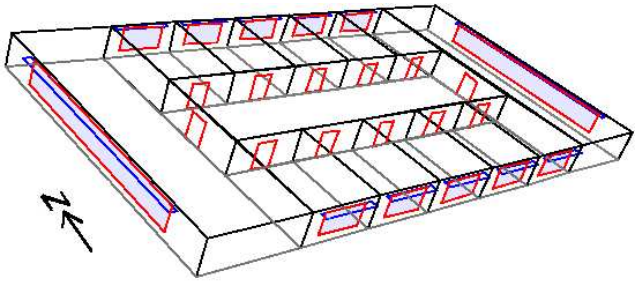
4.3 Simulation Models

We will use two simulation models. The first is a simple simulation model that has 4 independent variables and no HVAC system simulation (the zone's heating and cooling loads are assumed to be met at each time step). The second is a detailed simulation model that has 13 independent variables and a detailed HVAC system simulation. To determine the size of the HVAC system of the detailed simulation model, EnergyPlus executes a code that contains iterations. Thus, in the simple simulation model the component size $p(\cdot)$ in (4.2.3a) is a constant, but in the detailed simulation model $p(\cdot)$ is a discontinuous function of x.

In all optimization problems $f(x)$ is the annual primary energy consumption for lighting, fan, cooling and heating of a mid-story office floor. 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 cm wood siding, 10 cm insulation and 20 cm concrete. The ceiling and floor consist of carpet, 5 cm concrete, insulation and 18 cm concrete. Interior walls are 12 cm brick. The windows are low-emissivity double pane windows with Krypton gas fill and exterior shading device. We use TMY2 weather data for Houston Intercontinental (TX), Chicago O'Hare (IL),



(a) Simple office building.



(b) Detailed office building.

Figure 4.1: Buildings used in the numerical experiments.

and Seattle Tacoma (WA). Fig. 4.1 shows the office buildings.

4.3.1 Simple Simulation Model

The energy consumption of the gray shaded thermal zone in Fig. 4.1 is assumed to be representative for the energy consumption of an elongated office building. Both windows have an external shading device that is activated only during summer when the

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

The annual source energy consumption is

$$f(x) \triangleq \frac{Q_h(x)}{\eta_h} + \frac{Q_c(x)}{\eta_c} + 3\,E_l(x), \tag{4.3.1}$$

where $Q_h(\cdot)$ and $Q_c(\cdot)$ are the zone's annual heating and cooling load, respectively, $E_l(\cdot)$ is the zone's electricity consumption for lighting, and the efficiencies $\eta_h = 0.44$ and $\eta_c = 0.77$ are typical plant efficiencies that relate the zone load to the primary energy consumption for heating and cooling generation, including electricity consumption for fans and pumps (Huang and Franconi, 1999). The electricity consumption is multiplied by 3.0 to convert site electricity to source fuel energy consumption.

Tab. 4.1 lists the independent variables, which are the building azimuth α, the width of the west and east windows w_w and w_e, respectively, and the shading device transmittance τ.[3] The column with header x_b shows the values of the independent variables for the base design, l and u are the lower and upper bounds, and s is the step size of the independent variables. (The step size will be used in the optimization algorithms.)

[3] If $\alpha = 90°$, then the window that was initially facing west is facing north.

variable symbols	x_b	l	u	s	Houston, TX best iterate x^*	Chicago, IL best iterate x^*	Seattle, WA best iterate x^*
(a) *Optimization problems with simple simulation model and 4 independent variables.*							
α	0	-180	180	10	92.81	86.25	87.50
w_W	3	0.1	5.9	0.2	5.203	4.200	5.900
w_E	3	0.1	5.9	0.2	3.565	5.900	5.900
τ	0.5	0.2	0.8	0.1	0.7964	0.5875	0.5375
Cost at base design, $f(x_b)$ in kWh/(m^{2}a)					208.2	185.1	164.9
Cost at best iterate, $f(x^*)$ in kWh/(m^{2}a)					190.3	155.8	138.0
Maximum obtained reduction in %					8.58	15.82	16.32
(b) *Optimization problems with detailed simulation model and 13 independent variables.*							
w_N	0.5	0	1	0.05	0.9969	1.000	1.000
w_W	0.5	0	1	0.05	0.1813	0.4000	0.5688
o_W	0.5	0	1	0.05	1.000	0.3500	0.6688
w_E	0.5	0	1	0.05	0.2125	0.3000	0.8813
o_E	0.5	0	1	0.05	1.000	0.4500	0.9656
w_S	0.5	0	1	0.05	0.6406	0.9500	1.000
o_S	0.5	0	1	0.05	1.000	0.1000	0.1438
s_W	200	100	600	25	398.4	400.0	312.5
s_E	200	100	600	25	406.3	450.0	200.0
s_S	200	100	600	25	375.0	575.0	600.0
T_u	22	20	25	0.25	24.61	24.00	24.00
T_i	22	20	25	0.25	22.98	24.75	24.95
T_d	15	12	18	0.25	12.00	12.00	12.00
Cost at base design, $f(x_b)$ in kWh/(m^{2}a)					165.4	130.1	114.6
Cost at best iterate, $f(x^*)$ in kWh/(m^{2}a)					141.5	115.7	95.82
Maximum obtained reduction in %					14.45	11.02	16.41

Table 4.1: Variable symbols, initial value x_b, lower bound l, upper bound u and step size s of the independent variable. The variable symbols are explained in the text. The last three columns show the best obtained iterates x^*. The bottom rows show the corresponding cost function values and the obtained cost reductions for each optimization problem.

4.3.2 Detailed Simulation Model

We minimize the annual primary energy consumption for lighting, fan, cooling and heating for the mid-story office floor shown in Fig. 4.1. Lighting and fan electricity are

multiplied by 3.0 and then added to the cooling and heating energy of the cooling and heating coil. All exterior zones have daylighting control. The simulated HVAC system is a VAV system with DX coil and outside-air economizer. The heating and cooling coil capacities and the air flow rates are auto-sized by EnergyPlus. Tab. 4.1 lists the independent variables. The variable w_i, $i \in \{N,W,E,S\}$, linearly scales the window width and height. The subscripts indicate north, west, east, and south, respectively. (The location and shape of the windows are used in the daylighting calculations.) For the north and south windows a value of 0 corresponds to a window that covers 13.6% of the facade area and 1 corresponds to 64.8%. For the west and east windows a value of 0 corresponds to a window that covers 20.4% of the facade area and 1 corresponds to 71.3%. The variable o_i, $i \in \{W,E,S\}$, scales the depth of the window overhangs. A value of 0 corresponds to a window overhang depth of 0.05 m (measured from the facade) and 1 corresponds to 1.05 m. The variable s_i, $i \in \{W,E,S\}$, is the setpoint for the shading device in $\mathrm{W/m^2}$. If the total solar irradiation on the window exceeds s_i, then an external shading device with a transmittance of 0.5 is activated. The variable T_i, $i \in \{u,i\}$, is the setpoint for the zone air temperature for night cooling during summer and winter, respectively, in °C. The variable T_d is the cooling design supply air temperature that is used for the HVAC system sizing in °C.

4.4 Optimization Algorithms

We compare the performance of nine optimization algorithms. In the following section, we briefly describe the main features of all algorithms. For a more detailed description we refer the reader to Wetter and Wright (2003a) for the simple Genetic Algorithm and to the GenOpt manual (Wetter, 2004) for all other algorithms, as well as to the references cited therein. Since the performance of the optimization algorithms depends on the algorithm parameters, we list all algorithm parameters, which may be used as initial choices for similar problems. We did not tune the algorithm parameters but used values which we believe will give good performance for the examined problems. A detailed explanation of all parameters is beyond the scope of this Chapter, and we refer the reader to Wetter and Wright (2003a) and to the GenOpt manual for details.

We will now describe the optimization algorithms used in the numerical experiments.

4.4.1 Coordinate Search Algorithm

The Coordinate Search algorithm searches along each coordinate direction for a decrease in $f(\cdot)$. Let $k \in \mathbb{N}$ be the iteration number, $x_k \in \mathbf{X}$ be the current iterate, $\Delta_k \in \mathbb{Q}_+$ be a scaling factor, called the *mesh size factor*, and $s \in \mathbb{R}^n$ be as in Tab. 4.1. Then, our Coordinate Search algorithm tests if $f(x') < f(x_k)$ for any $x' \in \mathcal{L}_k$, where

$$\mathcal{L}_k \triangleq \left\{ x \in \mathbf{X} \mid x = x_k \pm \Delta_k s^i e_i, i \in \{1, \ldots, n\} \right\}. \tag{4.4.1}$$

If there exists an $x' \in \mathcal{L}_k$ that satisfies $f(x') < f(x_k)$, then the algorithm sets $x_{k+1} = x'$, $\Delta_{k+1} = \Delta_k$, and it replaces k by $k+1$. Otherwise, it sets $x_{k+1} = x_k$, decreases the mesh size factor by setting $\Delta_{k+1} = \Delta_k/2$, and it replaces k by $k+1$. If Δ_k is smaller than a user-specified limit, the search stops.

The Coordinate Search algorithm is a member of the family of Generalized Pattern Search (GPS) algorithms. For $\mathbf{X} = \mathbb{R}^n$, one can prove that any GPS method constructs a sequence of iterates with stationary accumulation points if $f(\cdot)$ is continuously differentiable and has bounded level sets (Audet and Dennis, 2003; Torczon, 1997). For a more detailed description of GPS algorithms and an extension to constraint problems, see for example Chapter 3, the paper by Audet and Dennis (2003) and the review by Kolda et al. (2003).

For the numerical experiments, we used a mesh size divider of 2, an initial mesh size exponent of 0, a mesh size exponent increment of 1 and 4 step reductions. Hence, $\Delta_0 = 1$ and, for the last iterations, $\Delta_k = 1/16$. Thus, the best iterate x^* satisfies $f(x^*) \leq f(x')$, for all $x' \in \{x \in \mathbf{X} \mid x = x^* \pm 1/16\, s^i e_i,\ i \in \{1,\ldots,n\}\}$.

4.4.2 Hooke-Jeeves Algorithm

The Hooke-Jeeves algorithm is also a member of the family of GPS algorithms and has therefore the same convergence properties on once continuously differentiable cost functions as the Coordinate Search algorithm. It adjusts the mesh size factor Δ_k using the same algorithm as the Coordinate Search algorithm but, in addition to the search on

$\mathcal{L}_k$, it also makes progressively bigger steps in the direction that has reduced the cost in previous iterations. As in the Coordinate Search algorithm, the iterates of the Hooke-Jeeves algorithm belong to a mesh of the form

$$\mathbb{M}(x_0, \Delta_k, s) \triangleq \left\{ x_0 + \Delta_k \sum_{i=1}^{n} m^i s^i e_i \mid m \in \mathbb{Z}^n \right\}. \tag{4.4.2}$$

The introduction of $\mathbb{M}(\cdot,\cdot,\cdot)$ is convenient for the discussion of a modified Particle Swarm Optimization algorithm and a hybrid algorithm below.

We used the same algorithm parameters for the Hooke-Jeeves algorithm as for the Coordinate Search algorithm.

4.4.3 Particle Swarm Optimization Algorithms

Particle Swarm Optimization (PSO) algorithms are population-based probabilistic optimization algorithms first proposed by Eberhart and Kennedy (Eberhart and Kennedy, 1995; Kennedy and Eberhart, 1995). At each iteration step, they compare the cost function value of a finite set of points, called *particles*. The change of each particle from one iteration to the next is modeled based on the social behavior of flocks of birds or schools of fish. Each particle attempts to change its location in $\mathbf{X}$ to a point where it had a lower cost function value at previous iterations, which models cognitive behavior, and in a direction where other particles had a lower cost function value, which models social behavior. Since our simulation model is computationally expensive, we

run the PSO algorithms with a much lower number of simulations than the ones in van den Bergh and Engelbrecht (2001), Kennedy et al. (2001) and Parsopoulos and Vrahatis (2002), which makes convergence to a minimum less likely.

We used a PSO algorithm with inertia weight and a PSO algorithm with constriction coefficient. Both algorithms used the von Neumann Topology, 16 particles, 20 generations, a seed of 0, a cognitive acceleration constant of 2.8, a social acceleration constant of 1.3 and velocity clamping with a maximum velocity gain of 0.5. For the PSO algorithm with inertia weight, we used an initial inertia of 1.2 and a final inertia of 0. For the PSO algorithm with constriction coefficient, we used a constriction gain of 0.5.

4.4.4 Particle Swarm Optimization Algorithm that Searches on a Mesh

This is a modification of the above PSO algorithm with constriction coefficient which is introduced in Wetter (2004). In this algorithm, the cost function $f\colon \mathbb{R}^n \to \mathbb{R}$ is replaced by the function $\widehat{f}\colon \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{Q}_+ \times \mathbb{R}^n \to \mathbb{R}$, defined as

$$\widehat{f}(x; x_0, \Delta, s) \triangleq f\big(\gamma(x)\big), \tag{4.4.3}$$

where $\gamma(x) \in \mathbb{M}(x_0, \Delta, s) \cap \mathbf{X}$ is the closest feasible mesh point and $\mathbb{M}(\cdot,\cdot,\cdot)$ is as in (4.4.2). Evaluating the cost function on the mesh reduces the number of simulations when the particles cluster.

We run this algorithm with two different settings for the algorithm parameters. In the first version, which we will call *PSO on mesh (1)*, we used the same parameters as for the PSO algorithm with constriction coefficient and, in addition, a mesh size divider of 2 and an initial mesh size exponent of 1. Thus, $\Delta = 1/2$ in (4.4.3). In the second version, which we will call *PSO on mesh (2)*, we increased the number of particles from 16 to 36 and increased the constriction gain from 0.5 to 1. This causes the particles to cluster later in the search.

4.4.5 Hybrid Particle Swarm and Hooke-Jeeves Algorithm

This hybrid global optimization algorithm does a Particle Swarm Optimization on a mesh for the first iterations, as described in the previous section. This is done for a user-specified number of generations. Then, it starts the Hooke-Jeeves algorithm using for the initial iterate the mesh point that attained the lowest cost function value. Since the PSO algorithm evaluates $f(\cdot)$ only on a finite number of points in $\mathbb{M}(x_0, \Delta_0, s) \cap \mathbf{X}$, the PSO search can be considered to be a global search of a GPS algorithm. Hence, this hybrid algorithm is also a member of the family of GPS algorithms.

We run this algorithm with two different settings for the algorithm parameters. In the first version, which we will call *PSO and Hooke-Jeeves (1)*, we used the same settings as for the algorithm *PSO on mesh (1)*. In addition, we used, as for the Hooke-Jeeves and the Coordinate Search algorithms, a mesh size exponent increment of 1. Because $\Delta_0 = 1/2$, we used 3 step reductions to obtain for the last iterations the same mesh size

factor as for the Hooke-Jeeves and the Coordinate Search algorithms, namely $\Delta_k = 1/16$. In the second version, which we will call *PSO and Hooke-Jeeves (2)*, we increased the constriction gain from 0.5 to 1 to obtain a bigger spread in the particles for the late generations, but we kept the number of particles at 16.

4.4.6 Simple Genetic Algorithm

Genetic Algorithms (GA) are algorithms that operate on a finite set of points, called a *population*. The different populations are called *generations*. They are derived on the principles of natural selection and incorporate operators for (1) fitness assignment, (2) selection of points for recombination, (3) recombination of points, and (4) mutation of a point. Our GA is an implementation of the *simple GA* described by Goldberg (1989), but we use a Gray (Press et al., 1993) rather than a pure binary encoding to represent the independent variables as a concatenated string of binary numbers.

The simple GA iterates either until a user-specified number of generations is exceeded, or until all iterates of the current generation have the same cost function value.

In the numerical experiments, we used a population size of 14, a maximum of 50 generations, 1 elite point and a probability for recombination and mutation of 1 and 0.02, respectively.

We selected a small population size because the number of independent variables is small and because we expected the cost function to have no significant local minima. The choice of a small population size was balanced by a high probability of recombination

and mutation. Small population sizes have also been used successfully in the solution of other small scale building optimization problems, see Caldas and Norford (2002).

4.4.7 Simplex Algorithm of Nelder and Mead

The Simplex algorithm of Nelder and Mead is a derivative free optimization algorithm. It constructs an n-dimensional simplex in the space of the independent variables. The cost function is evaluated at each of the $(n+1)$ simplex vertices. In each iteration step, the vertex with the highest cost function value is replaced by a new vertex. The new vertex is obtained either by reflecting the vertex with the highest cost function value at the centroid of the simplex, or by contracting and expanding the simplex. The Nelder and Mead Simplex algorithm is an often used algorithm despite the well known fact that it can fail to converge to a stationary point (see for example Kelley (1999b), Torczon (1989), Kelley (1999a), Wright (1996), McKinnon (1998) and Lagarias et al. (1998)), both in practice and theory, particularly if the dimension of independent variables is large, say bigger than ten (Torczon, 1989). Several improvements to the Simplex algorithm or algorithms that were motivated by the Simplex algorithm exist, see for example Kelley (1999a,b), Torczon (1989) and Tseng (1999). However, here we used the original Nelder-Mead algorithm, as described in Nelder and Mead (1965) with the extension of O'Neill (1971) and, in some of the numerical experiments, a modification of the stopping criteria, as described in the GenOpt manual.

For the numerical experiments, we used an accuracy of 0.01 and a step-size factor

of 0.1. In the experiments that are labeled *Nelder-Mead (1)*, we modified the stopping criterion as described in the GenOpt manual and prevented a new restart of the algorithm during the ten iterations that followed a previous restart. In the experiments that are labeled *Nelder-Mead (2)*, we did not modify the stopping criterion and did not prevent a restart of the algorithm. The second set of numerical experiments has been done because the first set showed poor performance. However, the change in algorithm parameters did not improve the performance.

4.4.8 Discrete Armijo Gradient Algorithm

We used the Discrete Armijo Gradient algorithm which is designed for the minimization of once continuously differentiable functions. The algorithm is described in Polak (1997) and in the GenOpt manual. It approximates gradients by finite differences, with the difference increment reduced as the optimization progresses, and does line searches using the Armijo step-size rule. If $f(\cdot)$ is once continuously differentiable and bounded from below, then the Discrete Armijo Gradient algorithm constructs sequences with stationary accumulation points. However, the algorithm is sensitive to discontinuities in $f(\cdot)$, and hence, we recommend to not use this algorithm if the simulation program contains adaptive solvers with loose precision settings, such as EnergyPlus. However, since the simple simulation model defines a cost function that has only small discontinuities, we were interested in how this algorithm performs in solving the problems that use the simple simulation model. As we will shortly see, it failed far from a solution.

We used the following algorithm parameters: $\alpha = 1/2$, $\beta = 0.8$, $\gamma = 0.1$, $k_0 = 0$, $k^* = -10$, $l_{max} = 50$, $\kappa = 25$, $\varepsilon_m = 0.01$ and $\varepsilon_x = 0.05$.

4.5 Numerical Experiments

The analysis presented here is in two parts. We will first compare the performance of the different optimization algorithms and then examine the cause of the discontinuities in the cost functions.

4.5.1 Comparison of the Optimization Results

For all optimizations, we used the optimization program GenOpt 2.0.0 (Wetter, 2001, 2004)[4] and EnergyPlus 1.1.0. All computations were done on Linux computers using the 2.4.20-8 kernel and AMD processors. On a 2.2 GHz processor, one simulation of the simple model takes 14 seconds, and one simulation of the detailed model takes 2 minutes and 20 seconds. Thus, 300 simulations of the simple model (which is what most optimization algorithms used in our experiments) takes 1 hour and 10 minutes, and 500 simulations of the detailed simulation model takes 19 hours and 30 minutes. The overhead of the optimization algorithm and the file I/O is negligible.

We first need to define some measures that we will use to compare the optimization results. Let $x_b \in \mathbf{X}$ be the value of the independent variables for the base design, as listed

[4]The Genetic Algorithm is currently only implemented in a development version.

in Tab. 4.1. For each optimization problem (i.e., for each simulation model with the corresponding weather data), we denote, for each optimization algorithm, by $x^* \in \mathbf{X}$ the iterate with the lowest cost function value, and we denote for each optimization problem by $\widehat{x} \in \mathbf{X}$ the iterate with the lowest cost function value obtained by any of the tested optimization algorithms. Then, for each optimization problem, we define the normalized cost reduction as

$$r(x^*) \triangleq \frac{f(x_b) - f(x^*)}{f(x_b)}, \tag{4.5.1}$$

and we define the distance to the maximum obtained reduction as

$$d\big(r(x^*)\big) \triangleq r(\widehat{x}) - r(x^*) = \frac{f(x^*) - f(\widehat{x})}{f(x_b)}. \tag{4.5.2}$$

Thus, $d\big(r(x^*)\big) = 0$ for the algorithm that achieves the biggest cost reduction.

Because of the discontinuities in the cost functions, we observed different behavior of the optimization algorithms on the problems that used the simple simulation model compared to the problems that used the detailed simulation model. The cost function, if evaluated by the simple simulation model, is rather smooth, but, if evaluated by the detailed simulation model, has discontinuities in the order of 2%, which makes optimization with descent algorithms difficult.

Tab. 4.2 shows the normalized cost reduction, the distance to the maximum obtained reduction and the number of simulations, and Fig. 4.2 shows a graphical representation

	Houston, TX			Chicago, IL			Seattle, WA		
	$r(x^*)$	$d\big(r(x^*)\big)$	m	$r(x^*)$	$d\big(r(x^*)\big)$	m	$r(x^*)$	$d\big(r(x^*)\big)$	m
Algorithm	[%]	[%]	–	[%]	[%]	–	[%]	[%]	–
(a) *Optimization problems with simple simulation model and 4 independent variables.*									
PSOIW	8.44	0.14	316	15.42	0.41	316	16.11	0.21	314
PSOCC	8.27	0.31	313	14.49	1.34	314	14.77	1.55	315
PSOCC on a mesh (1)	8.27	0.31	169	14.57	1.25	160	14.89	1.43	146
PSOCC on a mesh (2)	8.47	0.12	672	15.75	0.07	673	16.23	0.09	652
Nelder-Mead (1)	8.58	**0**	259	15.71	0.11	672	16.18	0.14	1232
Nelder-Mead (2)	8.58	**0**	226	15.71	0.11	902	16.25	0.07	1451
PSO and Hooke-Jeeves (1)	8.58	0.01	242	15.82	**0**	237	16.30	0.02	215
PSO and Hooke-Jeeves (2)	8.56	0.02	326	15.74	0.08	371	16.32	**0**	358
Hooke-Jeeves	8.58	0.01	103	15.82	**0**	113	16.30	0.02	97
Coordinate Search	8.58	0.01	105	15.82	**0**	119	16.30	0.02	116
Simple GA	8.53	0.05	194	15.52	0.31	185	16.23	0.09	176
Discrete Armijo Gradient	7.93	0.66	315	13.08	2.75	364	14.95	1.37	216
(b) *Optimization problems with detailed simulation model and 13 independent variables.*									
PSOIW	13.91	0.54	317	10.63	0.39	317	15.39	1.03	318
PSOCC	11.97	2.49	313	9.66	1.37	314	14.18	2.23	317
PSOCC on a mesh (1)	12.17	2.28	195	9.68	1.34	209	14.20	2.22	242
PSOCC on a mesh (2)	13.49	0.96	710	10.39	0.63	712	15.84	0.57	707
Nelder-Mead (1)	14.09	0.36	2330	4.27	6.76	1228	15.59	0.82	5846
Nelder-Mead (2)	13.98	0.48	1578	–	–	–	–	–	–
PSO and Hooke-Jeeves (1)	14.16	0.29	653	10.94	0.09	755	16.18	0.23	843
PSO and Hooke-Jeeves (2)	14.45	**0**	740	10.96	0.06	669	16.41	**0**	889
Hooke-Jeeves	14.27	0.18	555	5.93	5.10	600	16.32	0.09	574
Coordinate Search	9.60	4.86	430	4.74	6.29	552	13.04	3.37	501
Simple GA	14.06	0.40	586	11.02	**0**	592	16.35	0.07	583

Table 4.2: Normalized cost reduction $r(x^*)$, distance to the maximum obtained cost reduction $d\big(r(x^*)\big)$ and number of simulations m for all optimization problems.

of the distance to the maximum obtained cost reduction and the required number of simulations for each optimization problem.

We see that if the detailed simulation model is used, then the Coordinate Search algorithm tends to fail far from the minimum. On the same problems, the Hooke-Jeeves algorithm jammed less often compared to the Coordinate Search algorithm, which may be due to the larger steps that are taken in the global exploration.

All non-hybrid PSO algorithms come close to the minimum with a low number of

simulations. By restricting the iterates of the PSO algorithm with constriction coefficient to the mesh $\mathbb{M}(x_0, \Delta, s)$, the number of simulations could be reduced by 50% for the problem with 4 independent variables and by 30% for the problem with 13 independent variables. Restricting the iterates to the mesh does not significantly affect the accuracy as we can see by comparing the results of the *PSOCC* and the *PSO on mesh (1)* algorithm, which both use the same algorithm parameters. However, in the PSO algorithm that searches on a mesh, increasing the number of particles from 16 to 36 and increasing the constriction gain from 0.5 to 1 prevented the particles to cluster early in the search and yielded larger cost reductions at the expense of three to four times more simulations.

The simple GA, however, got consistently closer to the minimum than the PSO algorithms with a comparable number of simulations, except for one problem that used the simple simulation model. In this case, however, the difference in cost reduction is insignificant.

The overall best cost reductions has been achieved by the hybrid Particle Swarm and Hooke-Jeeves algorithm although with a higher number of simulations than the simple GA. For the hybrid algorithm, increasing the constriction gain from 0.5 to 1.0 did, in our experiments, only slightly change the results. We observed, however, that with a higher constriction gain the particles are more spread out in the early generations, which increases the chance to find a global minimum if the cost function has several minima.

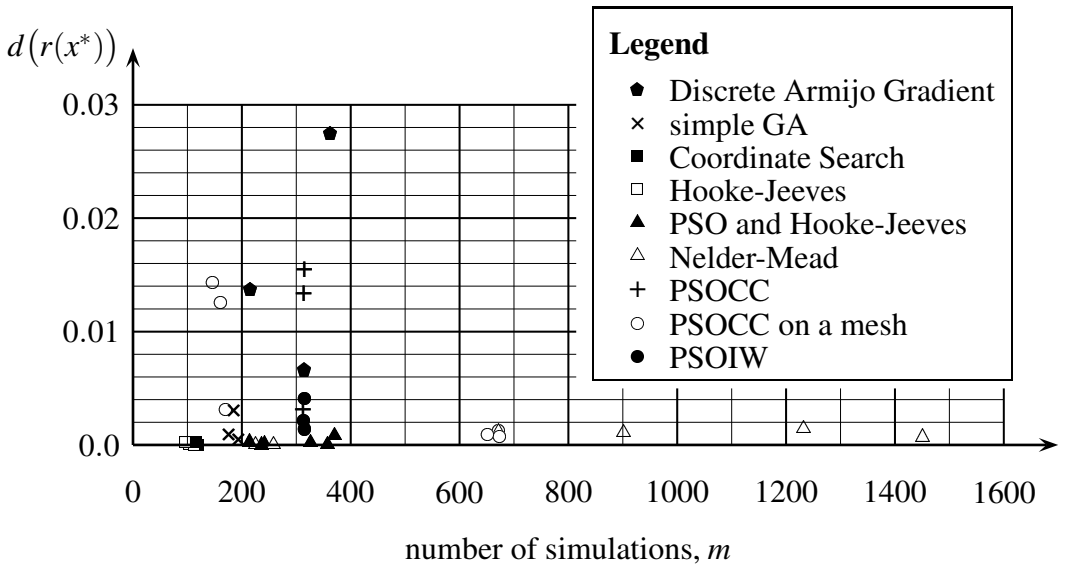
The Nelder-Mead algorithm did not perform well on our test problems. It required a high number of simulations, and in one test case it failed far from the minimum. We

believe that, in addition to the problems discussed in the literature (Kelley, 1999a,b; Lagarias et al., 1998; McKinnon, 1998; Torczon, 1989; Wright, 1996), some of the problems we observed when solving the optimization problems that used the detailed simulation model may have been caused by the stopping criterion. The stopping criterion described by Nelder and Mead (1965), which is the one implemented in GenOpt, requires the variance of the function values at the simplex vertices to be smaller than a prescribed limit. However, if $f(\cdot)$ has large discontinuities, then this stopping criterion may never be satisfied.

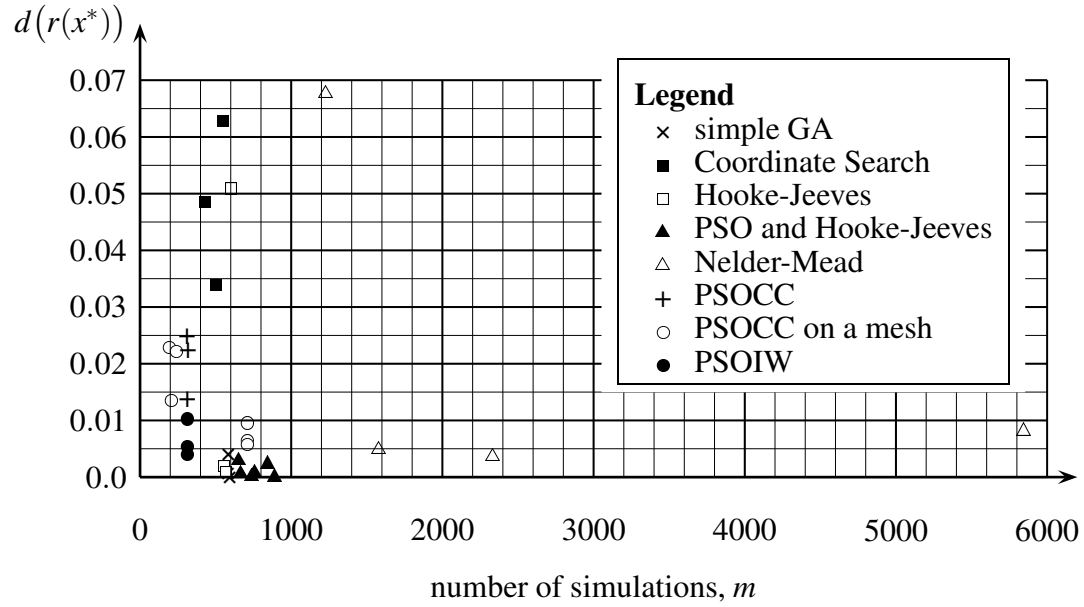
The Discrete Armijo Gradient algorithm failed on the simple problem far from the minimum. This is not surprising because the algorithm is sensitive to discontinuities in the cost function. We recommend to not use this algorithm if EnergyPlus is used to evaluate the cost function.

In summary, the simple GA got close to a solution with a low number of simulations. The hybrid Particle Swarm and Hooke-Jeeves algorithm achieved the biggest cost reduction but required more simulations. Whether the increased number of simulations is justified depends on the savings due to a better solution and the expenses of a higher computation time. However, one advantage of the hybrid algorithm is that the global search of the PSO algorithm increases the chance to get close to the global minimum rather than only a local minimum, and the Hooke-Jeeves algorithm then refines the search locally.

If the discontinuities in the cost function are small, then the Hooke-Jeeves algorithm achieved a good reduction in cost and required only few iterations.



(a) Optimization problems with simple simulation model and 4 independent variables.



(b) Optimization problems with detailed simulation model and 13 independent variables.

Figure 4.2: Number of simulations vs. distance to the maximum obtained cost reduction.

4.5.2 Discontinuities in the Cost Function

We observed that EnergyPlus computes for the detailed simulation model an energy consumption that has large discontinuities. Some of the discontinuities are caused by the adaptive grid generators – such as the ones used in computing the daylighting illuminance or in doing the variable time-step integration – some are caused by iterative solvers which fail to compute an accurate approximate solution and some are caused by programming errors.

Due to these discontinuities, for the optimization problem with the detailed simulation model and Chicago's weather data, the Hooke-Jeeves algorithm achieved only half of the cost reduction that was obtained by other algorithms. For this numerical experiment, we will now show the change in cost in a one-dimensional subspace of $\mathbf{X} \subset \mathbb{R}^{13}$. In particular, we will perturb one component of the independent variable and plot the change in cost function value. We will first introduce some notation. Let $x^*_{HJ} \in \mathbf{X} \subset \mathbb{R}^{13}$ denote the iterate with the lowest cost function value of the Hooke-Jeeves algorithm, let $T_u \in \mathbb{R}$ denote the room setpoint temperature for night cooling during summer, and let $e_{T_u} \in \mathbb{R}^{13}$ denote the coordinate vector along T_u. We define $T^*_{u,HJ} \triangleq \langle x^*_{HJ}, e_{T_u} \rangle$ and we define the normalized change in cost with respect to T_u as

$$\delta(T_u) \triangleq \frac{f\left(x^*_{HJ} + (T_u - T^*_{u,HJ})\, e_{T_u}\right)}{f(x^*_{HJ})}. \tag{4.5.3}$$

In Fig. 4.3 we show $\delta(T_u)$ for $T_u \in [21.9,\ 22.1]$ using 1201 equidistant support points.

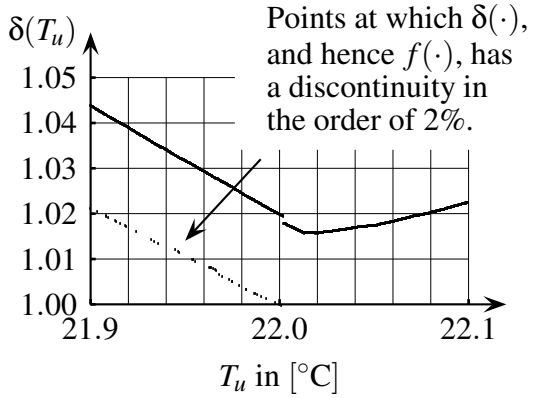


Figure 4.3: Normalized change of the cost function value $\delta(T_u)$ as a function of the zone air setpoint temperature for night cooling during the summer months.

For $T_u \leq 22.001°\text{C}$, $\delta(\cdot)$ has discontinuities in the order of 2%. At $T_u = 22°\text{C}$ is the discontinuity at which the Hooke-Jeeves algorithm got stuck, which is far from the minimum of $f(\cdot)$. The point-wise discontinuities in Fig. 4.3 are caused by round-off errors: Depending on the system's minimum air-flow fraction, which depends on the system sizing $p(\cdot)$, a different branch of an `if-then-else` statement is executed to determine the part load air-flow fraction, and the two branches of the `if-then-else` statement are, due to a programming error, such that they introduce a discontinuity in the part load air-flow fraction.[5]

We will now show that the cost function of the detailed simulation model also has large discontinuities that are of a different structure than those in Fig. 4.3. Let $x^*_{CS} \in \mathbb{R}^n$ denote the iterate with the lowest cost function value obtained by the Coordinate Search algorithm for Chicago, and let x^*_{GA} denote the iterate with the lowest cost function value obtained by the simple GA for Chicago. In Fig. 4.4, we show the normalized energy

[5]This will be fixed in future EnergyPlus versions.

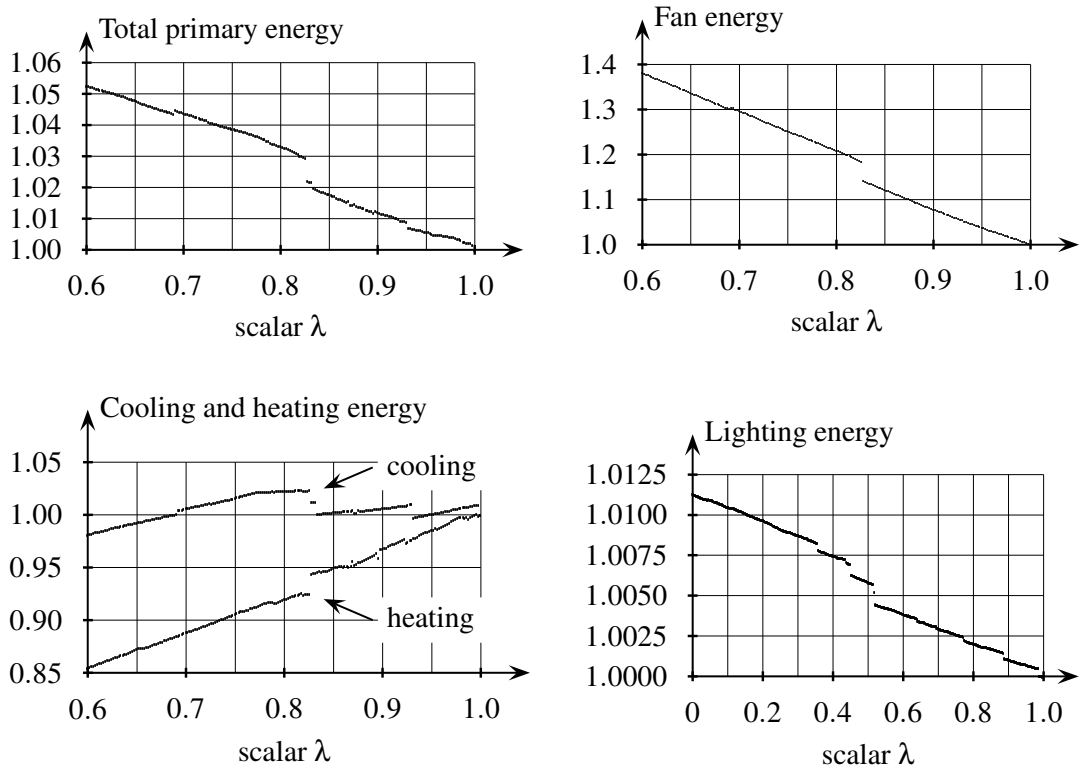


Figure 4.4: Normalized cost function value $f(x(\lambda))/f(x(1))$ and primary energy consumption for fan, cooling, heating and lighting, normalized by dividing it by the value at the minimum point of the simple GA. The functions are evaluated on the line between the minimum point obtained by the Coordinate Search algorithm (at $\lambda = 0$) and the minimum point obtained by the simple GA (at $\lambda = 1$) for Chicago, IL.

consumption for total primary energy, fan, cooling, heating and lighting energy. The normalized energy consumption is shown along part of the line $x(\lambda) \triangleq x^*_{CS} + \lambda \left(x^*_{GA} - x^*_{CS}\right)$. That is, $x(0) = x^*_{CS}$ and $x(1) = x^*_{GA}$. The graph shows discontinuities of the total primary energy consumption in the order of 1%. Clearly, such large discontinuities can cause optimization algorithms that require smoothness of the cost function to fail far from a minimum.

The biggest discontinuities seem to be caused by the fan sizing. At $\lambda = 0.827$, the

fan energy changes by 4% and consequently the cooling and heating energy is also discontinuous at this point. However, at $\lambda = 0.931$, the cooling energy changes by 1.2% while the fan energy is smooth around this point. While such a discontinuity is small if one is only interested in the cooling sizing in a simulation study, it can cause problems in optimization and sensitivity studies, particularly if the cooling energy contributes much to the cost function. The discontinuities in lighting energy are small (the lighting energy does not depend on the fan, cooling or heating energy). We believe that the discontinuities in the lighting energy are caused by a change in the spatial discretization that is used in computing the daylight illuminance. We further believe that the large discontinuities in the fan, heating and cooling energy are caused by the system auto-sizing, which we expect to compute a low-precision approximate solution to $p(x)$, which will then be used in (4.2.3) for the whole interval of time integration. In fact, the system sizing is done by first repetitively simulating a so-called warm-up day until some state variables do not change more than a tolerance which is fixed at compile time, and then one day is simulated to determine $p(\cdot)$. This is done for a winter and a summer day. Thus, a change in x can cause a discrete change in the number of warm-up days, and hence in the initial state $z_0(x)$ and consequently in the system size $p(x)$.

4.6 Conclusion

We observed that in the optimization problems that used the detailed simulation model with auto-sizing of the HVAC components, the cost function has discontinuities in the order of 2%. On such problems optimization algorithms that require smoothness of the cost function are likely to fail far from a solution, which is what we indeed observed in our numerical experiments. Such discontinuities make optimization difficult and can lead to limited economic gains. This can be prevented if the solvers are implemented in such a way that the approximation error can be controlled, and if the simulation program is written in such a way that the approximate solutions of the differential algebraic equations converge to a smooth function as the precision of the numerical solvers is increased. We developed a building energy simulation program that allows such an error control. It is presented in Chapter 2.

The biggest cost reduction has been obtained with the hybrid Particle Swarm and Hooke-Jeeves algorithm. If a user is willing to accept a slight decrease in accuracy at the benefit of fewer simulations, then the simple GA is a good choice. However, due to the stochastic operators, the PSO and the simple GA can occasionally fail to get close to a solution, particularly if the number of simulations is small. In such situations, the second search in a hybrid algorithm can further decrease the cost. However, with our limited number of numerical experiments, we could not determine how big the risk of failing is.

For neither of the problems that we examined do we recommend using either the Nelder-Mead or the Discrete Armijo Gradient algorithm.

Bibliography

Abramson, M. A. (2002). *Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems*. PhD thesis, Rice University, Houston, TX.

Al-Homoud, M. S. (1997). Optimum thermal design of office buildings. *International Journal of Energy Research*, 21:941–957.

Arasteh, D. K., Finlayson, E. U., and Huizenga, C. (1994). WINDOW 4.1: Program description. Technical Report LBL-35298, Lawrence Berkeley National Laboratory, Berkeley, CA, USA.

Arasteh, D. K., Reilly, M. S., and Rubin, M. D. (1989). A versatile procedure for calculating heat transfer through windows. In *ASHRAE Transactions*, volume 95.2, pages 755–765.

ASHRAE (1989). *Fundamentals*. American Society of Heating, Refrigeration and Air-Conditioning Engineers.

ASHRAE (2001). ANSI/ASHRAE Standard 140-2001, Standard method of test for the evaluation of building energy analysis computer programs.

Audet, C. and Dennis, Jr., J. E. (2000a). Pattern search algorithms for mixed variable programming. *SIAM Journal on Optimization*, 11(3):573–594.

Audet, C. and Dennis, Jr., J. E. (2000b). A pattern search filter method for nonlinear programming without derivatives. Technical Report TR 00-09, Rice University, Houston, Department of Computational and Applied Mathematics.

Audet, C. and Dennis, Jr., J. E. (2003). Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903.

Baum, D. R., Rushmeier, H. E., and Winget, J. M. (1989). Improving radiosity solutions through the use of analytically determined form-factors. *Computer Graphics*, 23(3):325–334.

Berdahl, P. and Fromberg, R. (1982). The thermal radiance of clear skies. *Solar Energy*, 29:299–314.

Berdahl, P. and Martin, M. (1984). Emissivity of clear skies. *Solar Energy*, 32:663–664.

Bertsekas, D. P. (1999). *Nonlinear Programming*. Athena Scientific, Belmont, MA, USA, 2nd edition.

Björsell, N., Bring, A., Eriksson, L., Grozman, P., Lindgren, M., Sahlin, P., Shapovalov, A., and Vuolle, M. (1999). IDA indoor climate and energy. In Nakahara, N., Yoshida,

H., Udagawa, M., and Hensen, J., editors, *Proc. of the 6-th IBPSA Conference*, pages 1035–1042, Kyoto, Japan.

BLAST (1999). *BLAST 3.0 Users Manual*. University of Illinois, Urbana-Champaign, IL, Department of Mechanical and Industrial Engineering, Building Systems Laboratory.

Booker, A., Dennis, Jr., J. E., Frank, P., Serafini, D., and Torczon, V. (1998). Optimization using surrogate objectives on a helicopter test example. In Borggaard, J. T., Burns, J., Cliff, E., and Schreck, S., editors, *Computational Methods in Optimal Design and Control*, volume 24 of *Progress in Systems and Control Theory*, pages 49–58. Birkhäuser, Boston.

Booker, A. J., Dennis, Jr., J. E., Frank, P. D., Serafini, D. B., Torczon, V., and Trosset, M. W. (1999). A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 17(1):1–13.

Brenan, K. E., Campbell, S. L., and Petzold, L. R. (1989). *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. North-Holland.

Brown, P. N., Hindmarsh, A. C., and Petzold, L. R. (1994). Using Krylov methods in the solution of large-scale differential-algebraic systems. *SIAM Journal on Scientific Computing*, 15:1467–1488.

Brown, P. N., Hindmarsh, A. C., and Petzold, L. R. (1998). Consistent initial condition

calculation for differential-algebraic systems. *SIAM Journal on Scientific Computing*, 19(5):1495–1512.

Bryan, H. J. and Clear, R. D. (1981). Calculating interior daylight illumination with a programmable hand calculator. *Journal of the Illuminating Engineering Society*, 10(4):219–227.

Caldas, L. G. and Norford, L. K. (2002). A design optimization tool based on a genetic algorithm. *Automation in Construction*, 11(2):173–184.

Carroll, W. L. (2003). Personal communication.

Choi, T. D. and Kelley, C. T. (2000). Superlinear convergence and implicit filtering. *SIAM Journal on Optimization*, 10(4):1149–1162.

Choudhary, R. (2004). *A Hierarchical Optimization Framework for Simulation-Based Architectural Design*. PhD thesis, University of Michigan.

Choudhary, R., Malkawi, A., and Papalambros, P. Y. (2003). A hierarchical design optimization framework for building performance analysis. In Augenbroe, G. and Hensen, J., editors, *Proc. of the 8-th IBPSA Conference*, Eindhoven, NL.

Clarke, F. H. (1990). *Optimization and nonsmooth analysis*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

Clarke, J. A. (2001). *Energy Simulation in Building Design*. Butterworth-Heinemann, Oxford, UK, 2nd edition.

Coddington, E. A. and Levinson, N. (1955). *Theory of ordinary differential equations*. McGraw-Hill Book Company, Inc., New York-Toronto-London.

Crawley, D. B., Lawrie, L. K., Winkelmann, F. C., Buhl, W. F., Huang, Y. J., Pedersen, C. O., Strand, R. K., Liesen, R. J., Fisher, D. E., Witte, M. J., and Glazer, J. (2001). EnergyPlus: creating a new-generation building energy simulation program. *Energy and Buildings*, 33(4):443–457.

Cuzzillo, B. R. and Pagni, P. J. (1998). Thermal breakage of double-pane glazing by fire. *J. of Fire Prot. Engr.*, 9(1):1–11.

Dennis, Jr., J. E. and Torczon, V. (1991). Direct search methods on parallel machines. *SIAM Journal on Optimization*, 1(4):448–474.

Dennis, Jr., J. E. and Torczon, V. (1997). Managing approximation models in optimization. In Alexandrov, N. M. and Hussaini, M. Y., editors, *Multidisciplinary Design Optimization: State of the Art*, ICASE/NASA Langley Workshop on Multidisciplinary Optimization, pages 330–347. SIAM.

Durand, F., Drettakis, G., and Puech, C. (1999). Fast and accurate hierarchical radiosity using global visibility. *ACM Transactions on Graphics (TOG)*, 18(2):128–170.

Eberhart, R. C. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, Nagoya, Japan. IEEE.

Evans, L. C. (1998). *Partial differential equations*. American Mathematical Society.

Finkel, D. E. (2003). *DIRECT Optimization Algorithm User Guide*. North Carolina State University, Center for Research in Scientific Computation, Raleigh, NC.

Finlayson, E. U., Arasteh, D. K., Huizenga, C., Rubin, M. D., and Reilly, M. S. (1993). WINDOW 4.0: Documentation of calculation procedures. Technical Report LBL-33943, Lawrence Berkeley National Laboratory, Berkeley, CA, USA.

Fontoynont, M., Laforgue, P., Mitanchey, R., Aizlewood, M., Butt, J., Carroll, W., Hitchcock, R., Erhorn, H., Boer, J. D., Dirksmöller, M., Michel, L., Paule, B., Scartezzini, J. L., Bodart, M., and Roy, G. (1999). IEA SHC Task 21/ECBCS Annex 29, Daylight in buildings, Subtask C1: Validation of daylighting simulation programmes. Technical Report T21/C1-/FRA/99-11, International Energy Agency.

Gablonsky, J. M. and Kelley, C. T. (2001). A locally-biased form of the DIRECT algorithm. *Journal of Global Optimization*, 21(1):27–37.

Gill, P. E., Murray, W., and Wright, M. H. (1981). *Practical optimization*. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London.

Goldberg, D. (1989). *Genetic Algorithm in Search, Optimization, and Machine Learning*. Adisson-Wesley.

Greenhalgh, D. and Marshall, S. (2000). Convergence criteria for genetic algorithms. *SIAM Journal for Computation*, 30(1):269–282.

Holman, J. P. (1997). *Heat Transfer*. McGraw-Hill, 8 edition.

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

Hottel, H. C. and Sarofim, A. F. (1967). *Radiative Transfer*. McGraw-Hill.

Huang, J. and Franconi, E. (1999). Commercial heating and cooling loads component analysis. Technical Report LBL-37208, Lawrence Berkeley National Laboratory, EETD.

Ineichen, P., Perez, R., and Seals, R. (1987). The importance of correct albedo determination for adequately modeling energy received by tilted surfaces. *Solar Energy*, 39(4):301–305.

Judkoff, R. and Neymark, J. (1995). International energy simulation test (BESTEST) and diagnostic method. Technical report, National Renewable Energy Laboratory.

Kays, W. M. and Crawford, M. E. (1993). *Convective Heat and Mass Transfer*. Mechanical Engineering Series. McGraw-Hill, 3rd edition.

Kelley, C. T. (1999a). Detection and remediation of stagnation in the Nelder–Mead algorithm using a sufficient decrease condition. *SIAM Journal on Optimization*, 10(1):43–55.

Kelley, C. T. (1999b). *Iterative methods for optimization*. Frontiers in Applied Mathematics. SIAM.

Kennedy, J. and Eberhart, R. C. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume IV, pages 1942–1948, Perth, Australia.

Kennedy, J., Eberhart, R. C., and Shi, Y. (2001). *Swarm Intelligence*. Morgan Kaufmann Publishers.

Kim, H. M., Michelena, N. F., Papalambros, P. Y., and Jiang, T. (2003). Target cascading in optimal system design. *Transaction of ASME: Journal of Mechanical Design*, 125:481–489.

Klein, S. A., Duffie, J. A., and Beckman, W. A. (1976). TRNSYS – A transient simulation program. *ASHRAE Transactions*, 82(1):623–633.

Kolda, T. G., Lewis, R. M., and Torczon, V. (2003). Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482.

Kreider, J. F. and Rabl, A. (1994). *Heating and Cooling of Buildings – Design for Efficiency*. McGraw-Hill, Inc.

Krishnakumar, K. (1989). Micro-genetic algorithms for stationary and non-stationary function optimization. In Rodriguez, G., editor, *Intelligent Control and Adaptive Systems*, pages 289–296. The International Society for Optical Engineering.

Laforgue, P. (1997). IEA SHC Task 21/ECBCS Annex 29, Daylight in buildings, Sub-

task C1: Draft report of Genelux simulations and other software results. Technical Report T21/C1/97-10, International Energy Agency.

Lagarias, J. C., Reeds, J. A., Wright, M. H., and Wright, P. E. (1998). Convergence properties of the Nelder-Mead simplex method in low dimensions. *SIAM Journal on Optimization*, 9(1):112–147.

Lewis, R. M. and Torczon, V. (1999). Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, 9(4):1082–1099.

Lewis, R. M. and Torczon, V. (2000). Pattern search methods for linearly constrained minimization. *SIAM Journal on Optimization*, 10(3):917–941.

Marion, W. and Urban, K. (1995). *User's Manual for TMY2s Typical Meteorological Years*. National Renewable Energy Laboratory, NREL, Golden, Co., USA.

Marsden, A. L., Wang, M., Dennis, Jr., J. E., and Moin, P. (2004). Optimal aeroacoustic shape design using the surrogate management framework. *Optimization and Engineering (in press)*.

Martin, M. and Berdahl, P. (1984). Characteristics of infrared sky radiation in the United States. *Solar Energy*, 33:321–336.

McKinnon, K. I. M. (1998). Convergence of the Nelder-Mead simplex method to a nonstationary point. *SIAM Journal on Optimization*, 9(1):148–158.

Musser, D. R., Derge, G. J., and Saini, A. (2001). *STL tutorial and reference guide: C++ programming with the standard template library*. Addison-Wesley, 2nd edition.

NCDC (1981). *Typical Meteorological Year*. National Climatic Data Center, Asheville, North Carolina.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The computer journal*, 7(4):308–313.

O'Neill, R. (1971). Algorithm AS 47 – Function minimization using a simplex procedure. *Appl. Stat. 20*, 20:338–345.

Parsopoulos, K. E. and Vrahatis, M. N. (2002). Recent approaches to global optimization problems through Particle Swarm Optimization. *Natural Computing*, 1:235–306.

Perez, R. (1999). Fortran function `irrpz.f`. Emailed by R. Perez to F. C. Winkelmann on May 21, 1999.

Perez, R., Ineichen, P., Seals, R., Michalsky, J., and Stewart, R. (1990). Modeling daylight availability and irradiance components from direct and global irradiance. *Solar Energy*, 44(5):271–289.

Perez, R., Seals, R., Ineichen, P., Stewart, R., and Menicucci, D. (1987). A new simplified version of the Perez diffuse irradiance model for tilted surfaces. *Solar Energy*, 39(3):221–231.

Perttunen, C. D., Jones, D. R., and Stuckman, B. E. (1993). Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Application*, 79(1):157–181.

Polak, E. (1971). *Computational Methods in Optimization; a Unified Approach*, volume 77 of *Mathematics in Science and Engineering*. New York, Academic Press.

Polak, E. (1997). *Optimization, Algorithms and Consistent Approximations*, volume 124 of *Applied Mathematical Sciences*. Springer Verlag.

Polak, E. and Wetter, M. (2003). Generalized pattern search algorithms with adaptive precision function evaluations. Technical Report LBNL-52629, Lawrence Berkeley National Laboratory, Berkeley, CA.

Press, W. H., Flannery, B. P., Tuekolsky, S. A., and Vetterling, W. T. (1993). *Numerical Recipes in C: The Art of Scientific Computing*, chapter 20. Cambridge University Press.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical recipes in C*. Cambridge University Press, Cambridge.

Rabl, A. (1985). *Active Solar Collectors and Their Applications*. Oxford University Press.

Rubinstein, F. (1999). Dimming characteristics of three controllable ballasts. Lighting

Controls Technical Note 2, Lawrence Berkeley National Laboratory, Lighting Research Group, Berkeley, CA.

Sahlin, P. and Bring, A. (1991). IDA solver – A tool for building and energy systems simulation. In Clarke, J. A., Mitchell, J. W., and de Perre, R. C. V., editors, *Proc. of the IBPSA Conference*, Nice, France.

Sahlin, P. and Sowell, E. F. (1989). A neutral format for building simulation models. In *Proc. of the IBPSA Conference*, pages 147–154, Vancouver, Canada.

Sellers, W. D. (1965). *Physical Climatology*. The University of Chicago Press.

Serafini, D. B. (1998). *A Framework for Managing Models in Nonlinear Optimization of Computationally Expensive Functions*. PhD thesis, Rice University.

SPARK (2003). *SPARK, Reference Manual*. Lawrence Berkeley National Laboratory and Ayres Sowell Associates Inc., Berkeley, CA, USA.

Stepanov, A. and Lee, M. (1995). The standard template library. Technical report, Hewlett-Packard, Palo Alto, CA, USA.

Strang, G. and Fix, G. J. (1973). *An Analysis of the Finite Element Method*. Prentice-Hall, Inc.

Stroustrup, B. (2000). *The C++ Programming Language*. Addison Wesley.

Torczon, V. (1989). *Multi-Directional Search: A Direct Search Algorithm for Parallel Machines*. PhD thesis, Rice University, Houston, TX.

Torczon, V. (1997). On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25.

Torczon, V. and Trosset, M. W. (1998). Using approximations to accelerate engineering design optimization. *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, Missouri, AIAA Paper 98-4800*.

Tseng, P. (1999). Fortified-descent simplicial search method: A general approach. *SIAM Journal on Optimization*, 10(1):269–288.

van den Bergh, F. and Engelbrecht, A. (2001). Effects of swarm size on cooperative particle swarm optimisers. In *GECCO*, San Francisco, CA.

Vartiainen, E. (2000). Daylight modelling with the simulation tool DeLight. Technical Report TKK-F-A799, Helsinki University of Technology, Finland, Dept. of Engineering Physics and Mathematics.

Wetter, M. (2001). GenOpt – a generic optimization program. In Lamberts, R., Negrão, C. O. R., and Hensen, J., editors, *Proc. of the 7-th IBPSA Conference*, volume I, pages 601–608, Rio de Janeiro, Brazil.

Wetter, M. (2004). GenOpt, generic optimization program, user manual, version 2.0.0. Technical Report LBNL-54199, Lawrence Berkeley National Laboratory, Berkeley, CA, USA.

Wetter, M. and Polak, E. (2003). A convergent optimization method using pattern search algorithms with adaptive precision simulation. In Augenbroe, G. and Hensen, J., editors, *Proc. of the 8-th IBPSA Conference*, volume III, pages 1393–1400, Eindhoven, NL.

Wetter, M. and Wright, J. (2003a). Comparison of a generalized pattern search and a genetic algorithm optimization method. In Augenbroe, G. and Hensen, J., editors, *Proc. of the 8-th IBPSA Conference*, volume III, pages 1401–1408, Eindhoven, NL.

Wetter, M. and Wright, J. (2003b). A comparison of deterministic and probabilistic optimization algorithms for nonsmooth simulation-based optimization. To appear in: *Building and Environment*, 39(8):989–999.

Winkelmann, F. (2001). Modeling windows in EnergyPlus. In Lamberts, R., Negrão, C. O. R., and Hensen, J., editors, *Proc. of the 7-th IBPSA Conference*, volume I, pages 457–464, Rio de Janeiro, Brazil.

Winkelmann, F. C., Birsdall, B. E., Buhl, W. F., Ellington, K. L., Erdem, A. E., Hirsch, J. J., and Gates, S. (1993). DOE-2 supplement, version 2.1E. Technical Report LBL-34947, Lawrence Berkeley National Laboratory, Berkeley, CA, USA.

Wright, J. and Farmani, R. (2001). The simultaneous optimization of building fabric construction, HVAC system size, and the plant control strategy. In Lamberts, R., Negrão, C. O. R., and Hensen, J., editors, *Proc. of the 7-th IBPSA Conference*, volume I, pages 865–872, Rio de Janeiro, Brazil.

Wright, J. and Loosemore, H. (2001). The multi-criterion optimization of building thermal design and control. In Lamberts, R., Negrão, C. O. R., and Hensen, J., editors, *Proc. of the 7-th IBPSA Conference*, volume I, pages 873–880, Rio de Janeiro, Brazil.

Wright, J. A., Loosemore, H. A., and Farmani, R. (2002). Optimization of building thermal design and control by multi-criterion genetic algorithm. *Energy and Buildings*, 34(9):959–972.

Wright, M. H. (1996). Direct search methods: once scorned, now respectable. In Griffiths, D. F. and Watson, G. A., editors, *Numerical Analysis 1995*, pages 191–208. Addison Wesley Longman (Harlow).

Xu, P. and Haves, P. (2001). Library of component reference models for fault detection (AHU and chiller), report to California Energy Commission. Technical report, LBNL.

Appendix A

BuildOpt – Model Description

A.1 Introduction

In this Chapter, we describe in detail all models that are implemented in the BuildOpt building energy simulation program.

A.1.1 Objective and Scope of the Simulation Program

BuildOpt is a thermal building and daylighting simulation program that has been developed to be used in conjunction with optimization algorithms that adaptively control the precision of the cost function evaluations as the optimization converges to a stationary point. In BuildOpt, the building's energy load is defined by a system of differential algebraic equations (DAE system) that is automatically constructed so that it is a model for the particular building specified in the BuildOpt input file. The DAE system is built using models that are once Lipschitz continuously differentiable in the building design parameters, in the state variables and in time. This allows to prove the existence and uniqueness of a solution of the DAE system that is once continuously differentiable in the building design parameters. The DAE system is solved using the DASPK solver (Brenan et al., 1989; Brown et al., 1994, 1998). Using smooth models and the DASPK solver allows the computation of numerical approximations to the solution of the DAE system that converge to a function that is smooth in the building design parameters as the tolerance of the DAE solver is tightened. This is required in order to make BuildOpt suited for use with Generalized Pattern Search (GPS) optimization algorithms with

adaptive precision cost function evaluations.

BuildOpt can be used to simulate the energy consumption for heating, cooling and lighting of a multi-zone building, to simulate the various temperatures and energy flows inside a building and its constructions, and to compute the daylight illuminance at user-specified reference points in each room. BuildOpt has a detailed multi-zone building model but it has no models for heating, ventilation, and air-conditioning (HVAC) components. It does have, however, a simple model that computes the energy input of the heating and/or cooling system to each room, and models for HVAC components can be added if necessary.

A.1.2 Model Description

Since BuildOpt has been developed to compute numerical approximations to the cost function in building design optimization problems, we show for all variables whether they depend on building design parameters that can be expressed as continuous variables and that are likely to be used in a building design optimization problem. The building design parameters can be, for example, the length and height of windows and walls, the building azimuth, the width of a shading overhang that is placed above the window, and the transmittance of a shading device that is placed inside or outside of the window.

The energy consumption computed by BuildOpt is not differentiable in the thickness of walls, floors and ceilings and in their material properties. The reason is that Build-

Opt computes the heat conduction in opaque constructions by a finite element method which uses a mesh generator to compute the number of spatial elements in each construction. The mesh generator determines the number of spatial elements as a function of the layer thickness and the material properties. If the number of elements changes, then the number of equations in the finite element method also changes, which introduces a discontinuity. However, by changing the mesh generator so that the number of elements is constant, all equations can be made once Lipschitz continuously differentiable in these parameters as well. By doing so, one can for example find the building's thermal mass that minimizes a weighted sum of construction and energy costs.

For the description of large models, we find it convenient to list the time-independent model parameters, the time-dependent model input data, and the outputs computed by the model. We will do that in sections with the title *Module Description*. For some large models which involve many equations, we show in addition in what order the equations are evaluated. We will do that by using the notation

$$a \overset{(1)}{\longleftarrow} b,\ c,$$

which denotes that the variable a is computed using equation (1) and that equation (1) depends on the variables b and c.

A.2 Conventions

We will now define conventions that are specific to the simulation models. They are cumulative to the conventions defined on page ix.

1. All equations are written so that energy fluxes are positive if they increase the energy of the system being analyzed.

2. For empirical formulas, we use the notation $\{y\} = z$ to indicate that the unit of y is z. Unless otherwise stated, angles are always in radians.

3. Unless otherwise stated, all variables except the temperature are in SI units, i.e., we measure mass in kilograms, length in meters, time in seconds, but temperature in degree Celsius rather than in Kelvin.

In each model, all symbols are described where they appear for the first time.

A.3 Approximations for Non-Differentiable Functions

We will now define once Lipschitz continuously differentiable approximations for non-differentiable functions. These approximations are used to smoothen the model equations in the following sections.

A.3.1 Approximation for P-Controller

Consider the P-controller defined by

$$p(x) \triangleq \begin{cases} 0, & \text{if } x \leq 0, \\ x, & \text{if } 0 < x < 1, \\ 1, & \text{if } 1 \leq x. \end{cases} \tag{A.3.1a}$$

We will now define a once Lipschitz continuously differentiable approximation to (A.3.1a). For any fixed precision parameter δ, with $0 < \delta \ll 1$, we require that the approximate

P-controller, which we denote by $\widetilde{p}(\cdot;\delta)$, satisfies the conditions

$$\widetilde{p}(0;\delta) = 0, \qquad \widetilde{p}(1;\delta) = 1, \tag{A.3.1b}$$

$$\widetilde{p}(\delta;\delta) = \delta, \qquad \widetilde{p}(1-\delta;\delta) = 1-\delta, \tag{A.3.1c}$$

$$\widetilde{p}'(0;\delta) = 0, \qquad \widetilde{p}'(1;\delta) = 0, \tag{A.3.1d}$$

$$\widetilde{p}'(\delta;\delta) = 1, \qquad \widetilde{p}'(1-\delta;\delta) = 1, \tag{A.3.1e}$$

$$\widetilde{p}(x;\delta) = p(x), \qquad \text{for all } x \in \mathbb{R} \setminus ((0,\delta) \cup (1-\delta,1)), \tag{A.3.1f}$$

where we used the notation $\widetilde{p}'(x;\delta)$ to denote $\widetilde{p}'(x;\delta) \triangleq d\widetilde{p}(x;\delta)/dx$. Using a piece-wise defined third order polynomial, these conditions are satisfied by the function

$$\widetilde{p}(x;\delta) \triangleq \begin{cases} 0, & \text{if } x < 0, \\ x\left(2\frac{x}{\delta} - \left(\frac{x}{\delta}\right)^2\right), & \text{if } 0 \le x < \delta, \\ x, & \text{if } \delta \le x < (1-\delta), \\ 1 - \widetilde{p}(1-x;\delta), & \text{if } (1-\delta) \le x < 1, \\ 1, & \text{if } 1 \le x, \end{cases} \tag{A.3.1g}$$

which defines our approximate P-controller. If we do not explicitly specify the parameter δ, we mean $\widetilde{p}(x) \triangleq \widetilde{p}(x;0.05)$. Fig. A.1 shows a section of $\widetilde{p}(\cdot;0.05)$.

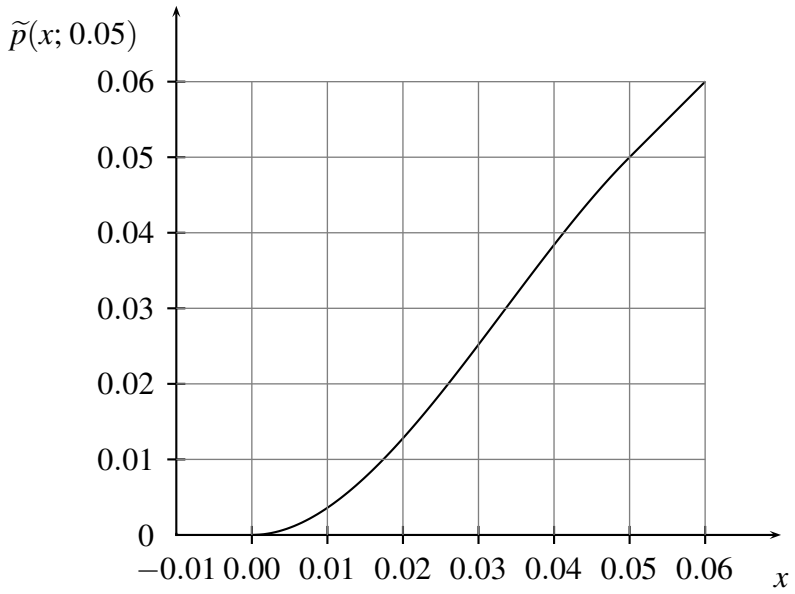


Figure A.1: Plot of $\widetilde{p}(x;\delta)$, as defined in (A.3.1g), for $-0.01 \leq x \leq 0.06$ and $\delta = 0.05$.

A.3.2 Approximation for Heaviside Function

Consider the Heaviside function

$$H(x) \triangleq \begin{cases} 0, & \text{if } x < 0, \\ 1, & \text{otherwise.} \end{cases} \tag{A.3.2a}$$

We will approximate the Heaviside function by the once Lipschitz continuously differentiable function

$$\widetilde{\mathrm{H}}(x;\delta) \triangleq \begin{cases} 0, & \text{if } x < -\delta, \\ \frac{1}{2}\left(\sin\left(\frac{x}{\delta}\frac{\pi}{2}\right)+1\right), & \text{if } -\delta \le x < \delta, \\ 1, & \text{if } \delta \le x. \end{cases} \tag{A.3.2b}$$

Equation (A.3.2b) is parametrized by δ to take the scaling of x into account. If we do not explicitly specify the parameter δ, we mean $\widetilde{\mathrm{H}}(x) \triangleq \widetilde{\mathrm{H}}(x;10^{-2})$.

A.3.3 Approximation for Minimum and Maximum Function

Equation (A.3.2b) is used to construct a once Lipschitz continuously differentiable approximation for the $\min\colon \mathbb{R}\times\mathbb{R}\to\mathbb{R}$ and the $\max\colon \mathbb{R}\times\mathbb{R}\to\mathbb{R}$ functions. In particular, for $a,b\in\mathbb{R}$, we define

$$\widetilde{\min}(a,b;\delta) \triangleq a+(b-a)\,\widetilde{\mathrm{H}}(a-b;\delta), \tag{A.3.3a}$$

$$\widetilde{\max}(a,b;\delta) \triangleq a+(b-a)\,\widetilde{\mathrm{H}}(b-a;\delta). \tag{A.3.3b}$$

We define the notation $\widetilde{\min}(a,b) \triangleq \widetilde{\min}(a,b;10^{-2})$ and $\widetilde{\max}(a,b) \triangleq \widetilde{\max}(a,b;10^{-2})$.

A.4 Physical Model

A.4.1 Introduction

We will first describe the models for external and internal heat gains, which may be caused by solar radiation or by heat gains due to people. Then, we will present the thermal models that describe the time rate of change of the building construction temperatures and the room temperatures. Afterwards, we present the daylighting model that we use to compute the room illuminance due to daylight, and the auxiliary energy used for electrical lighting to maintain a prescribed illuminance level.

A.4.2 External and Internal Heat Gains

A.4.2.1 Solar Geometry

Introduction We will first present models for the *solar time* and the *solar declination*. Then, we will deduce equations for the *zenith angle* of the sun $\theta_s(t)$ and the *azimuth* of the sun $\phi_s(t)$, which uniquely define the position of the sun in the sky. Using $\theta_s(t)$ and $\phi_s(t)$, we will deduce the equation for the sun's incidence angle on a surface. For a more detailed discussion, see Kreider and Rabl (1994) and Rabl (1985).

Model

Solar Time To compute the solar time, we first need to introduce some terminologies for the time of the day. The *Greenwich civil time* $t_{Gre,civ}(t)$ is the time along the meridian at zero longitude. The *local civil time* $t_{loc,civ}(t)$ is

$$\begin{aligned} t_{loc,civ}(t) &= t_{Gre,civ}(t) + \frac{L_{loc}}{360^\circ}\,24\text{h} = t_{Gre,civ}(t) + \frac{L_{loc}}{15^\circ/\text{h}}, \\ &\{t\} = \text{h}, \quad \{L\} = \text{DEG}, \end{aligned} \tag{A.4.1}$$

where L_{loc} is the local longitude.

The *standard time* $t_{std}(t)$ is the time of the time zone. Each time zone has a *reference meridian*, i.e., a meridian where $t_{std}(t) = t_{loc,civ}(t)$, which we will denote by L_{std}. For example, the reference meridian for U.S. Pacific Standard Time is 120° W.

Some countries have instituted an advancement of the clock by one hour during summer. This time is called *daylight saving time*. BuildOpt does not use daylight saving time.

The *solar time* $t_{sol}(t)$ is based on the apparent motion of the sun as seen from an observer on the earth surface. *Solar noon* is defined to be the time when the sun reaches the highest point in the sky. The deviation between solar time and local civil time is due to the earth's eccentric orbit. The difference between solar noon and noon of local civil

time is called the *equation of time* and denoted by $E_t(t)$. It is approximated by

$$\begin{aligned} E_t(t) &= 9.87 \sin 2B(t) - 7.53 \cos B(t) - 1.5 \sin B(t), \\ \{E_t\} &= \min, \end{aligned} \tag{A.4.2a}$$

with

$$B(t) = 360^\circ \frac{n(t) - 81}{364}, \tag{A.4.2b}$$

where n is the one-based day number (i.e., $n(t) \triangleq 1$ for January 1).

The solar time $t_{sol}(t)$ is

$$t_{sol}(t) \triangleq t_{loc,civ}(t) + \frac{E_t(t)}{60\text{min/h}}, \qquad \{t\} = \text{h}, \quad \{E_t\} = \min, \tag{A.4.3}$$

and hence,

$$\begin{aligned} t_{sol}(t) &= t_{std}(t) + \frac{L_{std} - L_{loc}}{15^\circ/\text{h}} + \frac{E_t(t)}{60\,\text{min/h}}, \\ &\{t\} = \text{h}, \quad \{L\} = \text{DEG}, \quad \{E_t\} = \min, \end{aligned} \tag{A.4.4}$$

where L_{std} is the reference meridian and L_{loc} is the local meridian.

Declination The *declination* $\delta(t)$ is defined as the angle between the equatorial plane and the solar beam as shown in Fig. A.2. The polar axis is inclined at an angle of

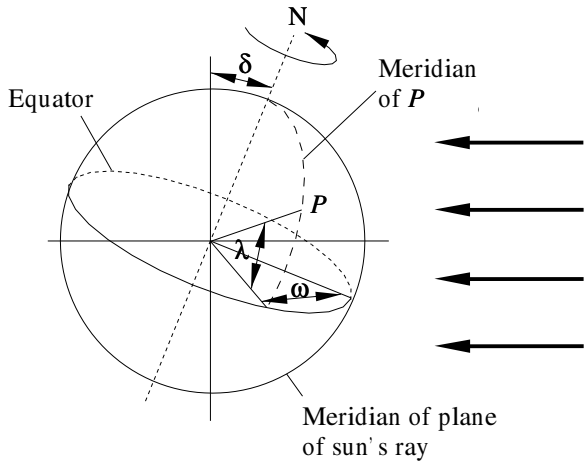


Figure A.2: Declination δ, latitude λ and solar hour ω.

23.45°. An earth revolution around the sun takes 365.25 days, and the winter solstice[1] is on December 21. Thus, the declination is

$$\sin\delta(t) = -\sin 23.45^\circ \cos\left(\frac{n(t)+10}{365.25}360^\circ\right), \tag{A.4.5}$$

where n is the one-based day number (i.e., $n(t) \triangleq 1$ for January 1).

Zenith Angle The *zenith angle* $\theta_s(t)$ of the sun is the angle between the earth surface normal and the sun's beam as shown in Fig. A.3. To compute the zenith angle, we need to introduce the *solar hour angle* $\omega(t)$. The solar hour angle is defined as the angle between the circle that passes through an observer, the north pole and the south pole, and the *hour circle*, which is defined as the circle that passes through the sun and

[1]Solstice is either of the two moments when the sun's apparent path is farthest north or south from the earth's equator. At solar noon on solstice, the sun is directly overhead the Tropic of Cancer or the Tropic of Capricorn, respectively, which are at 23.45° N and S.

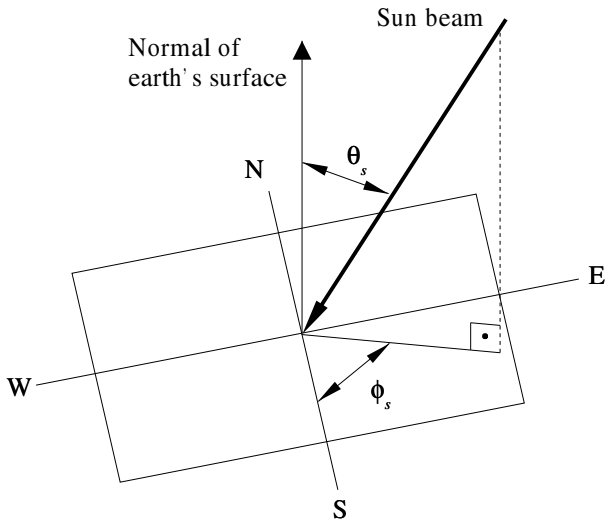


Figure A.3: Solar zenith angle and azimuth.

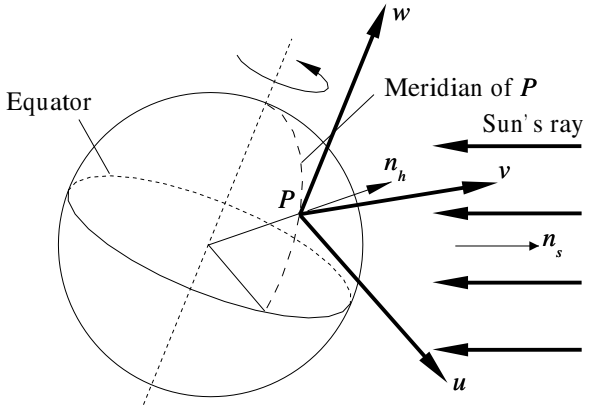


Figure A.4: Coordinate system used to obtain the solar incidence angle.

the north and south pole. Hence, the solar hour angle, if expressed in units of time, is the time that has elapsed since the sun crossed the observer's meridian the last time. It is

$$\omega(t) = \frac{(t_{sol}(t) - 12\,\mathrm{h})\,360^\circ}{24\,\mathrm{h}}, \qquad \{t\} = \mathrm{h}. \tag{A.4.6}$$

To obtain an expression for the zenith angle, we will introduce a Cartesian coordi-

nate system (u,v,w) that is fixed in the earth at the point P as shown in Fig. A.4. The directions u and v are relative to the longitude of P, with u pointing toward the sun at solar noon on equinox[2], v pointing toward east and w being parallel to the polar axis.

For a horizontal surface at latitude λ, the unit outward normal n_h and the unit vector $n_s(t)$ that points from the earth center to the sun are in the (u,v,w) coordinate system

$$n_h = \begin{pmatrix} \cos\lambda \\ 0 \\ \sin\lambda \end{pmatrix} \tag{A.4.7a}$$

and

$$n_s(t) = \begin{pmatrix} \cos\delta(t)\cos\omega(t) \\ -\cos\delta(t)\sin\omega(t) \\ \sin\delta(t) \end{pmatrix}. \tag{A.4.7b}$$

Therefore, the zenith angle can be obtained from

$$\cos\theta_s(t) = \langle n_h, n_s(t)\rangle = \cos\lambda\cos\delta(t)\cos\omega(t) + \sin\lambda\sin\delta(t). \tag{A.4.8}$$

Solar Azimuth To uniquely indicate the position of the sun, we require one more quantity, the *solar azimuth* ϕ_s, which is shown in Fig. A.3. We define $\phi_s \triangleq 0$ if the sun is

[2]Equinox is either of the two moments when the sun is exactly above the equator and day and night are of equal length.

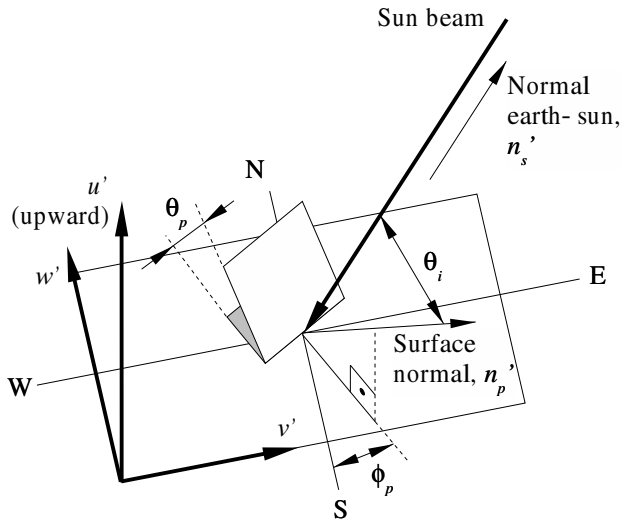


Figure A.5: Solar incidence angle on a tilted surface.

in the south, with positive values if the sun is in the west. The cosine of the solar azimuth can be expressed as (Sellers, 1965)

$$\cos\phi_s(t) = \frac{\sin\lambda\cos\theta_s(t) - \sin\delta(t)}{\cos\lambda\sin\theta_s(t)} \tag{A.4.9a}$$

and

$$\phi_s(t) = \begin{cases} -\arccos\phi_s, & \text{if } t_{sol}(t) < 12, \\ +\arccos\phi_s, & \text{otherwise.} \end{cases} \tag{A.4.9b}$$

Incidence Angle on a Tilted Surface To obtain an expression for the solar incidence angle on a tilted surface, we will introduce a new Cartesian coordinate system, which we denote by (u', v', w'). The primed coordinate system, which is shown in

Fig. A.4.2.1, is obtained by rotating (u, v, w) around the v axis so that

$$u' = u\cos\lambda + w\sin\lambda, \tag{A.4.10a}$$

$$v' = v, \tag{A.4.10b}$$

$$w' = -u\sin\lambda + w\cos\lambda. \tag{A.4.10c}$$

In the primed coordinate system, the outward unit normal of a tilted surface is

$$n'_p(x) = \begin{pmatrix} \cos\theta_p \\ -\sin\theta_p \sin\phi_p(x) \\ -\sin\theta_p \cos\phi_p(x) \end{pmatrix}, \tag{A.4.11}$$

where we defined the surface azimuth ϕ_p as $\phi_p \triangleq -90^\circ$ if the surface outward unit normal points toward east, $\phi_p \triangleq 0^\circ$ if it points toward south, and where we defined the surface tilt $\theta_p(x)$ as $\theta_p(x) \triangleq 0^\circ$ for ceilings and $\theta_p(x) \triangleq 90^\circ$ for walls.

The unit vector from the earth center to the sun is in the primed coordinate system

$$n'_s(t) = \begin{pmatrix} \cos\delta(t)\cos\omega(t)\cos\lambda + \sin\delta(t)\sin\lambda \\ -\cos\delta(t)\sin\omega(t) \\ -\cos\delta(t)\cos\omega(t)\sin\lambda + \sin\delta(t)\cos\lambda \end{pmatrix}. \tag{A.4.12}$$

Thus, the cosine of the incidence angle of the direct solar radiation on a surface with tilt

θ_p and azimuth $\phi_p(x)$ is

$$\begin{aligned}\cos\theta_i(x,t) &= \langle n'_p(x), n'_s(t)\rangle \\ &= \cos\theta_p\left(\cos\delta(t)\cos\omega(t)\cos\lambda + \sin\delta(t)\sin\lambda\right) \\ &+ \sin\theta_p \sin\phi_p(x)\cos\delta(t)\sin\omega(t) \\ &+ \sin\theta_p \cos\phi_p(x)\left(\cos\delta(t)\cos\omega(t)\sin\lambda - \sin\delta(t)\cos\lambda\right). \qquad \text{(A.4.13)}\end{aligned}$$

Module Description

Parameter

Variable	Description
L_{std}	reference meridian of time zone in DEG
L_{loc}	local meridian in DEG
λ	latitude
$\phi_p^1(x)$	azimuth of surface 1 in DEG
θ_p^1	tilt of surface 1 in DEG
$\vdots$	$\vdots$
$\phi_p^m(x)$	azimuth of surface m in DEG
θ_p^m	tilt of surface m in DEG

Input

Variable	Description
$n(t)$	one-based day number ($n \triangleq 1$ for January 1)
$t_{std}(t)$	standard time of time zone

Output

Variable	Description
$\phi_s(t)$	solar azimuth
$\theta_i^1(x,t)$	incidence angle on surface 1
$\vdots$	$\vdots$
$\theta_i^m(x,t)$	incidence angle on surface m

Algorithm

1: $B(t) \overset{(A.4.2)}{\longleftarrow} n(t)$;
2: $E_t(t) \overset{(A.4.2a)}{\longleftarrow} B(t)$;
3: $t_{sol}(t) \overset{(A.4.4)}{\longleftarrow} t_{std}(t)$, L_{std}, L_{loc}, $E_t(t)$;
4: $\omega(t) \overset{(A.4.6)}{\longleftarrow} t_{sol}(t)$;
5: $\delta(t) \overset{(A.4.5)}{\longleftarrow} n(t)$;
6: $\theta_s(t) \overset{(A.4.8)}{\longleftarrow} \lambda$, $\delta(t)$, $\omega(t)$;
7: $\cos\phi_s(t) \overset{(A.4.9)}{\longleftarrow} \lambda$, $\delta(t)$, $\omega(t)$, $\theta_s(t)$;
8: **for** $j = 1 \ldots m$; **do**
9: $\quad \theta_i^j(x,t) \overset{(A.4.13)}{\longleftarrow} \theta_p^j$, $\phi_p^j(x)$, $\omega(t)$, $\delta(t)$, λ;
10: **end for**

A.4.2.2 Solar Irradiation

Introduction We will now present models for the direct solar irradiation $H_{dir,til}(x,t)$ and the diffuse solar irradiation $H_{dif,til}(x,t)$ on a tilted surface. The model input data are the global horizontal radiation $H_{glo,hor}(t)$ and the diffuse horizontal radiation $H_{dif,hor}(t)$.

They are obtained by using cubic spline interpolation of TMY2 weather data (Marion and Urban, 1995) after shifting the time as described in Section A.4.2.4 on page 159.

Direct Solar Irradiation The direct solar irradiation on a horizontal surface is

$$H_{dir,hor}(t) = H_{glo,hor}(t) - H_{dif,hor}(t). \tag{A.4.14}$$

To obtain the direct solar irradiation an a tilted surface $H_{dir,til}(x,t)$, we use Lambert's cosine law, which states that the irradiation received by a surface is equal to the intensity impinging on the projected surface area. Thus, the direct component of the solar intensity is

$$I_{dir}(t) = \frac{H_{dir,hor}(t)}{\cos\theta_s(t)}, \tag{A.4.15}$$

where $\theta_s(t)$ is the solar incidence angle as defined in (A.4.8). Therefore, if $\theta_i(x,t)$ denotes the incidence angle of the solar radiation on a tilted surface, then the direct solar irradiation per actual unit area is

$$H_{dir,til}(x,t) = I_{dir}(t)\cos\theta_i(x,t). \tag{A.4.16a}$$

Diffuse Solar Irradiation To obtain the diffuse solar irradiation on a tilted surface $H_{dif,til}(x,t)$, BuildOpt has implemented a simple isotropic model and the more detailed Perez model. Both models have been validated by Ineichen et al. (1987). Users can

select which model should be used.

Isotropic Model In the isotropic model, which is due to Ineichen et al. (1987), the diffuse solar irradiation on a surface with tilt θ_p is

$$\widehat{H}_{dif,til}(t) = H_{dif,hor}(t)\,\frac{1+\cos\theta_p}{2}, \tag{A.4.17}$$

where the hat on $\widehat{H}_{dif,til}(t)$ indicates that the ground reflectance is not taken into account. The radiation reflected by the ground is

$$H_{ref,til}(t) = H_{glo,hor}(t)\,\rho_{gro}\,\frac{1-\cos\theta_p}{2}. \tag{A.4.18}$$

If no measurements of the ground reflectance ρ_{gro} are available, then a value of $\rho_{gro} = 0.2$ may be used, depending on the geographical location. For areas with light colored buildings, trees and mountains on the near horizon (such as in Geneva, Switzerland), this value is found to be too high (Ineichen et al., 1987).

The hemispherical diffuse irradiation $H_{dif,til}(t)$ on a surface with tilt θ_p is

$$H_{dif,til}(t) = H_{dif,hor}(t)\,\frac{1+\cos\theta_p}{2} + H_{glo,hor}(t)\,\rho_{gro}\,\frac{1-\cos\theta_p}{2} \tag{A.4.19}$$

Perez Model Perez et al. (1987) proposed a model which was later refined (Perez et al., 1990). The model is based on a geometric and an empirical component.

The geometric component describes the sky hemisphere using a circumsolar disk and a horizon band that are superimposed on an isotropic background. This configuration accounts for the most significant anisotropic effects in the atmosphere, namely forward scattering by aerosols and multiple Rayleigh scattering and retro-scattering near the horizon. The circumsolar brightening is assumed to originate from a point source at the center of the sun. The horizon brightening is assumed to be a line source at the horizon. In actuality, for clear skies, the horizon brightening is highest at the horizon and decreases in intensity away from the horizon. For overcast skies the horizon brightening has a negative value since the sky irradiation increases rather than decreases away from the horizon.

The empirical component establishes the value of a *circumsolar brightening coefficient* F_1 and a *horizon brightening coefficient* F_2 as a function of the irradiation conditions. F_1 and F_2 are parametrized by three quantities:

1. the position of the sun, defined by the zenith angle $\theta_s(t)$,

2. the brightness of the sky dome $\Delta(t)$, which is proportional to the horizontal diffuse irradiation, and

3. the sky clearness $\varepsilon(t)$, which is a function of the direct horizontal irradiation and the diffuse horizontal irradiation.

The governing equation is

$$\frac{\widehat{H}_{dif,til}(x,t)}{H_{dif,hor}(t)} = \frac{\big(1-F_1(t)\big)\,(1+\cos\theta_p)}{2} + F_1(t)\,\frac{a(x,t)}{b(t)} + F_2(t)\,\sin\theta_p. \qquad \text{(A.4.20)}$$

The coefficients $F_1(\cdot)$ and $F_2(\cdot)$ are obtained empirically from experimental data recorded on sloping surfaces. To compute $F_1(t)$ and $F_2(t)$, we first need to obtain an expression for the sky brightness, which depends on the relative air mass, and for the sky clearness. The relative air mass is (Perez, 1999)

$$m_{air}(t) = \frac{1}{\cos\big(\theta_s(t)\big)+0.15\,\big(93.9^\circ-\theta_s(t)\big)^{-1.253}}, \qquad \{m_{air}\}=-, \quad \{\theta_s\}=\text{DEG}. \qquad \text{(A.4.21)}$$

The sky brightness $\widehat{\Delta}(t)$, with $0 \le \widehat{\Delta}(t) \le 1$, is defined in Perez et al. (1990) as

$$\widehat{\Delta}(t) \triangleq H_{dif,hor}(t)\,\frac{m_{air}(t)}{I_0}, \qquad \{\Delta\}=-, \qquad \text{(A.4.22)}$$

where $I_0 = 1367.0\,\mathrm{W/m^2}$ is the extraterrestrial radiation. If $\theta_s(t)$ is close to 90°, then $\widehat{\Delta}(t)$ can be larger than unity, which can cause the diffuse radiation to be larger than the solar constant. To prevent this, we reformulate (A.4.22) as

$$\Delta(t) \triangleq \widetilde{\min}\left(H_{dif,hor}(t)\,\frac{m_{air}(t)}{I_0}, 1; 0.025\right), \qquad \{\Delta\}=-. \qquad \text{(A.4.23)}$$

The sky clearness $\varepsilon(t)$ is

$$\varepsilon(t) \triangleq \frac{\frac{H_{glo,hor}(t)}{H_{dif,hor}(t)} + 5.534\ 10^{-6}\,\theta_s^3(t)}{1 + 5.534\ 10^{-6}\,\theta_s^3(t)}, \qquad \{\theta_s\} = \text{DEG}. \tag{A.4.24}$$

A value of $\varepsilon = 1$ indicates an overcast sky, and a value of $\varepsilon = 8$ indicates a clear sky.

The circumsolar brightening coefficient $F_1(t)$ and the horizon brightening coefficient $F_2(t)$ are computed using the values tabulated in Tab. A.1 as

$$F_1(t) = \widetilde{\max}\big(0,\, F_{11}\big(\varepsilon(t)\big) + F_{12}\big(\varepsilon(t)\big)\,\Delta(t) + F_{13}\big(\varepsilon(t)\big)\,\theta_s(t);\ 0.01\big), \tag{A.4.25a}$$

$$F_2(t) = F_{21}\big(\varepsilon(t)\big) + F_{22}\big(\varepsilon(t)\big)\,\Delta(t) + F_{23}\big(\varepsilon(t)\big)\,\theta_s(t). \tag{A.4.25b}$$

In (A.4.25), $\theta_s(t)$ is in rad and not in DEG. The coefficients for the irradiance F_{ij}, with $i \in \{1,\ 2\}$ and $j \in \{1,\ 2,\ 3\}$, defined in Tab. A.1, are obtained from Perez (1999) and are of higher precision then those published in the earlier work by Perez et al. (1990).

Due to the table look-up, $F_1(\cdot)$ and $F_2(\cdot)$ are step functions. Therefore, we will interpolate the coefficients F_{ij} using the function $\widetilde{\mathrm{H}}(\cdot;\cdot)$ defined in (A.3.2b) rather than using directly the tabulated values.

ε bin	F_{11}	F_{12}	F_{13}	F_{21}	F_{22}	F_{23}
$\varepsilon \leq 1.065$	−0.0083117	0.5877285	−0.0620636	−0.0596012	0.0721249	−0.0220216
$1.065 < \varepsilon \leq 1.23$	0.1299457	0.6825954	−0.1513725	−0.0189325	0.0659650	−0.0288748
$1.23 < \varepsilon \leq 1.50$	0.3296958	0.4868735	−0.2210958	0.0554140	−0.0639588	−0.0260542
$1.50 < \varepsilon \leq 1.95$	0.5682053	0.1874525	−0.2951290	0.1088631	−0.1519229	−0.0139754
$1.95 < \varepsilon \leq 2.80$	0.8730280	−0.3920403	−0.3616149	0.2255647	−0.4620442	0.0012448
$2.80 < \varepsilon \leq 4.50$	1.1326077	−1.2367284	−0.4118494	0.2877813	−0.8230357	0.0558651
$4.50 < \varepsilon \leq 6.20$	1.0601591	−1.5999137	−0.3589221	0.2642124	−1.1272340	0.1310694
$6.20 < \varepsilon$	0.6777470	−0.3272588	−0.2504286	0.1561313	−1.3765031	0.2506212

Table A.1: Perez model coefficients for irradiance.

In (A.4.20), the term $F_1\, a/b$ accounts for the circumsolar region, with

$$a(x,t) = \widetilde{\max}(0,\ \cos\theta_i(x,t);\ 0.01), \tag{A.4.26a}$$

$$b(t) = \widetilde{\max}(\cos 85^\circ,\ \cos\theta_s(t);\ 0.01). \tag{A.4.26b}$$

Adding the radiation reflected by the ground, as defined in (A.4.18) to (A.4.20), yields the hemispherical diffuse irradiation $H_{dif,til}(x,t)$ on a surface with tilt θ_p as

$$\begin{aligned} H_{dif,til}(x,t) &= H_{dif,hor}(t) \left(\frac{\Big(1-F_1(t)\Big)\,(1+\cos\theta_p)}{2} + F_1(t)\,\frac{a(x,t)}{b(t)} + F_2(t)\,\sin\theta_p \right) \\ &\quad + H_{glo,hor}(t)\,\rho_{gro}\,\frac{1-\cos\theta_p}{2}. \end{aligned} \tag{A.4.27}$$

Module Description

Parameter

Variable	Description
n_m	number of model (1 = isotropic; 2 = Perez)
ρ_{gro}	ground reflectance, $\rho_{gro} \in [0, 1]$
θ_p^1	tilt of surface 1 in DEG
$\vdots$	$\vdots$
θ_p^m	tilt of surface m in DEG

Input

Variable	Description
$H_{glo,hor}(t)$	global horizontal irradiation per unit area
$H_{dif,hor}(t)$	diffuse horizontal irradiation per unit area
$\theta_s(t)$	solar zenith in DEG
$\theta_i^1(x,t)$	solar incidence angle on surface 1 in DEG
$\vdots$	$\vdots$
$\theta_i^m(x,t)$	solar incidence angle on surface m in DEG

Output

Variable	Description
$H_{dir,til}^1(x,t)$	direct solar irradiation on surface 1
$H_{dif,til}^1(x,t)$	diffuse solar irradiation on surface 1
$\vdots$	$\vdots$
$H_{dir,til}^m(x,t)$	direct solar irradiation on surface m
$H_{dif,til}^m(x,t)$	diffuse solar irradiation on surface m

Algorithm

{Direct irradiation}
$H_{dir,hor}(t) \overset{(A.4.14)}{\longleftarrow} H_{glo,hor}(t),\ H_{dif,hor}(t)$;
for j=1 ... m **do**
  $H^{j}_{dir,til}(x,t) \overset{(A.4.16a)}{\longleftarrow} H_{dir,hor}(t),\ \theta^{j}_{i}(x,t),\ \theta_s(t)$;
end for
{Diffuse irradiation}
if $n_m = 1$ **then**
  {Model 1: Isotropic}
  for j=1 ... m **do**
    $H^{j}_{dif,til}(t) \overset{(A.4.19)}{\longleftarrow} H_{dif,hor}(t),\ H_{glo,hor}(t),\ \theta^{j}_{p},\ \rho_{gro}$;
  end for
else
  {Model 2: Perez}
  $m_{air}(t) \overset{(A.4.21)}{\longleftarrow} \theta_s(t)$;
  $\Delta(t) \overset{(A.4.23)}{\longleftarrow} H_{dif,hor}(t),\ m_{air}(t)$;
  $\varepsilon(t) \overset{(A.4.24)}{\longleftarrow} H_{dir,hor}(t),\ H_{dif,hor}(t),\ \theta_s(t)$;
  $\{F_{1,i}\}^{3}_{i=1}(t) \longleftarrow \varepsilon(t)$; (see Tab. A.1)
  $\{F_{2,i}\}^{3}_{i=1}(t) \longleftarrow \varepsilon(t)$; (see Tab. A.1)
  $F_1(t) \overset{(A.4.25a)}{\longleftarrow} \{F_{1,j}\}^{3}_{j=1}(t),\ \Delta(t),\ \theta_s(t)$;
  $F_2(t) \overset{(A.4.25b)}{\longleftarrow} \{F_{2,j}\}^{3}_{j=1}(t),\ \Delta(t),\ \theta_s(t)$;
  $b(t) \overset{(A.4.26b)}{\longleftarrow} \theta_s(t)$;
  for j=1 ... m **do**
    $a(x,t) \overset{(A.4.26a)}{\longleftarrow} \theta^{j}_{i}(x,t)$;
    $H^{j}_{dif,til}(x,t) \overset{(A.4.27)}{\longleftarrow} H_{dif,hor}(t),\ H_{glo,hor}(t), F_1(t),\ F_2(t),\ \theta^{j}_{p},\ \rho_{gro},\ a(x,t),\ b(t)$;
  end for
end if

A.4.2.3 Air Density

The air density as a function of the altitude is

$$\rho_{air}(h) = \rho_{air,0} \exp(a/h), \tag{A.4.28}$$

where h is the elevation in meters, $\rho_{air,0} = 1.201\,\mathrm{kg/m^3}$ is the air density at sea level and $a = -1.219755\,10^{-4}\,\mathrm{m^{-1}}$ (see Judkoff and Neymark (1995)).

A.4.2.4 Weather Data

We use the Typical Meteorological Year 2 (TMY2) weather data set to obtain hourly weather data (Marion and Urban, 1995). The TMY2 weather data are based on solar time, which we compute using equation (A.4.4).

In the TMY2 weather data set, the recorded hourly solar radiation data are defined as the amount of radiative energy received during the solar hour ending at the hour indicated in the weather data file. Due to recording values that have been integrated over the last 60 minutes, the recorded radiation data have an average time delay of 30 minutes. Due to this time delay, the data set lists some hours where the direct solar radiation is nonzero but the sun is already behind the horizon. To reduce this problem and to increase the accuracy, we use solar data that were recorded 30 minutes in the future of the current solar time.

We interpolate all weather data using cubic spline interpolation (Press et al., 1992). For time stamps at which weather data are missing, the values of the last recorded hour are used.

A.4.2.5 Sky Temperature

Model To obtain an expression for the long-wave radiative heat exchange between the building and the possibly cloudy sky, we use the sky model from Martin and Berdahl (1984). Martin and Berdahl use weather data that are recorded in the Typical Meteorological Year (TMY) weather data (NCDC, 1981). Since we use TMY2 weather data which are different from TMY weather data, we have to slightly modify the computation of the infrared cloud amount $C(t)$.

To obtain the long-wave radiative heat exchange between a surface at ground level and the sky, the model computes a so-called *black-body sky temperature* $T_{sky}(t)$.

We will first obtain the *clear sky emissivity* $\varepsilon_{clear}(t)$. Martin and Berdahl describe an emissivity $\varepsilon_0(t)$ as a function of the dew point temperature $T_{dp}(t)$ as

$$\varepsilon_0(t) = 0.711 + 0.56 \left(\frac{T_{dp}(t)}{100} \right) + 0.73 \left(\frac{T_{dp}(t)}{100} \right)^2, \quad \{T_{dp}\} = {}^\circ\mathrm{C}. \tag{A.4.29}$$

They also describe a *diurnal correction* which takes different observed sky emissivities at day and night into account (Berdahl and Fromberg, 1982; Berdahl and Martin, 1984; Martin and Berdahl, 1984). The diurnal correction is

$$\Delta\varepsilon_h(t) = 0.013 \cos\left(2\pi \frac{t_{sol}(t)}{24}\right), \tag{A.4.30}$$

where $t_{sol}(t)$ is the solar time introduced in (A.4.4).

An *elevation correction*, takes the elevation of the observing weather station into account. The elevation correction is

$$\Delta\varepsilon_e(t) = 0.00012\,\big(P(t) - 1000\big), \quad \{P\} = \text{mbar}, \tag{A.4.31}$$

where $P(t)$ is the air pressure at the station.

The clear sky emissivity $\varepsilon_{clear}(t)$ is the sum of $\varepsilon_0(t)$, the diurnal correction $\Delta\varepsilon_h(t)$ and the elevation correction $\Delta\varepsilon_e(t)$:

$$\varepsilon_{clear}(t) = \varepsilon_0(t) + \Delta\varepsilon_h(t) + \Delta\varepsilon_e(t). \tag{A.4.32}$$

The presence of clouds is taken into account by introducing an *infrared cloud amount* $C(t)$, defined as

$$C(t) \triangleq \frac{\varepsilon(t) - \varepsilon_{clear}(t)}{1 - \varepsilon_{clear}(t)}, \tag{A.4.33}$$

where $\varepsilon(t)$ is the *cloudy sky emissivity*.

The infrared cloud amount $C(t)$ can also be expressed as

$$C(t) = n(t)\,\varepsilon_c(t)\,\Gamma(t), \tag{A.4.34}$$

where $n(t)$ is the fractional area of the sky that is covered by clouds, $\varepsilon_c(t)$ is the *hemispherical cloud emissivity*, and $\Gamma(t)$ is a factor that depends on the cloud base temperature. The factor $\Gamma(t)$ is small for high (cold) clouds and approaches unity for low clouds. Since usually no data for the cloud base temperature are available, Martin and Berdahl approximated $\Gamma(t)$ as a function of the cloud base height using the expression

$$\Gamma(t) = \exp\left(\frac{h(t)}{h_0}\right), \tag{A.4.35}$$

where $h(t)$ is the cloud base height and $h_0 \triangleq 8.2\,\text{km}$.

For multiple cloud layers, Martin and Berdahl added (A.4.34) for each cloud layer. However, the TMY2 weather data set contains only hourly information of (i) the *total sky cover*[3] $n_{to}(t)$, (ii) the *opaque sky cover*[4] $n_{op}(t)$, and (iii) the ceiling height h. Hence, we will obtain the sky cover due to high clouds from the total and the opaque sky cover. We assume that high clouds are thin clouds, such as cirrus clouds, and hence, compute

[3]The total sky cover is defined as the amount of the sky dome in tenths covered by clouds or obscuring phenomena (Marion and Urban, 1995).

[4]The opaque sky cover is defined as the amount of the sky dome in tenths covered by clouds or obscuring phenomena that prevents observing the sky or higher cloud layers (Marion and Urban, 1995).

their fraction that is visible from an observer on the ground as

$$n_{th}(t) = n_{to}(t) - n_{op}(t). \tag{A.4.36}$$

Assuming that the infrared and visual cloud amount are identical, we can rewrite (A.4.34) as

$$C(t) = n_{op}(t)\,\varepsilon_{c,op}\,\Gamma_{op}(t) + n_{th}(t)\,\varepsilon_{c,th}\,\Gamma_{th}(t). \tag{A.4.37}$$

We assume that all opaque clouds are completely opaque and hence we set $\varepsilon_{c,op} = 1$. To obtain $\Gamma_{op}(t)$, we use the ceiling height as recorded in the TMY2 data. If no ceiling height is recorded for the current hour, we assign a ceiling height of $h_{op} = 2\,\mathrm{km}$.

We assume that thin clouds have an emissivity of $\varepsilon_{c,th} = 0.4$, which is typical for high clouds (Martin and Berdahl, 1984), and that they are at a height of

$$h_{th}(t) = \widetilde{\max}\{h_{op}(t),\ 8\,\mathrm{km}\} \tag{A.4.38}$$

above the sea level.

Having obtained an expression for $C(t)$, we can compute $\varepsilon(t)$ from (A.4.33) as

$$\varepsilon(t) = \varepsilon_{clear}(t) + \big(1 - \varepsilon_{clear}(t)\big)\,C(t). \tag{A.4.39}$$

Finally, the black-body sky temperature can be computed as

$$T_{sky}(t) = \varepsilon(t)^{1/4}\, T_{db}(t), \tag{A.4.40}$$

where $T_{db}(t)$ is the dry bulb temperature at ground level.

Module Description

Parameter

Variable	Description
	none

Input

Variable	Description
$t_{sol}(t)$	solar time, as defined in (A.4.4)
$T_{db}(t)$	dry bulb temperature (from TMY2)
$T_{dp}(t)$	dew point temperature (from TMY2)
n_{to}	total sky cover ($n_{to}(t) \in \{0,\ 1,\ \ldots\ ,10\}$, from TMY2)
n_{op}	opaque sky cover ($n_{op}(t) \in \{0,\ 1,\ \ldots\ ,10\}$, from TMY2)
$h_{op}(t)$	ceiling height (from TMY2)
$P(t)$	atmospheric pressure at weather station (from TMY2)

Output

Variable	Description
$T_{sky}(t)$	black body sky temperature
$\varepsilon(t)$	cloudy sky emissivity

Algorithm

1: $h_{th}(t) \overset{(A.4.38)}{\longleftarrow} h_{op}(t)$;
2: $\Gamma_{op}(t) \overset{(A.4.35)}{\longleftarrow} h_{op}(t)$;
3: $\Gamma_{th}(t) \overset{(A.4.35)}{\longleftarrow} h_{th}(t)$;
4: $n_{th}(t) \overset{(A.4.36)}{\longleftarrow} n_{to}(t),\ n_{op}(t)$;
5: $C(t) \overset{(A.4.37)}{\longleftarrow} n_{op}(t),\ n_{th}(t),\ \Gamma_{op}(t),\ \Gamma_{th}(t),\ \varepsilon_{c,op},\ \varepsilon_{c,th}$;
6: $\varepsilon_0(t) \overset{(A.4.29)}{\longleftarrow} T_{dp}(t)$;
7: $\Delta\varepsilon_h(t) \overset{(A.4.30)}{\longleftarrow} t_{sol}(t)$;
8: $\Delta\varepsilon_e(t) \overset{(A.4.31)}{\longleftarrow} P(t)$;
9: $\varepsilon_{clear}(t) \overset{(A.4.32)}{\longleftarrow} \varepsilon_0(t),\ \Delta\varepsilon_h(t),\ \Delta\varepsilon_e(t)$;
10: $\varepsilon(t) \overset{(A.4.39)}{\longleftarrow} \varepsilon_{clear}(t),\ C(t)$;
11: $T_{sky}(t) \overset{(A.4.40)}{\longleftarrow} \varepsilon(t),\ T_{db}(t)$;

A.4.2.6 Heat Gains

Solar Gains To compute the solar gains per unit area for the room surfaces, we assume that all solar radiation that enters the room first hits the floor, and that the floor diffusely reflects the radiation to all other surfaces. We neglect multiple reflections and instead of using view factors between the floor and the other surfaces, we use area weighted solar distribution factors. The model is as follows:

Consider a room with N_f floor patches. Let $\mathbf{N}_f$ be the set of indices of all floor area patches and let $H(x,t)$ denote the solar radiation transmitted from the outside to the room. The solar radiation that is absorbed by the k-th floor patch per unit area is

$$q^k(x,t) = H(x,t)\,\frac{\varepsilon^k_{sol}}{\sum_{n\in\mathbf{N}_f} A^n(x)}. \tag{A.4.41a}$$

The solar radiation that is reflected from the k-th floor patch is

$$Q^k_r(x,t) = H(x,t)\,\frac{A^k(x)\,(1-\varepsilon^k_{sol})}{\sum_{n\in\mathbf{N}_f} A^n(x)}. \tag{A.4.41b}$$

Therefore, the solar radiation that is reflected by the whole floor area is

$$Q_r(x,t) = \sum_{n\in\mathbf{N}_f} Q^n_r(x,t) = H(x,t)\,\frac{\sum_{n\in\mathbf{N}_f} A^n(x)\,(1-\varepsilon^n_{sol})}{\sum_{n\in\mathbf{N}_f} A^n(x)}. \tag{A.4.41c}$$

We distribute $Q_r(x,t)$ to all non-floor areas, weighted by their solar absorptivity ε_{sol} and solar transmissivity for diffuse irradiation τ_{sol} (which is non-zero for windows). Let $\mathbf{M}$

be the index set of all non-floor surface patches. We compute the solar radiation that is absorbed or transmitted by a non-floor surface patch per unit area as

$$q^k(x,t) = Q_r(x,t)\,\frac{\varepsilon^k_{sol} + \tau^k_{sol}}{\sum_{m\in\mathbf{M}} A^m(x)\,(\varepsilon^m_{sol} + \tau^m_{sol})}. \tag{A.4.41d}$$

For opaque constructions, $q^k(x,t)$ is assumed to be completely absorbed. For windows, however, part of $q^k(x,t)$ is transmitted to the outside, and only a fraction of $q^k(x,t)$ is absorbed. Thus, we consider $q^k(x,t)$ as the solar radiation that hits the window from the room-side. What fraction of $q^k(x,t)$ will be absorbed by each window pane is determined in the window model (see Section A.4.3.3 on page 189).

Internal Gains The heat gains caused by appliances, such as computers, and by people are defined by hourly, weekly, and yearly schedules. The schedules can be defined as discrete, continuous or differentiable schedules, and it can be specified what fraction of the energy is radiative and what fraction is convective.

1. Discrete schedules do not use interpolation to obtain values at intermediate time steps. For discrete schedules, if t_i and t_{i+1} have schedule values s_i and s_{i+1}, we assign for $t \in [t_i, t_{i+1})$ the value $s(t) = s_i$.

2. Continuous schedules use linear interpolation to obtain values at intermediate time steps. If t_i and t_{i+1} have schedule values s_i and s_{i+1}, we assign for $t \in [t_i, t_{i+1})$ the value $s(t) = s_i + (s_{i+1} - s_i)\,\frac{t-t_i}{t_{i+1}-t_i}$.

3. Differentiable schedules use interpolation based on the once Lipschitz continuously differentiable Heaviside function, defined in (A.3.2b), to obtain values at intermediate time steps. If t_i and t_{i+1} have schedule values s_i and s_{i+1}, we assign for $t \in [t_i, t_{i+1})$ the value

$$s(t) = s_i + (s_{i+1} - s_i)\,\widetilde{\mathrm{H}}\left(t - t_i - \frac{t_{i+1} - t_i}{2}; \frac{t_{i+1} - t_i}{2}\right). \tag{A.4.41e}$$

This defines a once Lipschitz continuously differentiable schedule, which significantly improves the convergence properties of the DASPK solver that is used to solve the system of differential algebraic equations.

For radiative internal gains, the i-th surface is assumed to absorb the heat gain per unit area

$$H^i_{IR}(t) = \frac{\varepsilon^i_{IR}}{\sum_{n=1}^{N_{sur}} \varepsilon^n_{IR} A^n(x)}\, P(t), \tag{A.4.41f}$$

where $A^i(x)$ denotes the area of the i-th surface, with $i \in \{1, \ldots, N_{sur}\}$, ε^i_{IR} denotes the absorptivity of the i-th surface for infrared radiation, and $P(t)$ denotes the radiative internal heat gain of the zone.

A.4.3 Heat Transfer in the Building

We will now present the models that describe the heat transfer processes in the building.

A.4.3.1 Surface Heat Transfer

Introduction We will now present a simplified and a detailed model for the convective heat transfer coefficient $h_{con}(x,t)$. For exterior surfaces, the convective heat transfer coefficient $h_{con}(x,t)$ accounts for buoyancy driven free convection $h_{fre}(x,t)$ and for forced convection caused by wind $h_{for}(x,t)$. We define

$$h_{con}(x,t) \triangleq h_{fre}(x,t) + h_{for}(x,t). \tag{A.4.42}$$

For the design of thermal storage elements, such as concrete interior walls, the accurate computation of the room-side heat transfer coefficient is important since it couples the room air temperature to the storage element. Therefore, we give the user the option to select either a simplified or a detailed heat transfer model for the *interior* surfaces, but we always use a simplified model for the *exterior* surfaces.

We will also present an equation for the radiative heat transfer coefficient that is used to compute the radiative heat transfer between exterior surfaces and the sky and the ground.

Simplified Model

Convective Heat Transfer Coefficient for Free Convection For the simplified model, we will use constant convective heat transfer coefficients. EnergyPlus 1.1.0 uses for the simplified method convective heat transfer coefficients that were obtained from a correlation from Fig. 1 on page 22.4 in the ASHRAE Handbook of Fundamentals (ASHRAE, 1989). The values used in EnergyPlus are:

- For horizontal surfaces, i.e., surfaces with tilt $\theta_p \in (-22.5^\circ, 22.5^\circ)$:

$$h_{fre}(x,t) = \begin{cases} 0.948\mathrm{W}/(\mathrm{m}^2\,\mathrm{K}), & \text{for reduced convection,} \\ 4.040\mathrm{W}/(\mathrm{m}^2\,\mathrm{K}), & \text{for enhanced convection.} \end{cases} \tag{A.4.43a}$$

- For tilted surfaces:

$$h_{fre}(x,t) = \begin{cases} 2.281\mathrm{W}/(\mathrm{m}^2\,\mathrm{K}), & \text{for reduced convection,} \\ 3.076\mathrm{W}/(\mathrm{m}^2\,\mathrm{K}), & \text{for vertical surface,} \\ 3.870\mathrm{W}/(\mathrm{m}^2\,\mathrm{K}), & \text{for enhanced convection.} \end{cases} \tag{A.4.43b}$$

Reduced convection means that the heat flux is opposite to the buoyancy force, and enhanced convection means that the heat flux is in the same direction as the buoyancy force. Since the direction of the heat flux depends on the design parameter x, the equations above are discontinuous in x. Therefore, we will further simplify the model and

use for any surface and any heat flux $h_{fre} = 3\mathrm{W}/(\mathrm{m}^2\,\mathrm{K})$.

Convective Heat Transfer Coefficient for Forced Convection To take an increase in the convective heat transfer coefficient due to wind into account, we use in the simplified model $h_{for} = h_{fre}$.

Radiative Heat Transfer at Exterior Surfaces To calculate infrared radiation between the exterior building surface and the outside environment, we treat the environment as an enclosure with much larger area than the building surface. Then, the heat exchange is

$$q_{IR}(x,t) = \sigma \varepsilon_{IR} \left(T_{env}^4(t) - T_{sur}^4(x,t)\right), \tag{A.4.44a}$$

where $T_{env}(\cdot)$ denotes the environment temperature and $T_{sur}(\cdot,\cdot)$ denotes the surface temperature of the building construction element for which the heat transfer is computed.

Let $F_{sur-gro}$ denote the view factor from the surface to the ground. We define the environment temperature as

$$T_{env}(t) \triangleq \sqrt[4]{F_{sur-gro}\, T_{out}^4(t) - (1 - F_{sur-gro})\, T_{sky}^4(t)}, \tag{A.4.44b}$$

where $T_{out}(t)$ is the outside air temperature, and $T_{sky}(t)$ is the sky temperature defined in (A.4.40). For equation (A.4.44b), we assumed that the ground temperature is equal to the air temperature.

We linearize (A.4.44a) to obtain

$$q_{IR}(x,t) = 4\sigma\varepsilon_{IR}T_{env}^3(t)\,(T_{env}(t) - T_{sur}(x,t)). \tag{A.4.44c}$$

Thus, we can define a radiative heat transfer coefficient as

$$h_{IR}(t) = 4\sigma\varepsilon_{IR}T_{env}^3(t). \tag{A.4.44d}$$

Radiative Heat Transfer To compute the radiative heat transfer between the room surfaces, we define a *radiation temperature* $T^*(x,t)$ as

$$T^*(x,t) \triangleq \frac{\sum_{n=1}^{N_{sur}} \varepsilon_{IR}^n A^n(x) T_{sur}^n(x,t)}{\sum_{n=1}^{N_{sur}} \varepsilon_{IR}^n A^n(x)}. \tag{A.4.45a}$$

We assume that each surface only exchanges radiation with an imaginary surface of much larger area which is at temperature $T^*(x,t)$. By using a linearized heat transfer coefficient, defined as

$$h_{IR}^n(x,t) \triangleq 4\sigma\varepsilon_{IR}^n T^*(x,t)^3, \tag{A.4.45b}$$

we can write

$$q_{IR}^n(x,t) = h_{IR}^n(x,t)\left(T^*(x,t) - T^n(x,t)\right). \tag{A.4.45c}$$

Equations (A.4.45c) and (A.4.45a) are consistent with conservation of energy because

$$\begin{aligned}\sum_{n=1}^{N_{sur}} A^n q_{IR}^n(x,t) &= 4\sigma\left(T^*(x,t)\right)^3 \sum_{n=1}^{N_{sur}} \varepsilon_{IR}^n A^n \left(T^*(x,t) - T^n(x,t)\right) \\ &= 4\sigma\left(T^*(x,t)\right)^3 \left(\sum_{n=1}^{N_{sur}} \varepsilon_{IR}^n A^n T^*(x,t) - \sum_{n=1}^{N_{sur}} \varepsilon_{IR}^n A^n T^n(x,t)\right) \\ &= 0. \end{aligned} \tag{A.4.45d}$$

Detailed Model We will now present the detailed model which the user can select for computing the room-side heat transfer coefficient.

Convective Heat Transfer Coefficient for Free Convection In the detailed model, the convective heat transfer coefficient for free convection $h_{fre}(x,t)$ is for each surface computed as a function of the temperature difference $(T_\infty(x,t) - T_{sur}(x,t))$, with $T_\infty(x,t) \triangleq T_{air}(x,t)$ for interior surfaces and $T_\infty(x,t) \triangleq T_{out}(t)$ for exterior surfaces. We use the once Lipschitz continuously differentiable equation

$$h_{fre}(x,t) = \widetilde{\max}(1, \widehat{h}_{fre}(x,t); 0.1), \tag{A.4.46a}$$

where

$$\widehat{h}_{fre}(x,t) = \begin{cases} 1.310\left|T_\infty(x,t) - T_{sur}(x,t)\right|^{1/3}, & \text{for vert. surface,} \\ 1.510\left|T_\infty(x,t) - T_{sur}(x,t)\right|^{1/3}, & \text{for horiz. surf., enhanced conv.,} \\ 0.760\left|T_\infty(x,t) - T_{sur}(x,t)\right|^{1/3}, & \text{for horiz. surf., reduced conv.} \end{cases} \tag{A.4.46b}$$

Enhanced convection means that the heat flux is in the same direction as the buoyancy force, and reduced convection in the other direction. We smoothen the computation by using (A.4.46a) rather than only (A.4.46b) because the convective heat transfer, if computed as $q(x,t) = c\left(T_\infty(x,t) - T_{sur}(x,t)\right)\left|T_\infty(x,t) - T_{sur}(x,t)\right|^{1/3}$, for some $c \neq 0$, is not Lipschitz continuously differentiable in the temperature difference as $|T_\infty(x,t) - T_{sur}(x,t)| \to 0$.

Note that we described the convective heat transfer coefficient as a function of the temperature difference only. More detailed correlations that account for the increased air velocity near the room surfaces when the HVAC system is running can be found in Clarke (2001), but they are not implemented in the current version of BuildOpt.

Radiative Heat Transfer for Interior Surfaces In the detailed model, the radiative heat transfer for interior surfaces is computed as in the simplified model.

Module Description For Convective Heat Transfer Coefficients

Parameter

Variable	Description
θ_p	surface tilt in DEG (0° for floor; ±90° for wall; 180° for ceiling)
For exterior surfaces only:	
n_m	number of model (1 = simplified model; 2 = detailed model)
ε_{IR}	infrared emissivity
$F_{sur-gro}$	view factor from surface to ground

Input

Variable	Description
$T_{sur}(x,t)$	surface temperature
$T_\infty(x,t)$	air temperature ($T_{out}(t)$ or $T_{air}(x,t)$)
For exterior surfaces only:	
$T_{sky}(t)$	sky temperature as defined in (A.4.40)

Output

Variable	Description
$h_{con}(x,t)$	convective heat transfer coefficient

A.4.3.2 Heat Conductance in Opaque Material

We will now describe how the heat conductance through walls, ceilings and floors is computed.

Governing Equation The one-dimensional heat conductance in composite constructions that are composed of layers of opaque materials can be described by

$$\frac{\partial}{\partial z}\left(k(z)\frac{\partial T(x;z,t)}{\partial z}\right) = C(z)\frac{\partial T(x;z,t)}{\partial t}, \tag{A.4.47a}$$

for $(z,t) \in \Omega \times (0,\tau)$, for some $\tau > 0$, where $\Omega \triangleq \big(0\,,l\big)$ and $l > 0$ denotes the construction length and $T(x;\cdot,\cdot)$ denotes the construction temperature. We make the convention that the room-side surface is at $z = 0$ and that the outside surface is at $z = l$. The coefficients $k,C:\ \mathbb{R} \to \mathbb{R}$ are piece-wise constant functions for the heat conductivity and the specific heat capacity per unit volume, respectively. We use the initial data

$$T(x;z,0) = \eta(x;z), \quad z \in \Omega, \tag{A.4.47b}$$

where $\eta(x;\cdot)$ is continuous, and we use the boundary conditions, for $t \in (0,\tau)$,

$$k(0)\,\frac{\partial T(x;0,t)}{\partial z} - h_0(x;t)\,T(x;0,t) \;=\; g_0(x;t), \tag{A.4.47c}$$

$$-k(l)\,\frac{\partial T(x;l,t)}{\partial z} - h_1(x;t)\,T(x;l,t) \;=\; g_1(x;t), \tag{A.4.47d}$$

where $h_0(x;\cdot)$, $h_1(x;\cdot)$, $g_0(x;\cdot)$, and $g_1(x;\cdot)$ are once Lipschitz continuously differentiable functions if the weather data and the internal heat gains are interpolated using a once Lipschitz continuously differentiable function (see Section A.4.2.4 and Section A.4.2.6). They are defined as

$$h_0(x;t) \;\triangleq\; h_{con,0}(x;t) + h_{lw,0}(x;t) \tag{A.4.48a}$$

$$h_1(x;t) \;\triangleq\; h_{com,1}(x;t) + h_{lw,1}(x;t) \tag{A.4.48b}$$

where $h_{con,0}(x,\cdot)$ is the room-side convective heat transfer coefficient, $h_{lw,0}(x,\cdot)$ is the room-side radiative heat transfer coefficient, $h_{con,1}(x,\cdot)$ is the outside convective heat transfer coefficient, and $h_{lw,1}(x,\cdot)$ is the outside radiative heat transfer coefficient, all defined in Section A.4.3.1 on page 169.

The functions $g_0(x;\cdot)$ and $g_1(x;\cdot)$ are defined as

$$g_0(x;t) \triangleq -h_{con,0}(t)\,T_{\infty,0}(x;t) - h_{lw,0}(x;t)\,T_{env,0}(x;t) - g_0''(x;t), \quad \text{(A.4.49a)}$$

$$g_1(x;t) \triangleq -h_{con,1}(t)\,T_{\infty,l}(x;t) - h_{lw,1}(x;t)\,T_{env,1}(x;t) - g_1''(x;t), \quad \text{(A.4.49b)}$$

where $T_{\infty,0}(x;\cdot)$ and $T_{\infty,1}(x;\cdot)$ are the air temperatures outside the thermal boundary layer, $T_{env,0}(x;\cdot)$ is the radiation temperature introduced in (A.4.45a), $T_{env,1}(x;\cdot)$ is for exterior constructions the environment temperature introduced in (A.4.44b) and for interior construction the room radiation temperature $T^*(x,t)$ introduced in (A.4.45a) of the adjacent zone, and $g_0''(x;\cdot)$ and $g_1''(x;\cdot)$ are the short wave radiative heat gains of the surface due to solar radiation or due to radiative internal heat gains caused by appliances, such as lights and computers, which are once Lipschitz continuously differentiable functions if the weather data and the internal heat gains are interpolated using a once Lipschitz continuously differentiable function (see Section A.4.2.4 and Section A.4.2.6).

By introducing the notations

$$T_t(x;z,t) \triangleq \frac{\partial T(x;z,t)}{\partial t}, \qquad T_z(x;z,t) \triangleq \frac{\partial T(x;z,t)}{\partial z}, \quad \text{(A.4.50a)}$$

and the second order differential operator

$$\mathcal{L}(\cdot) \triangleq -\frac{\partial}{\partial z}\left(k(z)\frac{\partial(\cdot)}{\partial z}\right), \tag{A.4.50b}$$

equation (A.4.47) can be formulated in the more concise form, for $z \in \Omega$ and $t \in (0, \tau)$,

$$\begin{aligned} C(z)\,T_t(x;z,t) + \mathcal{L}T(x;z,t) &= 0, && \text{(A.4.51a)}\\ T(x;z,0) &= \eta(x;z), && \text{(A.4.51b)}\\ k(0)\,T_z(x;0,t) - h_0(t)\,T(x;0,t) &= g_0(x;t), && \text{(A.4.51c)}\\ -k(l)\,T_z(x;l,t) - h_1(t)\,T(x;l,t) &= g_1(x;t). && \text{(A.4.51d)} \end{aligned}$$

In composite walls, the material properties of adjacent layers, say in material 1 and material 2, can be so that $k_1/k_2 \approx 10^{-1}$, $C_1/C_2 \approx 10^2$, and hence the ratio of the thermal diffusivities $\alpha \triangleq k/C$ can be $\alpha_1/\alpha_2 \approx 10^1$. Thus, the thermal diffusivity in adjacent layers can vary by an order of magnitude. Because the coefficients $k(\cdot)$ and $C(\cdot)$ are discontinuous, there is no function $u(x;\cdot,\cdot)$ that is two times continuously differentiable in z at the material interfaces and that satisfies (A.4.51). Thus, equation (A.4.51) does not have a classical solution, and we therefore seek a weak solution using the Galerkin method.

Galerkin Approximations In the Galerkin method (Evans, 1998; Strang and Fix, 1973), one builds a weak solution of (A.4.51) by first constructing solutions of certain finite-

dimensional approximations to (A.4.51) and then passing to limits. We will use continuous, piece-wise linear functions to approximate a weak solution of (A.4.51). We will now explain our implementation.

For $j \in \{1, 2, \ldots\}$, let $\phi_j : \Omega(\cdot) \to \mathbb{R}$ be piece-wise linear hat functions, which we will call *test functions*, and which are defined for $0 \leq z_{j-1} < z_j < z_{j+1} \leq l$ and for all $z \in (0, l)$ by

$$\phi_j(z) \triangleq \begin{cases} \frac{z - z_{j-1}}{z_j - z_{j-1}}, & \text{if } z \in \big[z_{j-1}, z_j\big), \\ \frac{z - z_{j+1}}{z_j - z_{j+1}}, & \text{if } z \in \big[z_j, z_{j+1}\big), \\ 0, & \text{otherwise.} \end{cases} \tag{A.4.52}$$

For a fixed positive integer N, we construct an approximate solution $T_N(x;\cdot,\cdot)$ of the form

$$T_N(x;z,t) = \sum_{j=1}^{N} \gamma_N^j(x;t)\,\phi_j(z), \tag{A.4.53}$$

where the coefficients $\{\gamma_N^j(x;\cdot)\}_{j=1}^N$ are so that

$$\gamma_N^j(x;0) = \int_\Omega \eta(x;z)\,\phi_j(z)dz, \qquad j \in \{1,\ldots,N\}, \tag{A.4.54a}$$

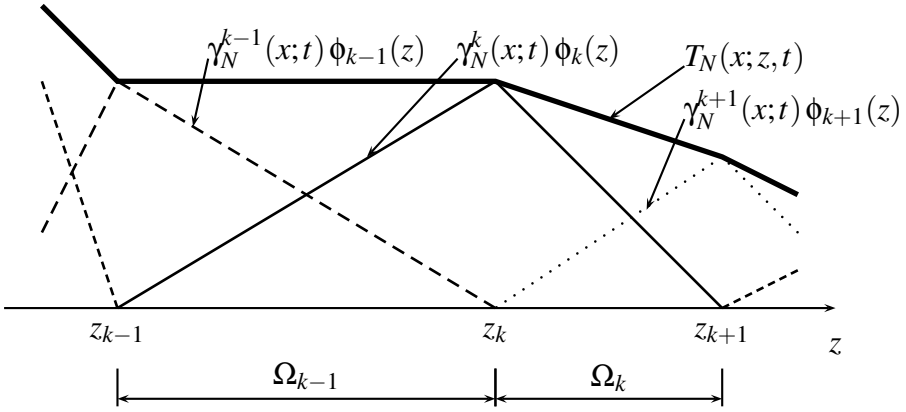


Figure A.6: Approximate solution constructed by the Galerkin method.

and

$$\int_\Omega \left(C(z)\frac{\partial T_N(x;z,t)}{\partial t} + \mathcal{L}T_N(x;z,t) \right) \phi_j(z)\,ds = 0, \qquad t \in (0,\tau), \quad j \in \{1,\ldots,N\}. \tag{A.4.54b}$$

Fig. A.6 shows an example of an approximate solution $T_N(x;\cdot,\cdot)$ constructed using piecewise linear hat functions.

Construction of Test Functions An expression for the coefficients $\{\gamma_N^j(x;\cdot)\}_{j=1}^N$ is obtained by substituting (A.4.53) into (A.4.54b). Using integration by parts, we obtain,

for $k \in \{1,\ldots,N\}$,

$$\begin{aligned} 0 &= \sum_{j=1}^{N} \int_{\Omega} C(z) \frac{d\gamma_N^j(x;t)}{dt} \phi_j(z)\,\phi_k(z) + \gamma_N^j(x;t)\,\mathcal{L}\big(\phi_j(z)\big)\,\phi_k(z)\,dz \\ &= \sum_{j=1}^{N} \left(\frac{d\gamma_N^j(x;t)}{dt} \int_{\Omega} C(z)\,\phi_j(z)\,\phi_k(z)\,dz + \gamma_N^j(x;t) \int_{\Omega} k(z) \frac{d\phi_j(z)}{dz} \frac{d\phi_k(z)}{dz} dz \right) \\ &\quad - k(z) \frac{dT_N(x;z)}{dz} \phi_k(z) \Big|_{z=0}^{l}. \end{aligned} \tag{A.4.55}$$

We now divide the domain Ω into $N-1$ subdomains $\Omega_j \triangleq \big(z_j, z_{j+1}\big)$, with $j \in \{1,\ldots,N-1\}$, which we will call *elements*. The locations of the support points z_j are so that each point of discontinuity of $k(\cdot)$ and $C(\cdot)$ coincides with some z_j.

The integrals in (A.4.55) are zero except for $j = k-1$, for $j = k$, and for $j = k+1$. Since the elements Ω_j differ from each other only in a scaling factor, it is convenient to evaluate the different components of (A.4.55) on a so-called *master element*, which is indicated by a superscript "M" and which is shown in Fig. A.7.

In terms of the master element, the integrals in (A.4.55) have the solutions

$$\int_{\Omega^M} \phi_1^M(z)\,\phi_2^M(z)\,dz = \int_{z_1^M}^{z_2^M} \frac{z - z_1^M}{z_2^M - z_1^M} \frac{z_2^M - z}{z_2^M - z_1^M} dz = \frac{z_2^M - z_1^M}{6}, \tag{A.4.56a}$$

$$\int_{\Omega^M} \big(\phi_1^M(z)\big)^2\,dz = \int_{\Omega^M} \big(\phi_2^M(z)\big)^2\,dz = \frac{z_2^M - z_1^M}{3}, \tag{A.4.56b}$$

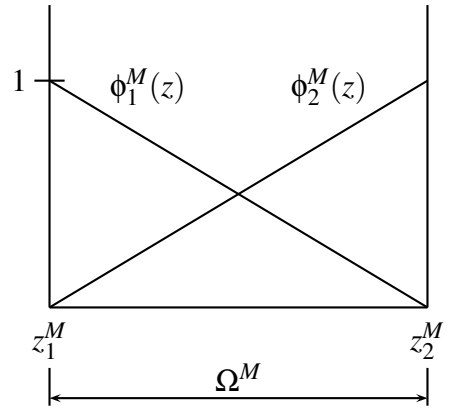


Figure A.7: Master element with master basis functions.

and

$$\begin{aligned}\int_{\Omega^M} \frac{d\phi_1^M(z)}{dz}\,\frac{d\phi_2^M(z)}{dz}\,dz &= -\int_{\Omega^M}\left(\frac{d\phi_1^M(z)}{dz}\right)^2 dz \\ &= -\int_{\Omega^M}\left(\frac{d\phi_2^M(z)}{dz}\right)^2 dz = \frac{1}{z_1^M - z_2^M}. \quad \text{(A.4.56c)}\end{aligned}$$

Implementation of the Boundary Conditions The boundary conditions of the third-type, defined in (A.4.51c) and in (A.4.51d), are in terms of the approximate solution $T_N(x;\cdot,\cdot)$,

$$k(0)\,\frac{\partial T_N(x;0,t)}{\partial z} = g_0(x;t) + h_0(t)\,T_N(x;0,t), \quad \text{(A.4.57a)}$$

$$-k(l)\,\frac{\partial T_N(x;l,t)}{\partial z} = g_1(x;t) + h_1(t)\,T_N(x;l,t). \quad \text{(A.4.57b)}$$

Substituting (A.4.53) into (A.4.57) and substituting (A.4.57) for the boundary term of (A.4.55) yields at the room-side surface the expression

$$\begin{aligned} 0 &= \sum_{j=1}^{2} \left(\frac{d\gamma_N^j(x;t)}{dt} \int_{\Omega_1} C(z)\,\phi_j(z)\,\phi_1(z)\,dz + \gamma_N^j(x;t) \int_{\Omega_1} k(z)\,\frac{d\phi_j(z)}{dz}\,\frac{d\phi_1(z)}{dz}\,dz \right) \\ &\quad + g_0(x;t) + h_0(t)\,\gamma_N^1(x;t). \end{aligned} \tag{A.4.58a}$$

To obtain (A.4.58a), we used the fact that $\phi_1(0) = 1$ and $\phi_j(z) = 0$ for $z \in \Omega_1$ and $j > 2$. Hence, $T_N(x;0,t) = \gamma_N^1(x;t)$. Similarly, we obtain for the exterior boundary condition, at $z = l$,

$$\begin{aligned} 0 &= \sum_{j=N-1}^{N} \left(\frac{d\gamma_N^j(x;t)}{dt} \int_{\Omega_{N-1}} C(z)\,\phi_j(z)\,\phi_N(z)\,dz \right. \\ &\quad \left. + \gamma_N^j(x;t) \int_{\Omega_{N-1}} k(z)\,\frac{d\phi_j(z)}{dz}\,\frac{d\phi_N(z)}{dz}\,dz \right) \\ &\quad + g_1(x;t) + h_1(t)\,\gamma_N^N(x;t). \end{aligned} \tag{A.4.58b}$$

The initial conditions $\gamma_N^j(x;0)$ follow directly from (A.4.54a) and are

$$\gamma_N^j(x;0) = \eta(x;z_j), \qquad j \in \{1,\ldots,N\}. \tag{A.4.59}$$

The equations (A.4.55) together with (A.4.59) define a system of N linear equations of

the form

$$A\frac{d\gamma_N(x;t)}{dt} = B(t)\gamma_N(x;t)+c(x;t), \qquad t\in(0,\tau) \tag{A.4.60a}$$

$$\gamma_N^j(x;0) = \eta(x;z_j), \qquad j\in\{1,\ldots,N\}, \tag{A.4.60b}$$

where $A,B\in\mathbb{R}^N\times\mathbb{R}^N$ and $c\in\mathbb{R}^N$. The functions $B(\cdot)$ and $c(x;\cdot)$ depend on time due to the boundary conditions, which are computed by models that define the boundary conditions as once Lipschitz continuously differentiable functions if the weather data and the internal heat gains are interpolated using a once Lipschitz continuously differentiable function (see Section A.4.2.4 and Section A.4.2.6). Hence, the functions $B(\cdot)$ and $c(x;\cdot)$ are once Lipschitz continuously differentiable in time. This implies that (A.4.60) has a unique solution that is continuously differentiable in $\gamma_N(\cdot;0)$, in t and in x (see for example Coddington and Levinson (1955, Theorem 7.1 and 7.2)). Note that A is time-invariant.

Computational Procedure Next, we outline the computer implementation of the Galerkin method.

Grid Generation We will first present a heuristic approach to generate the spatial grid.

As mentioned above, the difference in the thermal diffusivity of adjacent layer mate-

rials can be an order of magnitude. Hence, if the spatial grid generation does not account for the material properties, then the time rate of change of the different nodes can be significantly different from each other, which can cause the system of ordinary differential equations to be stiff. Thus, we attempt to generate the spatial grid so that under the assumption of equal heat transfer, each node temperature has a similar time rate of change. In addition, we refine the grid near the boundary of the domain Ω.

From dimensionless analysis, one can obtain a characteristic time, called the *Fourier* number, as

$$Fo \triangleq \frac{\alpha t}{L^2}, \tag{A.4.61}$$

where α denotes the thermal diffusivity, t denotes time and L denotes the characteristic length (Holman, 1997). We like to generate the spatial grid so that the ratio (t/Fo) is equal to an arbitrary constant Π, which we define as

$$\Pi \triangleq \left(\frac{t}{Fo}\right)^{1/2} = \frac{L}{\sqrt{\alpha}}. \tag{A.4.62}$$

Now, let $K \in \mathbb{N}$ denote the number of material layers in an opaque composite construction, and let $\{l_k\}_{k=1}^{K}$ denote the thickness of each material layer. In view of (A.4.62), we compute the time constant of each material layer as

$$\Pi_k = \frac{l_k}{\sqrt{\alpha_k}}, \qquad k \in \{1, \ldots, K\}, \tag{A.4.63}$$

and we compute the estimated number of elements $\widehat{N} \in \mathbb{R}$ for the construction as[5]

$$\widehat{N} = N_{ref} \frac{\sum_{k=1}^{K} \Pi_k}{\Pi_{ref}}, \tag{A.4.64}$$

where $N_{ref} \in \mathbb{N}$ is a user-specified number of elements for a reference material, which we define as a concrete construction with thickness $L_{ref} \triangleq 0.20\,\mathrm{m}$ and thermal diffusivity $\alpha_{ref} \triangleq 3.64 \cdot 10^{-7}\,\mathrm{m}^2/\mathrm{s}$. Hence, $\Pi_{ref} \triangleq L_{ref}/\sqrt{\alpha_{ref}} = 331.4\,\mathrm{s}^{1/2}$.

Next, we define the number of elements for each material layer $N_k \in \mathbb{N}$ as

$$N_k = \left\lceil \widehat{N} \frac{\Pi_k}{\sum_{i=1}^{K} \Pi_i} \right\rceil, \qquad k \in \{1, \ldots, K\}, \tag{A.4.65}$$

where the notation $\lceil \cdot \rceil$ is defined for $s \in \mathbb{R}$ as $\lceil s \rceil \triangleq \min\{k \in \mathbb{Z} \mid k \geq s\}$.

Then, we divide each material layer $k \in \{1, \ldots, K\}$ in compartments of length $\Delta_k \triangleq l_k/N_k$. Finally, in the elements that are coupled to the boundary conditions, we place one more node in the element center. Thus, the total number of elements in the construction is $N = 2 + \sum_{k=1}^{K} N_k$.

Construction of Linear System of Equations Let $A_{i,j}$ and $B_{i,j}$ denote the (i,j)-th element of the matrix A or B, respectively. To compute the matrices A and B, it is convenient to first perform the integrations in (A.4.55) and in (A.4.58) over each element Ω_k, with $k \in \{1, \ldots, N-1\}$, and then add the contributions to the appropriate element of

[5] We use $\widehat{N}$ as a real number and will round later to an integer.

A or B.

To do so, we start by setting all elements of A and B equal to zero. For an element Ω_k, with $k \in \{1,\ldots,N-1\}$, all basis functions are zero, except $\phi_k(\cdot)$ and $\phi_{k+1}(\cdot)$. Thus, for the k-th element, we have the possible combinations $(i,j) \in \{(k,k), (k,k+1), (k+1,k), (k+1,k+1)\}$. We evaluate the integrals in (A.4.55) for those combinations (setting the boundary terms equal to zero for *every* index k), and then add the contributions of each combination (i,j) to the element $A_{i,j}$ or $B_{i,j}$.

After doing this for $k \in \{1,\ldots,N\}$, we add the contributions of the boundary term of (A.4.58) to the matrix B and the vector c. (Note that the matrix A is independent of the boundary equations.)

Solution of the Equations The system of ordinary differential equations defined in (A.4.60) is coupled to the ordinary differential equations that describe the time rate of change of the room air temperatures and to the algebraic equations that describe the window surface temperatures, which are assumed to have negligible thermal mass. The coupling is due to the boundary conditions, which are functions of the room air temperature and the surface temperatures of other constructions. All equations of this system of differential algebraic equations are solved simultaneously using the commercial DAE solver DASPK (Brenan et al., 1989; Brown et al., 1994, 1998). In Brown et al. (1994), it is described how the DASPK solver computes consistent initial conditions $\{\eta(x;z_j)\}_{j=1}^{N}$.

Module Description

Parameter

Variable	Description
N_{ref}	number of elements for reference construction
$\{l^i\}_{i\in\mathbf{N}_l}$	thickness of each layer
$\{k^i\}_{i\in\mathbf{N}_l}$	thermal conductivity of each layer
$\{C^i\}_{i\in\mathbf{N}_l}$	specific heat capacity per unit volume of each layer

Output

Variable	Description
$\{\frac{d\gamma_N^j(x;t)}{dt}\}_{j=1}^N$	time derivative of each node temperature
$\{z^j\}_{j=1}^N$	spatial coordinate of each node

Input

Variable	Description
$T_{\infty,0}(x;t)$	room-side temperature outside the convective boundary layer
$T_{env,0}(x;t)$	room-side radiation temperature

Continued on next column.

Input (continued)

Variable	Description
$T_{\infty,l}(x;t)$	outside temperature outside the convective boundary layer
$T_{env,1}(x;t)$	outside radiation temperature
$h_{con,0}(x;t)$	room-side convective heat transfer coefficient
$h_{com,1}(x;t)$	outside convective heat transfer coefficient
$h_{lw,0}(x;t)$	room-side radiative heat transfer coefficient
$h_{lw,1}(x;t)$	outside radiative heat transfer coefficient
$g_0(x;t)$	short-wave radiation absorbed by the room-side surface
$g_1(x;t)$	short-wave radiation absorbed by the outside surface

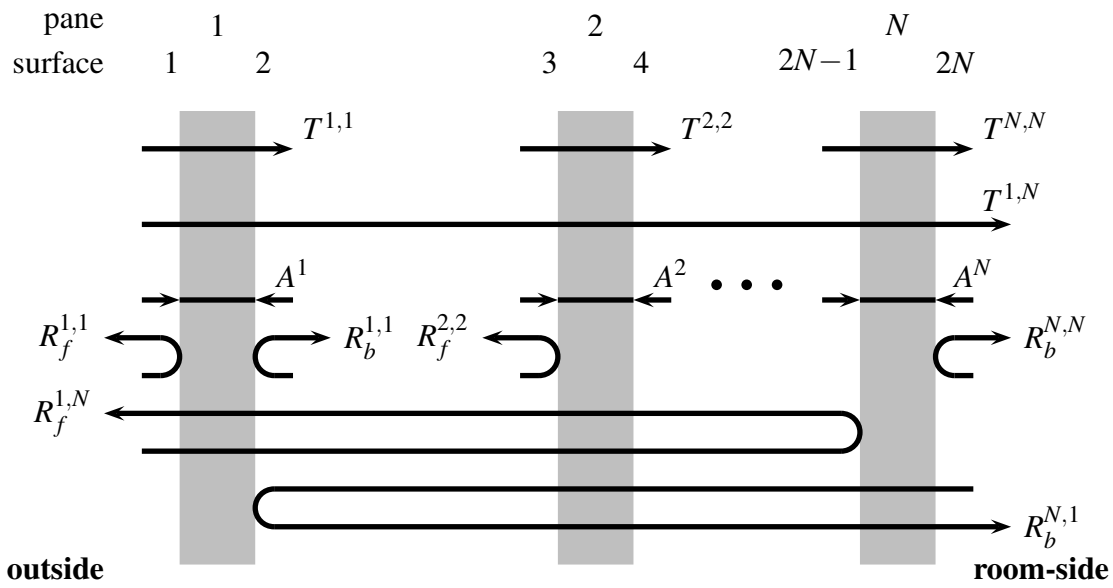


Figure A.8: Transmittance and absorbtance of a window with N panes ($N > 1$) and radiation source at the outside.

A.4.3.3 Window Simulation for Short-Wave Radiation

Introduction We will now present how the short-wave absorbtance and the transmittance of each pane, and the short-wave transmittance of the window are computed for a given angle of incidence, or, alternatively, for hemispherical irradiation. We will also present how the front and the back reflectance of the window are computed. After describing how these properties are computed, we describe how they are modified to take into account either an exterior or an interior shading device. The absorbtance of each pane, the transmittance of the window, and the short-wave absorbtance of the interior shading device are then used in the models that compute the temperature of the panes and the daylight illuminance.

Our short-wave radiation model is similar to the WINDOW 4 model (Arasteh et al., 1989; Finlayson et al., 1993). For the panes and surfaces, we will use the same numbering scheme as WINDOW 4 which is shown in Fig. A.8 and in Fig. A.10. First, we need to introduce some definitions and nomenclatures.

Definitions and Nomenclatures Our computation is based on spectral properties that are for energy calculations averaged over the whole solar spectrum and for daylighting calculations averaged over the visible spectrum.[6] Since the models are equal for the radiation data averaged over the solar and for the radiation data averaged over the visible spectrum, we will call these averaged properties for both spectra *total* properties.

The computation of the window properties are done for specular irradiation (i.e., for a particular angle of incidence ϕ_i) and for hemispherical irradiation. Our model requires as parameters the spectrally averaged glass properties for each pane at normal incidence angle. In our model, we first convert these properties to specular and hemispherical properties for each pane, and then compute the properties for the window. We will denote properties for normal incidence by $P^{\perp}$, directional properties by $P^{\sphericalangle}$ and hemispherical properties by $P^{\cap}$.

We will compute a directional and a hemispherical total absorbtance of each pane,

[6] A database with these properties can be obtained from `http://windows.lbl.gov/`.

denoted by $\{A^{\lhd,j}\}_{j=1}^{N}$ and by $\{A^{\cap,j}\}_{j=1}^{N}$. They are defined for the j-th pane as the ratio between the absorbed radiation q_{abs}^{j} and the total directional irradiation or the total hemispherical irradiation, respectively, $H^{\lhd,1}$ and $H^{\cap,1}$. Hence,

$$A^{\lhd,j} \triangleq \frac{q_{abs}^{\lhd,j}}{H^{\lhd,1}}, \tag{A.4.66a}$$

$$A^{\cap,j} \triangleq \frac{q_{abs}^{\cap,j}}{H^{\cap,1}}. \tag{A.4.66b}$$

Similarly, the directional and hemispherical total transmittance of the window, $T^{\lhd}$ and $T^{\cap}$, are defined as the ratio between the radiation transmitted through the window, $H_T^{\lhd}$ or $H_T^{\cap}$, and the total directional or hemispherical irradiation at the window outside, $H^{\lhd,1}$ or $H^{\cap,1}$, i.e.,

$$T^{\lhd} \triangleq \frac{H_T^{\lhd}}{H^{\lhd,1}}, \tag{A.4.66c}$$

$$T^{\cap} \triangleq \frac{H_T^{\cap}}{H^{\cap,1}}. \tag{A.4.66d}$$

Most of the equations apply for directional as well as for hemispherical irradiation. If no ambiguity arises, we will omit the superscript.

To facilitate the notation, we introduce an index set that contains the index of all

panes. For a window with N panes, we define this index set as

$$\mathbf{N} \triangleq \{1, \ldots, N\}. \tag{A.4.67a}$$

We also introduce an index set that contains certain directional modes of irradiations: it contains for directional irradiation the incidence angles $\phi_i \in \{0^\circ, 10^\circ, \ldots, 90^\circ\}$ and an index for hemispherical irradiation. We define this index set as

$$\mathbf{I} \triangleq \{0^\circ, 10^\circ, \ldots, 90^\circ, \text{hemispherical}\}. \tag{A.4.67b}$$

We also introduce a *front absorbtance* A_f^j and a *back absorbtance* A_b^j for the j-th pane. These terms are defined as the absorbtance that the pane would have if the pane was standing isolated with no other panes present, and if the irradiation was only from the front side or only from the back side, respectively. Front and back absorbtance are equal for uncoated glass, but in general different for coated glass.

In Fig. A.8, $T^{i,j}$ denotes the total transmittance through the panes i and j, and through all panes in between, $R^{i,j}$ denotes the total reflectance from pane i to j as if they were standing alone. Reflection is denoted by a subscript "f" if the irradiation is from the front side (i.e., the outside) and by a subscript "b" if the irradiation is from the back side (i.e., the room-side).

Short wave radiation can strike the window from the outside (e.g., solar radiation) and from the inside (e.g., solar radiation reflected in the room and lighting). Thus, equation (A.4.66) depends on whether the irradiation comes from outside or from inside. Properties that correspond to irradiation from outside and from inside will be denoted by $P_{\rightarrow}$ and $P_{\leftarrow}$, respectively. If no ambiguity arises, we will omit the subscript. Note that we do *not* flip the counter for the pane and the subscript for the front and back if we change the radiation source from the outside to the inside, since our notation is based on the geometry defined in Fig. A.8. For example, $R_b^{N,2}$ is the reflectance of a window where the outside pane is removed and the radiation source is placed in the room.

Model Parameter The model uses the following glass properties:

$\{T^{\perp,j,j}\}_{j=1}^{N}$ spectrally averaged transmittance of each pane at normal incidence,

$\{R_f^{\perp,j,j}\}_{j=1}^{N}$ spectrally averaged front reflectance of each pane at normal incidence, and

$\{R_b^{\perp,j,j}\}_{j=1}^{N}$ spectrally averaged back reflectance of each pane at normal incidence.

If an exterior, or alternatively, an interior shading device is present, then its transmittance and reflectance must be specified by the user. For an exterior shading device, τ^1 denotes the total transmittance and ρ^1 denotes the total reflectance. For an interior shading device, the corresponding symbols are τ^N and ρ^N.

Angular Properties of Glass Transmittance and reflectance of glass do not vary much for incidence angles smaller than 50° to 60°, but approach 0 or 1, respectively, as the in-

cidence angle tends to 90°. During most of the time, the incidence angle of the sun is larger than 45°. Therefore, the directional dependency of the glass properties at shallow incidence angles need to be taken into account properly.

To compute the angular dependency of the solar transmittance and the solar front and back reflectance, we use a similar model as the WINDOW 4 program (Finlayson et al., 1993). The model uses the solar transmittance $T^{\perp}$ and the solar front and back reflectance $R_f^{\perp}$ and $R_b^{\perp}$, all at normal incidence angle, of a single pane and computes their directional values using a regression fit. The regression fits were developed for uncoated clear glass and for uncoated bronze glass. However, the regressions are in WINDOW 4 and in our model as well used for coated glass and for uncoated glass. If $T^{\perp} > 0.645$, the fit for clear glass is used, otherwise the fit for bronze glass is used.

The angular variation of the transmittance and the reflectance are

$$\bar{T}^{\sphericalangle}(\phi) = \sum_{k=0}^{4} \bar{T}_k \cos^k(\phi), \tag{A.4.68a}$$

$$\bar{R}^{\sphericalangle}(\phi) = \sum_{k=0}^{4} \bar{R}_k \cos^k(\phi) - \bar{T}^{\sphericalangle}(\phi), \tag{A.4.68b}$$

where $\phi \triangleq 0$ for normal incidence and the coefficients $\{\bar{T}_k\}_{k=0}^4$ and $\{\bar{R}_k\}_{k=0}^4$ are listed in Tab. A.2.

After computing (A.4.68), the angular transmittance, the angular front and the angu-

		index k				
		0	1	2	3	4
If $T^{\perp} > 0.645$:	$\bar{T}_k$	−0.0015	3.355	−3.840	1.460	0.0288
If $T^{\perp} > 0.645$:	$\bar{R}_k$	0.999	−0.563	2.043	−2.532	1.054
If $T^{\perp} \leq 0.645$:	$\bar{T}_k$	−0.002	2.813	−2.341	−0.05725	0.599
If $T^{\perp} \leq 0.645$:	$\bar{R}_k$	0.997	−1.868	6.513	−7.862	3.225

Table A.2: Coefficients used in the regression formula (A.4.68), reproduced from Finlayson et al. (1993).

lar back reflectance are computed as

$$T^{\sphericalangle}(\phi) = \begin{cases} T^{\perp}\bar{T}^{\sphericalangle}_{clr}(\phi), & \text{if } T^{\perp} > 0.645, \\ T^{\perp}\bar{T}^{\sphericalangle}_{bnz}(\phi), & \text{otherwise,} \end{cases} \tag{A.4.69a}$$

$$R^{\sphericalangle}(\phi) = \begin{cases} R^{\perp}\left(1-\bar{R}^{\sphericalangle}_{clr}(\phi)\right)+\bar{R}^{\sphericalangle}_{clr}(\phi), & \text{if } T^{\perp} > 0.645, \\ R^{\perp}\left(1-\bar{R}^{\sphericalangle}_{bnz}(\phi)\right)+\bar{R}^{\sphericalangle}_{bnz}(\phi), & \text{otherwise.} \end{cases} \tag{A.4.69b}$$

The hemispherical values are computed as

$$\begin{aligned} T^{\cap} &= \frac{\int_0^{\pi/2} T^{\sphericalangle}(\phi)\cos(\phi)\sin(\phi)\,d\phi}{\int_0^{\pi/2}\cos(\phi)\sin(\phi)\,d\phi} \\ &= 2\int_0^{\pi/2} T^{\sphericalangle}(\phi)\cos(\phi)\sin(\phi)\,d\phi, \end{aligned} \tag{A.4.70a}$$

$$\begin{aligned} R^{\cap} &= \frac{\int_0^{\pi/2} R^{\sphericalangle}(\phi)\cos(\phi)\sin(\phi)\,d\phi}{\int_0^{\pi/2}\cos(\phi)\sin(\phi)\,d\phi} \\ &= 2\int_0^{\pi/2} R^{\sphericalangle}(\phi)\cos(\phi)\sin(\phi)\,d\phi. \end{aligned} \tag{A.4.70b}$$

The equations (A.4.70) are integrated numerically using the trapezoidal rule. For the

numerical integration, we use the support points $T^{\lhd}(\phi_i)$, $R_f^{\lhd}(\phi_i)$ and $R_b^{\lhd}(\phi_i)$, with $\phi_i \in \{0^\circ, 10^\circ, \ldots, 90^\circ\}$, which we compute using equation (A.4.69).

Specular and Hemispherical Transmittance and Reflectance We will now describe how we obtain the following properties for directional irradiation at incidence angles $\phi_i \in [0^\circ, 90^\circ]$ and for hemispherical irradiation:

1. the total transmittances from outside to inside and from inside to outside for directional and for hemispherical irradiation, i.e., $T^{\lhd,1,N}$, $T^{\lhd,N,1}$, $T^{\cap,1,N}$, $T^{\cap,N,1}$, and

2. the window's front and back reflectances for directional and for hemispherical irradiation, i.e., $R_f^{\lhd,1,N}$, $R_b^{\lhd,N,1}$, $R_f^{\cap,1,N}$, $R_b^{\cap,N,1}$.

We will first develop a model that computes the above properties for radiation striking the window from the outside. To simplify the notation, we will omit the subscript "$\rightarrow$" and the superscript "$\lhd$" or "$\cap$" in the equations (A.4.71) to (A.4.78c) below. The model will be developed for a single pane window, for a double pane window, and for a window with more than two panes. Based on this development, we present equations that are valid for any window with more than one pane. The model takes into account multiple reflections between panes. We will assume that the window consists of infinite large parallel panes, and hence, we will neglect frame effects. How to take into account an exterior or an interior shading device will be presented later.

Single Pane Window For single pane windows, we have $N = 1$ and hence

$$T^{1,N} = T^{1,1}, \quad R_f^{1,N} = R_f^{1,1}, \quad R_b^{N,1} = R_b^{1,1}. \tag{A.4.71}$$

Double Pane Window We will first show how to obtain the transmittance and then the reflectance. Let H^1 denote the irradiation on the front of pane number 1, and let H^2 denote the irradiation on the front of pane number 2. Then, we have

$$H^1 T^{1,N} = H^N T^{N,N}, \tag{A.4.72}$$

where $T^{1,N}$ and H^N are unknown. H^N is the sum of the radiation transmitted through pane 1, which is equal to $H^1 T^{1,1}$, and the fraction of it which underwent multiple reflections between pane 1 and pane N. By using the geometric series

$$\sum_{k=0}^{\infty} a x^k = \frac{a}{1-x}, \qquad \text{for } |x| < 1, \quad a \in \mathbb{R}, \tag{A.4.73}$$

we can write

$$\begin{aligned} H^N &= H^1 T^{1,1} + H^1 T^{1,1} (R_f^{N,N} R_b^{1,1}) + \dots \\ &= H^1 T^{1,1} \sum_{k=0}^{\infty} (R_f^{N,N} R_b^{1,1})^k = \frac{H^1 T^{1,1}}{1 - R_f^{N,N} R_b^{1,1}}. \end{aligned} \tag{A.4.74a}$$

Substituting (A.4.74a) in (A.4.72) and solving for $T^{1,N}$ yields

$$T^{1,N} = \frac{T^{1,1}\,T^{N,N}}{1 - R_f^{N,N}\,R_b^{1,1}}. \tag{A.4.74b}$$

The reflectance $R_f^{1,N}$ is defined as the ratio between the radiation that is reflected and leaves the window to the outside (at the front surface) and the irradiation H^1. It is composed of the front reflectance at pane number 1 and the part of the radiation that has been transmitted through pane number 1, underwent multiple reflections between pane number N and pane number 1 and then has been transmitted again through pane number 1. Thus,

$$\begin{aligned} H^1 R_f^{1,N} &= H^1 R_f^{1,1} + H^1 T^{1,1} R_f^{N,N} T^{1,1} + H^1 T^{1,1} R_f^{N,N} \left(R_b^{1,1} R_f^{N,N}\right) T^{1,1} \\ &\quad + H^1 T^{1,1} R_f^{N,N} \left(R_b^{1,1} R_f^{N,N}\right)^2 T^{1,1} + \ldots \\ &= H^1 \left(R_f^{1,1} + \left(T^{1,1}\right)^2 R_f^{N,N} \sum_{k=0}^{\infty} \left(R_b^{1,1} R_f^{N,N}\right)^k \right) \\ &= H^1 \left(R_f^{1,1} + \frac{\left(T^{1,1}\right)^2 R_f^{N,N}}{1 - R_b^{1,1} R_f^{N,N}} \right), \end{aligned} \tag{A.4.75a}$$

or after dividing by H_1,

$$R_f^{1,N} = R_f^{1,1} + \frac{\left(T^{1,1}\right)^2 R_f^{N,N}}{1 - R_f^{N,N} R_b^{1,1}}. \tag{A.4.75b}$$

The back reflectance, i.e., the reflectance if the irradiation source is inside the building, is obtained from geometric considerations by changing in (A.4.75b) the subscript "*b*"

with "f" (and vice versa) and the index "1" with "N" (and vice versa). Thus,

$$R_b^{N,1} = R_b^{N,N} + \frac{\left(T^{N,N}\right)^2 R_b^{1,1}}{1 - R_f^{N,N} R_b^{1,1}}. \tag{A.4.75c}$$

Window with Three or More Panes For a window with more than two panes, we treat the first $(N-1)$ panes as if they were *one* pane with properties $T^{1,N-1}$, $R_f^{1,N-1}$ and $R_b^{N-1,1}$. Thus, we obtain

$$T^{1,N} = \frac{T^{1,N-1}\, T^{N,N}}{1 - R_f^{N,N} R_b^{N-1,1}}, \tag{A.4.76a}$$

$$R_f^{1,N} = R_f^{1,N-1} + \frac{\left(T^{1,N-1}\right)^2 R_f^{N,N}}{1 - R_f^{N,N} R_b^{N-1,1}}, \tag{A.4.76b}$$

$$R_b^{N,1} = R_b^{N,N} + \frac{\left(T^{N,N}\right)^2 R_b^{N-1,1}}{1 - R_f^{N,N} R_b^{N-1,1}}. \tag{A.4.76c}$$

Note that the equations (A.4.76) are similar to the equations for double pane windows. Furthermore, we can replace in (A.4.76) the superscript "1" with any $i \in \{1, \ldots, (N-1)\}$. Therefore we can state the following recursive computation procedure for windows with $N > 1$.

General Computation Procedure for Windows with more than one Pane We will describe the computation procedure by two indexed loops. The properties $T^{i,j}$, $R_f^{i,j}$ and $R_b^{i,j}$ are obtained using an outer loop with loop counter $i = 1, \ldots, (N-1)$ and an inner loop with loop counter $j = (i+1), \ldots, N$. Inside the loops, we do the following

computations:

First, we compute the denominator of (A.4.76), which is

$$a^{i,j} = 1 - R_f^{j,j} R_b^{j-1,i}, \tag{A.4.77}$$

and then we compute

$$\begin{aligned} T^{i,j} &= \frac{T^{i,j-1} T^{j,j}}{a^{i,j}}, && \text{(A.4.78a)} \\ R_f^{i,j} &= R_f^{i,j-1} + \frac{(T^{i,j-1})^2 R_f^{j,j}}{a^{i,j}}, && \text{(A.4.78b)} \\ R_b^{j,i} &= R_b^{j,j} + \frac{(T^{j,j})^2 R_b^{j-1,i}}{a^{i,j}}. && \text{(A.4.78c)} \end{aligned}$$

This ends the computations inside the loops.

Specular and Hemispherical Absorbtance We will now show how to compute the short-wave absorbtance of each pane for directional irradiation, which we denote by $\{A_{\rightarrow}^{\lhd,j}\}_{j=1}^{N}$ and $\{A_{\leftarrow}^{\lhd,j}\}_{j=1}^{N}$, and for hemispherical irradiation, which we denote by $\{A_{\rightarrow}^{\cap,j}\}_{j=1}^{N}$ and $\{A_{\leftarrow}^{\cap,j}\}_{j=1}^{N}$. The model is based on the results described in (A.4.78).

We will develop the model for the situation where the radiation source is outside the building. We will again omit the superscripts "$\lhd$" and "$\cap$", because the models are identical for both situations.

We first compute the front absorbtance. For single pane windows, they are

$$A^1_{\rightarrow} = 1 - T^{1,1} - R^{1,1}_f, \tag{A.4.79}$$

$$A^1_{\leftarrow} = 1 - T^{1,1} - R^{1,1}_b, \tag{A.4.80}$$

and for multi pane windows, they are, for $j \in \mathbf{N}$,

$$A^j_f = 1 - T^{j,j} - R^{j,j}_f, \tag{A.4.81a}$$

$$A^j_b = 1 - T^{j,j} - R^{j,j}_b. \tag{A.4.81b}$$

Next, to facilitate the formulation, we introduce the auxiliary variables

$$T^{1,0} = 1, \quad R^{0,1}_b = 0, \quad R^{N+1,N}_f = 0. \tag{A.4.82}$$

Let H^1 denote the irradiation source, placed outside the building. Because the absorbtance $A^j_{\rightarrow}$ is composed of the back and the front absorbtance of the j-th pane, we

can write for any pane, with $j \in \mathbf{N}$,

$$\begin{aligned} H^1 A^j_{\rightarrow} &= H^1 \left(T^{1,j-1} + T^{1,j-1} \left(R_f^{j,N} R_b^{j-1,1} \right) + T^{1,j-1} \left(R_f^{j,N} R_b^{j-1,1} \right)^2 + \dots \right) A_f^j \\ &\quad + H^1 \left(T^{1,j} R_f^{j+1,N} + T^{1,j} R_f^{j+1,N} \left(R_b^{j,1} R_f^{j+1,N} \right) + \dots \right) A_b^j \\ &= H^1 \left(T^{1,j-1} \sum_{k=0}^{\infty} \left(R_f^{j,N} R_b^{j-1,1} \right)^k \right) A_f^j \\ &\quad + H^1 \left(T^{1,j} R_f^{j+1,N} \sum_{k=0}^{\infty} \left(R_b^{j,1} R_f^{j+1,N} \right)^k \right) A_b^j. \end{aligned} \tag{A.4.83a}$$

Using the geometric series introduced in (A.4.73) and dividing by H^1 yields

$$A^j_{\rightarrow} = \frac{T^{1,j-1}}{1 - R_f^{j,N} R_b^{j-1,1}} A_f^j + \frac{T^{1,j} R_f^{j+1,N}}{1 - R_b^{j,1} R_f^{j+1,N}} A_b^j. \tag{A.4.83b}$$

If either of the denominator in (A.4.83b) is zero, then no radiation reaches the pane j and hence we set $A^j_{\rightarrow} = 0$.

Next, we compute (A.4.77) to (A.4.78c) and (A.4.83b) for the situation where the radiation source is inside the building. This yields the remaining terms of (A.4.78), with $j < i$, and the absorbtances $\{A^{\cap,j}_{\leftarrow}\}_{j=1}^N$. Since we assume the short-wave radiation from the room to be diffuse, we compute directional and hemispherical values for (A.4.78), but only hemispherical values for (A.4.83b).

To facilitate the calculation, we will introduce dummy variables, indicated by a tilde, i.e., we will write "$\widetilde{P}$" instead of "P", that can be used in equations (A.4.77), (A.4.78),

(A.4.81), (A.4.82) and (A.4.83b). In particular, we will define the dummy variables, for all $\widehat{i} \in \mathbf{I}$,

$$\begin{aligned} \{\widetilde{T}^{\widehat{i},j,j}\}_{j=1}^{N} &= \{T^{\widehat{i},j,j}\}_{j=N}^{1}, \\ \{\widetilde{R}_f^{\widehat{i},j,j}\}_{j=1}^{N} &= \{R_b^{\widehat{i},j,j}\}_{j=N}^{1}, \quad \{\widetilde{R}_b^{\widehat{i},j,j}\}_{j=1}^{N} = \{R_f^{\widehat{i},j,j}\}_{j=N}^{1}. \end{aligned} \tag{A.4.84}$$

Then, we compute (A.4.77), (A.4.78), (A.4.81), (A.4.82) and (A.4.83b), again with the loop counter counting upward (as indicated) whereas in all equations every property is replaced with its dummy variable as defined in (A.4.84). This yields, for all $\widehat{i} \in \mathbf{I}$, with outer loop $i = 1, \ldots, (N-1)$ and with inner loop $j = (i+1), \ldots, N$, the transmittances

$$T^{\widehat{i},N+1-i,N+1-j} = \widetilde{T}^{\widehat{i},i,j}, \tag{A.4.85a}$$

and the absorbtances

$$\{A_{\leftarrow}^{\widehat{i},j}\}_{j=N}^{1} = \{\widetilde{A}_{\rightarrow}^{\widehat{i},j}\}_{j=1}^{N}. \tag{A.4.86a}$$

Exterior and Interior Shading Device The models presented above are all for windows with no interior or exterior shading device.

We will now develop expressions that correct the values obtained by the above models so that either an interior or an exterior shading device is taken into account. We will assume that the shading devices are diffuse reflectors, that only one shading device is

active at any time and that the irradiation from the room side is diffuse. We will develop corrections for the absorbtances $A^{\cap,j}_{\leftarrow}$ and $A^{i,j}_{\rightarrow}$, and for the transmittances $T^{i,1,N}$ and $T^{\cap,N,1}$, with $i \in \mathbf{I}$ and $j \in \mathbf{N}$. We will also develop an equation to compute what radiation is absorbed by the room-side shading device, because this absorbed energy will be released by convection and by infrared radiation to the room.

Since all equations for directional irradiation are also valid for hemispherical irradiation, we will develop the model for directional irradiation only. The expressions for hemispherical irradiation can be obtained by replacing the superscript "$\lhd$" with the superscript "$\cap$".

First, we will assume that the radiation source is on the outside.

Exterior Shading Device, Irradiation from Outside Exterior shading devices reduce the irradiation that strikes the surface number 1. Since the energy transmitted through the panes and the energy absorbed by each pane is proportional to the transmittance or the absorbtance, respectively, and proportional to the irradiation, we will reduce $T^{1,N}$ and $\{A^{j}_{\rightarrow}\}_{j=1}^{N}$ instead of reducing the incoming radiation.

Let ρ^1 denote the reflectance and let τ^1 denote the transmittance of the exterior shading device. Let $H_0^{\lhd}$ denote the irradiation outside the exterior shading device, and let $H^{\lhd,1}$ denote the irradiation on the pane number 1 (see Fig. A.8). The irradiation $H^{\lhd,1}$ is composed of the fraction of $H_0^{\lhd}$ that has been transmitted through the shading device

(and has not yet been reflected), and the radiation that underwent multiple reflections between the window and the shading device. We will assume that the reflection from the shading device is diffuse. However, the irradiation $(H_0^{\triangleleft}\tau^1)$ is directional. Thus, we have

$$\begin{aligned} H^{\triangleleft,1} &= H_0^{\triangleleft}\tau^1 + H_0^{\triangleleft}\tau^1\left(R_f^{\triangleleft,1,N}\rho^1\right) + H_0^{\triangleleft}\tau^1\left(R_f^{\triangleleft,1,N}\rho^1\right)\left(R_f^{\cap,1,N}\rho^1\right) \\ &\quad + H_0^{\triangleleft}\tau^1\left(R_f^{\triangleleft,1,N}\rho^1\right)\left(R_f^{\cap,1,N}\rho^1\right)^2 + \ldots \\ &= H_0^{\triangleleft}\tau^1 + H_0^{\triangleleft}\tau^1\left(R_f^{\triangleleft,1,N}\rho^1\right)\sum_{k=0}^{\infty}\left(R_f^{\cap,1,N}\rho^1\right)^k \\ &= H_0^{\triangleleft}\tau^1 + H_0^{\triangleleft}\tau^1\frac{R_f^{\triangleleft,1,N}\rho^1}{1-R_f^{\cap,1,N}\rho^1}. \end{aligned} \tag{A.4.87}$$

Equation (A.4.87) shows that the irradiation that strikes the pane number 1 is attenuated by a factor

$$c^{\triangleleft,1} \triangleq \tau^1\left(1+\frac{R_f^{\triangleleft,1,N}\rho^1}{1-R_f^{\cap,1,N}\rho^1}\right). \tag{A.4.88a}$$

Therefore, the absorbtance of each pane $j \in \mathbf{N}$ is, if the window has an exterior shading device,

$$A_{\tau^1,\rightarrow}^{\triangleleft,j} = c^{\triangleleft,1}A_{\rightarrow}^{\triangleleft,j}. \tag{A.4.88b}$$

Since the energy transmitted through the overall construction is also proportional to the irradiation, we have for the window transmittance with exterior shading device

$$T^{\triangleleft,1-\tau,N} = c^{\triangleleft,1}T^{\triangleleft,1,N}. \tag{A.4.88c}$$

The equations (A.4.88) are also valid for hemispherical irradiation.

Interior Shading Device, Irradiation from Outside We will now develop an equation that accounts for the increase in absorbed radiation of each pane due to the radiation that is reflected at an interior shading device. We will also develop an equation for the radiation that is absorbed by the shading device, and corrections for the transmittance of the window including shading device. We will assume that the shading device is a diffuse reflector with reflectance ρ^N and transmittance τ^N.

Let $\Delta H^j_{\rightarrow,f}$ denote the increase in irradiation at the front side of the j-th pane due to the radiation that is reflected at the interior shading device. We will introduce the dummy variables $R_b^{\cap,0,1} = 0$. Then, for $j \in \mathbf{N}$,

$$\begin{aligned} \Delta H^j_{\rightarrow,f} &= H^1 \, T^{\triangleleft,1,N} \rho^N \sum_{k=0}^{\infty} \left(R_b^{\cap,N,1} \rho^N \right)^k T^{\cap,N,j} \, R_b^{\cap,j-1,1} \\ &= H^1 \, \frac{T^{\triangleleft,1,N} \rho^N \, T^{\cap,N,j} \, R_b^{\cap,j-1,1}}{1 - R_b^{\cap,N,1} \rho^N}. \end{aligned} \tag{A.4.89a}$$

Similarly, by introducing the dummy variable $T^{\cap,N,N+1} = 1$, the increase in irradiation at the back surface can be described as, with $j \in \mathbf{N}$,

$$\begin{aligned} \Delta H^j_{\rightarrow,b} &= H^1 \, T^{\triangleleft,1,N} \rho^N \sum_{k=0}^{\infty} \left(R_b^{\cap,N,1} \rho^N \right)^k T^{\cap,N,j+1} \\ &= H^1 \, \frac{T^{\triangleleft,1,N} \rho^N \, T^{\cap,N,j+1}}{1 - R_b^{\cap,N,1} \rho^N}. \end{aligned} \tag{A.4.89b}$$

By assumption, the shading device reflects diffusely, and hence $\Delta H^{j}_{\rightarrow,f}$ and $\Delta H^{j}_{\rightarrow,b}$ are diffuse irradiations. Therefore, for $j \in \mathbf{N}$, with $R_b^{\cap,0,1} = 0$ and $T^{\cap,N,N+1} = 1$, the directional absorbtances for the panes are, if the window has an interior shading device,

$$A^{\lhd,j}_{\tau^N,\rightarrow} = A^{\lhd,j}_{\rightarrow} + \frac{T^{\lhd,1,N}\,\rho^N\,T^{\cap,N,j}\,R_b^{\cap,j-1,1}}{1-R_b^{\cap,N,1}\,\rho^N}A_f^{\cap,j} + \frac{T^{\lhd,1,N}\,\rho^N\,T^{\cap,N,j+1}}{1-R_b^{\cap,N,1}\,\rho^N}A_b^{\cap,j}, \qquad \text{(A.4.90)}$$

where $A_f^{\cap,j}$ and $A_b^{\cap,j}$ are defined in (A.4.81).

Let $A^{\lhd,\tau^N}_{\rightarrow}$ denote the fraction of the outside irradiation H_1 that is absorbed by the interior shading device. It can be obtained from

$$\begin{aligned} H^1 A^{\lhd,\tau^N}_{\rightarrow} &= H^1\,T^{\lhd,1,N}\sum_{k=0}^{\infty}\left(\rho^N R_b^{\cap,N,1}\right)^k\left(1-\rho^N-\tau^N\right) \\ &= H^1\,\frac{T^{\lhd,1,N}}{1-\rho^N R_b^{\cap,N,1}}\left(1-\rho^N-\tau^N\right). \end{aligned} \qquad \text{(A.4.91a)}$$

Thus, after dividing by H^1, we have

$$A^{\lhd,\tau^N}_{\rightarrow} = \frac{T^{\lhd,1,N}}{1-\rho^N R_b^{\cap,N,1}}\left(1-\rho^N-\tau^N\right). \qquad \text{(A.4.91b)}$$

Let $H^{\lhd,1,N+\tau}$ denote the directional transmittance through all panes and through the

shading device. We have

$$H^1 H^{\triangleleft,1,N+\tau} = H^1 \frac{T^{\triangleleft,1,N}}{1-\rho^N R_b^{\cap,N,1}} \tau^N. \tag{A.4.92a}$$

After dividing by H^1, we obtain the overall transmittance of the window and the shading device as

$$T^{\triangleleft,1,N+\tau} = \frac{T^{\triangleleft,1,N}}{1-\rho^N R_b^{\cap,N,1}} \tau^N. \tag{A.4.92b}$$

Exterior Shading Device, Irradiation from Inside We will now assume that a diffuse radiation source is inside the room. Let H^N denote the irradiation at the pane N from the room side. We define the dummy variables $T^{\cap,1,0} = 1$ and $R_f^{\cap,N+1,N} = 0$. Then, similarly as in (A.4.89), for $j \in \mathbf{N}$, the increase in irradiation at the pane front and back surface is

$$\begin{aligned} \Delta H_{\leftarrow,f}^{\cap,j} &= H^N T^{\cap,N,1} \rho^1 \sum_{k=0}^{\infty} \left(R_f^{\cap,1,N} \rho^1 \right)^k T^{\cap,1,j-1} \\ &= H^N \frac{T^{\cap,N,1} \rho^1 T^{\cap,1,j-1}}{1-R_f^{\cap,1,N} \rho^1}, \end{aligned} \tag{A.4.93a}$$

$$\begin{aligned} \Delta H_{\leftarrow,b}^{\cap,j} &= H^N T^{\cap,N,1} \rho^1 \sum_{k=0}^{\infty} \left(R_f^{\cap,1,N} \rho^1 \right)^k T^{\cap,1,j} R_f^{\cap,j+1,N} \\ &= H^N \frac{T^{\cap,N,1} \rho^1 T^{\cap,1,j} R_f^{\cap,j+1,N}}{1-R_f^{\cap,1,N} \rho^1}. \end{aligned} \tag{A.4.93b}$$

Therefore, for $j \in \mathbf{N}$, we can obtain for windows with exterior shading device the absorbtances as

$$A^{\cap,j}_{\tau^1,\leftarrow} = A^{\cap,j}_{\leftarrow} + \frac{T^{\cap,N,1}\,\rho^1\,T^{\cap,1,j-1}}{1-R^{\cap,1,N}_f\,\rho^1}A^{\cap,j}_f + \frac{T^{\cap,N,1}\,\rho^1\,T^{\cap,1,j}\,R^{\cap,j+1,N}_f}{1-R^{\cap,1,N}_f\,\rho^1}A^{\cap,j}_b, \qquad \text{(A.4.94)}$$

where $A^{\cap,j}_f$ and $A^{\cap,j}_b$ are as in (A.4.81).

The transmittance through the window and shading device is

$$T^{\cap,N,1-\tau} = T^{\cap,N,1}\sum_{k=0}^{\infty}\left(\rho^1 R^{\cap,1,N}_f\right)^k \tau^1 = \frac{T^{\cap,N,1}\,\tau^1}{1-\rho^1 R^{\cap,1,N}_f}. \qquad \text{(A.4.95)}$$

To obtain the distribution of the short-wave radiation in the room enclosure, we need to know the reflectance of the window with shading device. It is obtained from the radiation balance

$$H^N R^{\cap,N,1-\tau}_b = H^N R^{\cap,N,1}_b + H^N T^{\cap,N,1}\,\rho^1 \sum_{k=0}^{\infty}\left(R^{\cap,1,N}_f\,\rho^1\right)^k T^{\cap,1,N}. \qquad \text{(A.4.96)}$$

Thus,

$$R^{\cap,N,1-\tau}_b = R^{\cap,N,1}_b + \frac{T^{\cap,N,1}\,\rho^1\,T^{\cap,1,N}}{1-R^{\cap,1,N}_f\,\rho^1}. \qquad \text{(A.4.97)}$$

Interior Shading Device, Irradiation from Inside To obtain the absorbtance of each pane and the overall transmittance, we will compute how much the irradiation on

pane N is attenuated due to the interior shading device. Let $H^{\cap,r}$ denote the irradiation on the room side facing surface of the shading device, and let $H^{\cap,N}$ denote the room-side irradiation on the pane N. Then, we have

$$H^{\cap,N} = H^{\cap,r}\,\tau^N \left(1 + R_b^{\cap,N,1}\,\rho^N \sum_{k=0}^{\infty} \left(R_b^{\cap,N,1}\,\rho^N\right)^k\right). \tag{A.4.98}$$

Thus, the irradiation is attenuated by a factor

$$c^{\cap,N} = \tau^N \left(1 + \frac{R_b^{\cap,N,1}\,\rho^N}{1 - R_b^{\cap,N,1}\,\rho^N}\right). \tag{A.4.99}$$

Hence, the absorbtance of each pane and the overall transmittance of the window and the shading device is, for $j \in \mathbf{N}$,

$$A_{\tau^N,\leftarrow}^{\cap,j} = c^{\cap,N} A_{\leftarrow}^{\cap,j}, \tag{A.4.100a}$$

$$T^{\cap,N+\tau,1} = c^{\cap,N}\,T^{\cap,N,1}. \tag{A.4.100b}$$

For the thermal heat balance of the zone, we also need to know what part of $H^{\cap,r}$ is absorbed by the shading device. Let q_{abs} denote the radiation that is absorbed by the shading device for a given room-side irradiation $H^{\cap,r}$. We define the absorbtance of the shading device for room-side irradiation by

$$A_{\leftarrow}^{\tau^N} \triangleq \frac{q_{abs}}{H^{\cap,r}}. \tag{A.4.101}$$

We can obtain $A_{\leftarrow}^{\tau^N}$ from

$$\begin{aligned} H^{\cap,r} A_{\leftarrow}^{\tau^N} &= H^{\cap,r}\left(1-\tau^N-\rho^N\right) \\ &\quad + H^{\cap,r}\tau^N R_b^{\cap,N,1} \sum_{k=0}^{\infty}\left(R_b^{\cap,N,1}\rho^N\right)^k \left(1-\tau^N-\rho^N\right). \end{aligned} \tag{A.4.102}$$

Thus,

$$A_{\leftarrow}^{\tau^N} = \left(1-\tau^N-\rho^N\right)\left(1+\frac{\tau^N R_b^{\cap,N,1}}{1-R_b^{\cap,N,1}\rho^N}\right). \tag{A.4.103}$$

We also need to know the back reflectance of the window with interior shading device, which we denote by $R_b^{\cap,N+\tau,1}$. Recalling that $c^{\cap,N}$, defined in (A.4.99), already takes multiple reflections between the shading device and the panes into account, we can write

$$H^{\cap,r} R_b^{\cap,N+\tau,1} = H^{\cap,r}\rho^N + c^{\cap,N} H^{\cap,r} R_b^{\cap,N,1}\tau^N. \tag{A.4.104}$$

Hence,

$$R_b^{\cap,N+\tau,1} = \rho^N + c^{\cap,N} R_b^{\cap,N,1}\tau^N. \tag{A.4.105}$$

Module Description

Parameter

Variable	Description
N	number of panes
$\{T^{\perp,j,j}\}_{j\in\mathbf{N}}$	solar transmittance of panes at normal incidence
$\{R_f^{\perp,j}\}_{j\in\mathbf{N}}$	front solar reflectance of panes at normal incidence
$\{R_b^{\perp,j}\}_{j\in\mathbf{N}}$	back solar reflectance of panes at normal incidence
τ^1	solar transmittance of exterior shading device (or 0 if no shading device is installed)
ρ^1	solar reflectance of exterior shading device (or 0 if no shading device is installed)

Continued on next column.

Parameter (continued)

Variable	Description
τ^N	solar transmittance of interior shading device (or 0 if no shading device is installed)
ρ^N	solar reflectance of interior shading device (or 0 if no shading device is installed)

Output

Variable	Description
$\{A^{i,j}_{\rightarrow}\}_{i\in\mathbf{I},j\in\mathbf{N}}$	absorbtance for irradiation from outside
$\{A^{i,j}_{\leftarrow}\}_{i\in\mathbf{I},j\in\mathbf{N}}$	absorbtance for irradiation from inside
$\{A^{i,j}_{\tau^1,\rightarrow}\}_{i\in\mathbf{I},j\in\mathbf{N}}$	absorbtance for irradiation from outside, exterior shading device
$\{A^{i,j}_{\tau^N,\rightarrow}\}_{i\in\mathbf{I},j\in\mathbf{N}}$	absorbtance for irradiation from outside, interior shading device
$\{T^{i,1,N}\}_{i\in\mathbf{I}}$	transmittance from outside to inside, no shading device
Continued on next column.	

Output (continued)

Variable	Description
$\{T^{i,1-\tau,N}\}_{i\in\mathbf{I}}$	transmittance from outside to inside with exterior shading device
$\{T^{i,1,N+\tau}\}_{i\in\mathbf{I}}$	transmittance from outside to inside with interior shading device
$\{A^{\cap,j}_{\tau^1,\leftarrow}\}_{j\in\mathbf{N}}$	hemispherical absorbtance for irradiation from inside, exterior shading device
$\{A^{\cap,j}_{\tau^N,\leftarrow}\}_{j\in\mathbf{N}}$	hemispherical absorbtance for irradiation from inside, interior shading device
Continued on next page.	

Output (continued)

Variable	Description
$R_b^{\cap,N,1-\tau}$	back reflectance with exterior shading device
$R_b^{\cap,N+\tau,1}$	back reflectance with interior shading device
$A_{\rightarrow}^{\lhd,\tau^N}$	absorbtance of interior shading device for irradiation from outside
$A_{\leftarrow}^{\tau^N}$	absorbtance of interior shading device for irradiation from inside

Algorithm – Angular properties of glass

data $R_f^{\perp}, R_b^{\perp}, T^{\perp}$

if $T^{\perp} > 0.645$ **then**

  select from Tab. A.2: $\{\bar{T}_k\}_{k=0}^{4} = \{\bar{T}_{clr,k}\}_{k=0}^{4}$; $\{\bar{R}_k\}_{k=0}^{4} = \{\bar{R}_{clr,k}\}_{k=0}^{4}$;

else

  select from Tab. A.2: $\{\bar{T}_k\}_{k=0}^{4} = \{\bar{T}_{bnz,k}\}_{k=0}^{4}$; $\{\bar{R}_k\}_{k=0}^{4} = \{\bar{R}_{bnz,k}\}_{k=0}^{4}$;

end if

for all $\phi_i \in \{(i-1)\,\pi/18\}_{i=2}^{9}$ **do**

  $\bar{T}^{\sphericalangle}(\phi_i) \overset{\text{(A.4.68a)}}{\longleftarrow} \{\bar{T}_k\}_{k=0}^{4},\ \phi_i$;

  $\bar{R}^{\sphericalangle}(\phi_i) \overset{\text{(A.4.68b)}}{\longleftarrow} \{\bar{R}_k\}_{k=0}^{4},\ \phi_i$;

  $T^{\sphericalangle}(\phi_i) \overset{\text{(A.4.69a)}}{\longleftarrow} T^{\perp},\ \bar{T}^{\sphericalangle}(\phi_i)$;

  $R_f^{\sphericalangle}(\phi_i) \overset{\text{(A.4.69b)}}{\longleftarrow} R_f^{\perp},\ \bar{R}^{\sphericalangle}(\phi_i)$;

  $R_b^{\sphericalangle}(\phi_i) \overset{\text{(A.4.69b)}}{\longleftarrow} R_b^{\perp},\ \bar{R}^{\sphericalangle}(\phi_i)$;

end for

assign $R_f^{\sphericalangle}(\phi_i = 0) = R_f^{\perp},\ R_b^{\sphericalangle}(\phi_i = 0) = R_b^{\perp},\ T^{\sphericalangle}(\phi_i = 0) = T^{\perp}$;

assign $R_f^{\sphericalangle}(\phi_i = \pi/2) = R_b^{\sphericalangle}(\phi_i = \pi/2) = 1,\ T^{\sphericalangle}(\phi_i = \pi/2) = 0$;

$T^{\cap} \overset{\text{(A.4.70a)}}{\longleftarrow} \{T^{\sphericalangle}(\phi_i)\}_{i=1}^{10}$;

$R_f^{\cap} \overset{\text{(A.4.70b)}}{\longleftarrow} \{R_f^{\sphericalangle}(\phi_i)\}_{i=1}^{10}$;

$R_b^{\cap} \overset{\text{(A.4.70b)}}{\longleftarrow} \{R_b^{\sphericalangle}(\phi_i)\}_{i=1}^{10}$;

output $\{T^i\}_{i\in\mathbf{I}},\ \{R_f^i\}_{i\in\mathbf{I}},\ \{R_b^i\}_{i\in\mathbf{I}}$;

Algorithm – Window properties, no shading device

data $\{\{T^{i,j,j}\}_{j\in\mathbf{N}}\}_{i\in\mathbf{I}}$, $\{\{R_f^{i,j,j}\}_{j\in\mathbf{N}}\}_{i\in\mathbf{I}}$, $\{R_b^{j,j}\}_{j\in\mathbf{N}}$;
if $N = 1$ **then**
  $\{$Single pane window$\}$
  for all $\phi_i \in \mathbf{I}$ **do**
    $T^{1,N}$, $R_f^{1,N}$, $R_b^{N,1}$, $A_{\rightarrow}^{1}$, $A_{\leftarrow}^{1} \overset{(A.4.71)}{\longleftarrow} T^{1,1}$, $R_f^{1,1}$, $R_b^{1,1}$;
    $A_{\rightarrow}^{1}$, $A_{\leftarrow}^{1} \overset{(A.4.80)}{\longleftarrow} T^{1,1}$, $R_f^{1,1}$, $R_b^{1,1}$;
  end for
end if
Continued on next page.

Algorithm – Window properties, no shading device (continued)

if $N > 1$ **then**
  {Multi-pane window}
  for all $\phi_i \in \mathbf{I}$ **do**
    for $i = 1, \ldots, (N-1)$ **do**
      for $j = (i+1), \ldots, N$ **do**
        $a^{i,j} \overset{\text{(A.4.77)}}{\longleftarrow} R_f^{j,j},\ R_b^{j-1,i};$
        if $a^{i,j} = 0$ **then**
          set $T_{i,j} = 0; \quad R_f^{i,j} = R_b^{j,i} = 1;$
        else
          $T^{i,j} \overset{\text{(A.4.78a)}}{\longleftarrow} T^{i,j-1},\ T^{j,j},\ a^{i,j};$
          $R_f^{i,j} \overset{\text{(A.4.78b)}}{\longleftarrow} R_f^{i,j-1},\ R_f^{j,j},\ T^{i,j-1},\ a^{i,j};$
          $R_b^{j,i} \overset{\text{(A.4.78c)}}{\longleftarrow} R_b^{j,j},\ R_b^{j-1,i},\ T^{j,j},\ a^{i,j};$
        end if
      end for
    end for
    introduce (A.4.82) $T^{1,0} = 1, \quad R_b^{0,1} = 0, \quad R_f^{N+1,N} = 0;$
    for all $j \in \mathbf{N}$ **do**
      $A_f^j \overset{\text{(A.4.81a)}}{\longleftarrow} T^{j,j},\ R_f^{j,j};$
      $A_b^j \overset{\text{(A.4.81b)}}{\longleftarrow} T^{j,j},\ R_b^{j,j};$
      $A_{\rightarrow}^j \overset{\text{(A.4.83b)}}{\longleftarrow} T^{1,j-1},\ R_f^{j,N},\ R_b^{j-1,1},\ R_f^{j+1,N},\ R_b^{j,1},\ T^{1,j},\ A_f^j,\ A_b^j;$
      save $A_f^j, A_b^j, A_{\rightarrow}^j;$
    end for
  end for
end if

Continued on next page.

Algorithm – Window properties, no shading device (continued)

35: **for** hemispherical irradiation only **do**

36: assign (A.4.84);

37: execute Step 2 to Step 33, but in Step 30, do not save A_f^j, A_b^j: ;

38: assign (A.4.85) and (A.4.86);

39: **end for**

40: return $\{T^{i,1,N}\}_{i\in\mathbf{I}}$, $\{\{R_f^{i,j}\}_{j\in\mathbf{N}}\}_{i\in\mathbf{I}}$, $\{R_b^{\cap,j}\}_{j\in\mathbf{N}}$, $\{\{A_{\rightarrow}^{i,j}\}_{j\in\mathbf{N}}\}_{i\in\mathbf{I}}$, $\{A_{\leftarrow}^{\cap,j}\}_{j\in\mathbf{N}}$;

Algorithm – Irradiation from Outside, Exterior Shading Device

1: data ρ^1, τ^1, $\{R_f^{i,1,N}\}_{i\in\mathbf{I}}$, $\{\{A^{i,j}_{\rightarrow}\}_{j\in\mathbf{N}}\}_{i\in\mathbf{I}}$, $\{T^{i,1,N}\}_{i\in\mathbf{I}}$;

2: **for all** $\phi_i \in \mathbf{I}$ **do**

3: $c^{i,1} \overset{\text{(A.4.88a)}}{\longleftarrow} \tau^1, \rho^1, R_f^{i,1,N}, R_f^{\cap,1,N}$;

4: **for all** $j \in \mathbf{N}$ **do**

5: $A^{i,j}_{\tau^1,\rightarrow} \overset{\text{(A.4.88b)}}{\longleftarrow} c^{i,1}, A^{i,j}_{\rightarrow}$;

6: **end for**

7: $T^{i,1-\tau,N} \overset{\text{(A.4.88c)}}{\longleftarrow} c^{i,1}, T^{i,1,N}$;

8: **end for**

Algorithm – Irradiation from Outside, Interior Shading Device

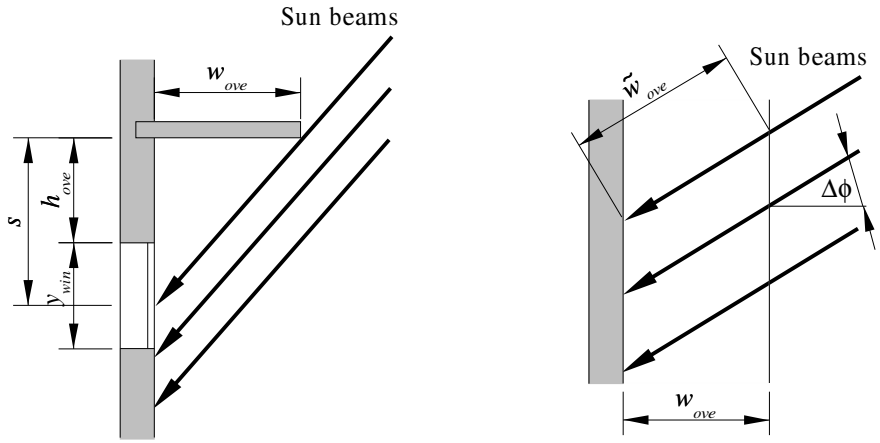
data ρ^N, τ^N, $\{T^{i,1,N}\}_{i\in\mathbf{I}}$, $\{T^{\cap,N,j}\}_{j\in\mathbf{N}}$, $\{R_b^{\cap,j,1}\}_{j\in\mathbf{N}}$,
$\{\{A_{\rightarrow}^{i,j}\}_{j\in\mathbf{N}}\}_{i\in\mathbf{I}}$, $\{A_f^{\cap,j}\}_{j\in\mathbf{N}}$, $\{A_b^{\cap,j}\}_{j\in\mathbf{N}}$;
introduce $R_b^{\cap,0,1}=0$ and $T^{\cap,N,N+1}=1$;
for all $\phi_i \in \mathbf{I}$ **do**
  for all $j \in \mathbf{N}$ **do**
    $A_{\tau^N,\rightarrow}^{i,j} \overset{(A.4.90)}{\longleftarrow} \rho^N,\ T^{i,1,N},\ T^{\cap,N,j+1},\ T^{\cap,N,j},\ R_b^{\cap,j-1,1},\ R_b^{\cap,N,1},\ R_b^{\cap,N,1},\ A_{\rightarrow}^{i,j},$
    $A_f^{\cap,j},\ A_b^{\cap,j}$;
  end for
  $A_{\rightarrow}^{i,\tau^N} \overset{(A.4.91b)}{\longleftarrow} \tau^N,\ \rho^N,\ T^{i,1,N},\ R_b^{\cap,N,1}$;
  $T^{i,1,N+\tau} \overset{(A.4.92b)}{\longleftarrow} \tau^N,\ \rho^N,\ T^{i,1,N},\ R_b^{\cap,N,1}$;
end for

Algorithm – Irradiation from Inside, Exterior Shading Device

data ρ^1, $T^{\cap,N,1}$, $\{T^{\cap,1,j}\}_{j\in\mathbf{N}}$, $\{R_f^{\cap,j,N}\}_{j\in\mathbf{N}}$, $R_b^{\cap,N,1}$,
$\{A_f^{\cap,j}\}_{j\in\mathbf{N}}$, $\{A_b^{\cap,j}\}_{j\in\mathbf{N}}$, $\{A_{\leftarrow}^{\cap,j}\}_{j\in\mathbf{N}}$;
introduce $T^{\cap,1,0}=1$ and $R_f^{\cap,N+1,N}=0$;
for all $j \in \mathbf{N}$ **do**
  $A_{\tau^1,\leftarrow}^{\cap,j} \overset{(A.4.94)}{\longleftarrow} \rho^1,\ A_{\leftarrow}^{\cap,j},\ T^{\cap,N,1},\ T^{\cap,1,j-1},\ T^{\cap,1,j},\ R_f^{\cap,1,N},\ R_f^{\cap,j+1,N},$
  $A_f^{\cap,j},\ A_b^{\cap,j}$;
end for
$R_b^{\cap,N,1-\tau} \overset{(A.4.97)}{\longleftarrow} \rho^1,\ R_f^{\cap,1,N},\ R_b^{\cap,N,1},\ T^{\cap,N,1},\ T^{\cap,1,N}$;

Algorithm – Irradiation from Inside, Interior Shading Device

1: data ρ^N, τ^N, $R_b^{\cap,N,1}$, $\{A_{\leftarrow}^{\cap,j}\}_{j\in\mathbf{N}}$;
2: $c^{\cap,N} \overset{(A.4.99)}{\longleftarrow} \rho^N$, τ^N, $R_b^{\cap,N,1}$;
3: **for all** $j \in \mathbf{N}$ **do**
4: $A_{\tau^N,\leftarrow}^{\cap,j} \overset{(A.4.100a)}{\longleftarrow} c^{\cap,N}$, $A_{\leftarrow}^{\cap,j}$;
5: **end for**
6: $A_{\leftarrow}^{\tau^N} \overset{(A.4.103)}{\longleftarrow} \rho^N$, τ^N, $R_b^{\cap,N,1}$;
7: $R_b^{\cap,N+\tau,1} \overset{(A.4.105)}{\longleftarrow} \rho^N$, τ^N, $c^{\cap,N}$, $R_b^{\cap,N,1}$;



(a) Side view.

(b) Top view.

Figure A.9: Infinite long window overhang.

A.4.3.4 Window Overhang

Introduction We will now present a simple model for window overhangs. The window overhang is assumed to be of infinite length, which allows neglecting end effects. The model is based on the geometric data $y_{win}(x)$, h_{ove} and $w_{ove}(x)$ which are shown in Fig. A.9, and on the wall azimuth $\phi_p(x)$, the solar azimuth $\phi_s(t)$ and the solar zenith angle $\theta_s(t)$, which are described in Section A.4.2.1 on page 141.

Mathematical Description Let $\phi_p(x)$ be the azimuth of a vertical wall, and let $\phi_s(t)$ be the azimuth of the sun, with $\phi_p(x) \triangleq 0$ and $\phi_s(t) \triangleq 0$ due south. Then, the solar azimuth is, relative to the wall azimuth,

$$\Delta\phi(x,t) = \phi_s(t) - \phi_p(x). \tag{A.4.106}$$

Let w_{ove} denote the width of the overhang as shown in Fig. A.9. The horizontal component $\widetilde{w}_{ove}$ of the distance which the sun beam travels below the overhang before it hits the wall is

$$\widehat{w}_{ove}(x,t) = \frac{1}{\cos\Delta\phi(x,t)}\, w_{ove}(x). \tag{A.4.107a}$$

When the window is exposed to direct radiation, we have $\Delta\phi(x,t) \in [-90^\circ, +90^\circ]$. Since (A.4.107a) is not defined for $\Delta\phi(x,t) \in \{-90^\circ, +90^\circ\}$, we rewrite it as

$$\widetilde{w}_{ove}(x,t) = \frac{1}{\cos\Big(\widetilde{\min}\big\{89^\circ, \widetilde{\max}\{-89^\circ, \Delta\phi(x,t)\}\big\}\Big)}\, w_{ove}(x). \tag{A.4.107b}$$

The distance s in Fig. A.9 is

$$\widehat{s}(x,t) = \frac{1}{\tan\theta_s(t)}\, \widetilde{w}_{ove}(x,t). \tag{A.4.108a}$$

The solar azimuth $\theta_s(t)$ can vary between 0° (in the tropics) and $+90^\circ$ at sun set. For $\theta_s(t) \in \{0^\circ, 90^\circ\}$, equation (A.4.108a) is not defined, because of a division by zero or because $\tan 90^\circ$ is infinity. Therefore, we rewrite (A.4.108a) as

$$s(x,t) = \frac{1}{\tan\Big(\min\big\{89^\circ, \max\{1^\circ, \theta_s(t)\}\big\}\Big)}\, \widetilde{w}_{ove}(x,t). \tag{A.4.108b}$$

By using (A.4.107b) and (A.4.108b), we obtain the ratio of the window that is shaded

by the overhang as

$$\widehat{r}(x,t) = \frac{s(x,t) - h_{ove}}{y_{win}(x)}. \tag{A.4.109a}$$

Equation (A.4.109a) holds only for $s(x,t) \in [h_{ove}, h_{ove} + y_{win}(x)]$. Thus, we rewrite it in the form

$$r(x,t) = \widetilde{\min}\left\{1, \widetilde{\max}\left\{0, \frac{s(x,t) - h_{ove}}{y_{win}(x)}\right\}\right\}. \tag{A.4.109b}$$

Equation (A.4.109b) is once Lipschitz continuously differentiable, and tends to zero if the window is completely in the shadow, and to one if the window is completely in the sun.

Module Description

Parameter

Variable	Description
ϕ_p	wall azimuth
h_{ove}	vertical distance from top of window to overhang
$y_{win}(x)$	window height
$w_{ove}(x)$	overhang width

Input

Variable	Description
$\phi_s(t)$	solar azimuth

Output

Variable	Description
$r(x,t)$	ratio of window that is shaded by the overhang

Algorithm

1: $\Delta\phi(x,t) \overset{\text{(A.4.106)}}{\longleftarrow} \phi_s(t),\ \phi_p(x);$

2: $\widetilde{w}_{ove}(x,t) \overset{\text{(A.4.107b)}}{\longleftarrow} \Delta\phi(x,t);$

3: $s(x,t) \overset{\text{(A.4.108b)}}{\longleftarrow} \theta_s(t),\ \widetilde{w}_{ove}(x,t);$

4: $r(x,t) \overset{\text{(A.4.109b)}}{\longleftarrow} s(x,t),\ h_{ove},\ y_{win}(x);$

A.4.3.5 Window Heat Balance Equations

We will first present the heat balance equations for the radiative, conductive and convective heat transfer in the window which are nonlinear functions of the window pane temperature. Next, we will simplify and linearize the nonlinear system of equations and present its solution using a vector-matrix formulation.

As we did for the optical window model, we use the set $\mathbf{N} \triangleq \{1, 2, \ldots, N\}$ to denote the index of the window pane and the set $\mathbf{I} \triangleq \{0^\circ, 10^\circ, \ldots, 90^\circ, \text{hemispherical}\}$ to denote the different modes of irradiation. First, we will assume that the window is not shaded by a window overhang, and that the shading device is either fully activated or fully deactivated. How to take partially shaded windows into account is presented in Section A.4.3.7 on page 238.

The directional total absorbtances $\{A_{\rightarrow}^{\lhd,j}(x)\}_{j\in\mathbf{N}}$ and the hemispherical total absorbtances $\{A_{\rightarrow}^{\cap,j}(x), A_{\leftarrow}^{\cap,j}(x)\}_{j\in\mathbf{N}}$ are obtained from Section A.4.3.3. Similarly, the directional transmittances of the whole window construction for irradiation from outside, i.e., $T^{\lhd,1-\tau,N}(x)$, $T^{\lhd,1,N}$ and $T^{\lhd,1,N+\tau}(x)$, and their hemispherical values $T^{\cap,1-\tau,N}(x)$, $T^{\cap,1,N}$ and $T^{\cap,1,N+\tau}(x)$ are also obtained from Section A.4.3.3.[7]

In Section A.4.3.3, we determined the directional values only for the incidence angles

[7] The absorbtances and transmittances depend on x if the shading device transmittance is a component of the design parameter.

$\phi \in \{0^\circ,\ 10^\circ,\ \ldots\ ,\ 90^\circ\}$, but not for angles in between. Hence, we will convert these discrete values to a 5-th order polynomial of the form

$$P_{a_\chi}\Big(\cos\big(\phi(x,t),x\big)\Big) = \sum_{j=0}^{5} a_{\chi(x),j}(x)\,\cos^j\phi(x,t), \quad \phi \in [0,\pi/2], \tag{A.4.110a}$$

where the symbol $\chi(x)$ stands for either $\{A^{\triangleleft,i}_{\rightarrow}(x)\}_{i\in\mathbf{N}}$, $\{A^{\triangleleft,i}_{\leftarrow}(x)\}_{i\in\mathbf{N}}$, $\{A^{\triangleleft,i}_{\rightarrow}(x)\}_{i\in\mathbf{N}}$, $\{A^{\triangleleft,i}_{\tau^1,\rightarrow}(x)\}_{i\in\mathbf{N}}$, $\{A^{\triangleleft,i}_{\tau^N,\rightarrow}(x)\}_{i\in\mathbf{N}}$, or for $T^{\triangleleft,1-\tau,N}(x)$, $T^{\triangleleft,1,N}$, $T^{\triangleleft,1,N+\tau}(x)$, respectively. The coefficients $\{a_{\chi(x),j}(x)\}_{j=1}^{6}$ are determined with least square curve fitting. Equation (A.4.110a) is used to compute the absorbtance of each pane surface $\{S^i(x,t)\}_{i\in\mathbf{N}}$, which will be used in (A.4.111a), (A.4.112a) and (A.4.113a) below. Even though we do not need the transmittance values for the window heat balance, we will compute their polynomial in this model here since they will be used for the room heat balance.

We will now formulate the steady-state energy balance at each pane surface. The short-wave and the infrared radiation emitted or reflected by the room are assumed to be diffuse. Each pane absorbs part of the room's *short-wave* radiation. All panes are assumed to be opaque for infrared radiation, and the radiation exchange between the panes is assumed to be gray and diffuse. Using the variables and the numeration scheme

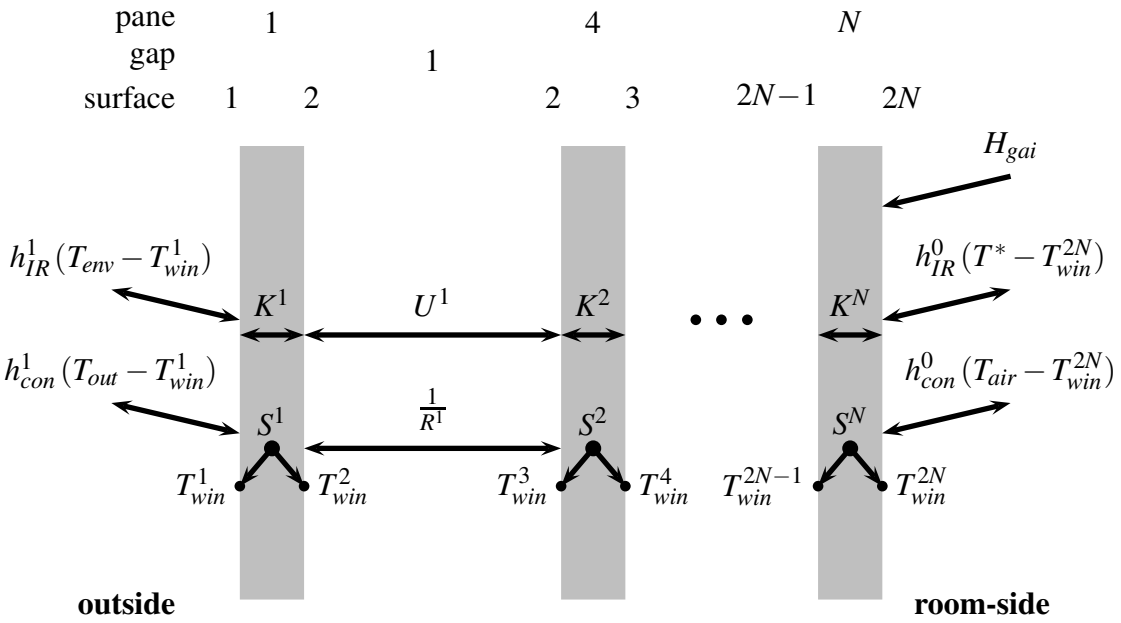


Figure A.10: Nomenclature for heat balance of a window with N panes ($N > 1$).

as shown in Fig. A.10, we can write for the outside facing surface

$$\begin{aligned} 0 &= K^1\left(T^2_{win}(x,t) - T^1_{win}(x,t)\right) + h^1_{con}(x,t)\left(T_{out}(t) - T^1_{win}(x,t)\right) \\ &\quad + h^1_{IR}(t)\left(T_{env}(t) - T^1_{win}(x,t)\right) + \frac{S^1(x,t)}{2}, \end{aligned} \tag{A.4.111a}$$

where K^1 is the conductance through the pane from surface number 1 to 2, $h^1_{con}(\cdot,\cdot)$ is described in Section A.4.3.1, $h^1_{IR}(\cdot,\cdot)$ is as in (A.4.44a), and $T_{env}(\cdot,\cdot)$ is as in (A.4.44b). $S^1(x,t)$ is the absorbed part of the room's *short wave* radiation from zone lights, from the solar radiation that is reflected by the room and emitted toward the window, and from the exterior solar radiation.

The heat conductance through the i-th pane is

$$K^i = \frac{k^i_{glass}}{l^i}, \qquad i \in \{1,\ldots,N\}, \tag{A.4.111b}$$

where k^i_{glass} denotes the thermal conductivity and l^i denotes the thickness of the pane. For short-wave irradiation, the absorbtance of all panes are computed for all solar incidence angles $\phi(x,t)$ as

$$\begin{aligned} S^i(x,t) &= H^1_{dir,til}\big(\phi(x,t)\big)\, P_{A^{\triangleleft,i}_{\rightarrow}}\big(\phi(x,t),x\big) + H^1_{dif,til}\big(\phi(x,t)\big) A^{\cap,i}_{\rightarrow}(x) \\ &\quad + H^0_{sw}(x,t)\, A^{\cap,i}_{\leftarrow}(x), \end{aligned} \tag{A.4.111c}$$

where $i \in \mathbf{N}$. In (A.4.111c), the irradiation $H^1_{dir,til}(\cdot)$ and $H^1_{dif,til}(\cdot)$ are obtained from Section A.4.2.2, and $P(\cdot)$ are the polynomials defined in (A.4.110a). The subscript "*sw*" denotes short-wave radiation.

We will now formulate the heat balance for surfaces that neither face the exterior nor the room. The heat balance is formulated for the surface number 2, the heat balance for any surface with index $i \in \{3,\ldots,2N-1\}$ is similar. For the surface number 2, we have

$$\begin{aligned} 0 = K^1 \left(T^1_{win}(x,t) - T^2_{win}(x,t)\right) + \frac{\sigma}{R^1_{rad}} \left(T^3_{win}(x,t)^4 - T^2_{win}(x,t)^4\right) \\ + U^1\big(T^2_{win}(x,t), T^3_{win}(x,t)\big) \left(T^3_{win}(x,t) - T^2_{win}(x,t)\right) + \frac{S^1(x,t)}{2}, \end{aligned} \tag{A.4.112a}$$

where K^1 denotes the conductance of the pane number 1, $U^1\big(T^2_{win}(x,t), T^3_{win}(x,t)\big)$ de-

notes the convective and conductive heat transfer of the gas gap number 1, and R^1_{rad} denotes the radiative heat transfer resistance between the surface number 2 and the surface number 3.

To obtain the conductance in the gas gap $U^1(T^2_{win}(x,t), T^3_{win}(x,t))$, we first introduce the mean gap temperature

$$T^1_{gap}(x,t) \triangleq \frac{T^2_{win}(x,t) + T^3_{win}(x,t)}{2}. \tag{A.4.112b}$$

The fluid properties of the gas gap can be approximated as

$$p^1(x,t) = p_0 + \frac{dp}{dT}\,(T^1_{gap}(x,t) - 273), \tag{A.4.112c}$$

where the symbol p stands for either the viscosity μ, the mass density ρ, the Prandtl number Pr or the thermal conductivity k. The Grashoff number is (Kays and Crawford, 1993)

$$Gr^1(x,t) = g\,\frac{(u^1)^3\,|T^2_{win}(x,t) - T^3_{win}(x,t)|\,\rho^1(x,t)^2}{T^1_{gap}(x,t)\,\mu^1(x,t)^2}, \tag{A.4.112d}$$

where u^1 is the gap width. Note that $Gr^1(x,t)$ is not differentiable in $T_{win}(x,t)$. However, as we will shortly see, the heat conductance through the gap can be approximated by a constant, and hence, we do not smoothen (A.4.112d). For vertical gaps, the Nusselt

number is (Arasteh et al., 1989)

$$Nu^1(x,t) = \left(1 + 1.9764\,10^{-17}\,Ra^1(x,t)^{4.422}\right)^{0.091}, \tag{A.4.112e}$$

where $Ra^1(x,t)$ is the Rayleigh number (Kays and Crawford, 1993), which is defined as

$$Ra^1(x,t) = Gr^1(x,t)\,Pr^1(x,t). \tag{A.4.112f}$$

Now, the conductance of the gap can be computed as

$$U^1(x,t) = \frac{k^1(x,t)}{u^1\,Nu^1(x,t)}. \tag{A.4.112g}$$

Fig. A.11 shows (A.4.112g) for different gap widths u and for different temperature differences $(T^1_{win}(x,t) - T^2_{win}(x,t))$.

We assume that the glass is opaque for infrared radiation and we approximate the panes by infinitely large parallel planes. Then, the radiative heat transfer coefficient of the gap number 1 is

$$R^1_{rad} = \left(\frac{1-\varepsilon^2_{IR}}{\varepsilon^2_{IR}} + 1 + \frac{1-\varepsilon^3_{IR}}{\varepsilon^3_{IR}}\right)^{-1}. \tag{A.4.112h}$$

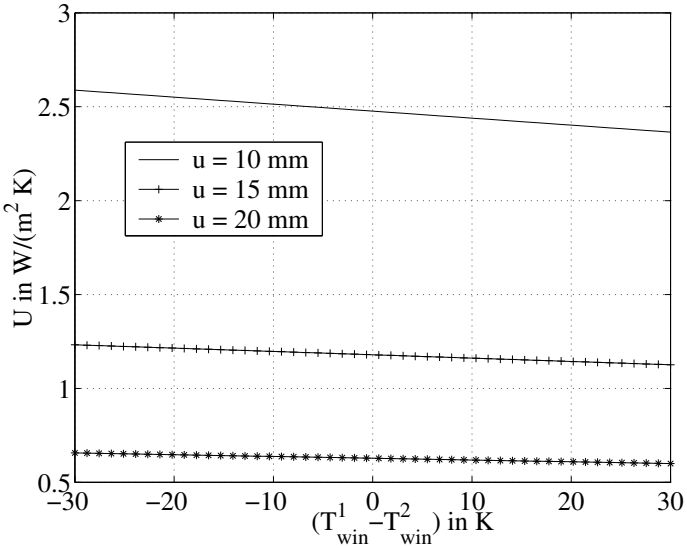


Figure A.11: Conductivity of window gap for different gap widths u as a function of the temperature difference $T^1_{win}(x,t) - T^2_{win}(x,t)$, with $T^1_{win}(x,t) = 15°\text{C}$.

Similarly to (A.4.111a), we can write for the room facing surface

$$\begin{aligned} 0 = {} & H_{IR}(x,t) + K^N \left(T^{2N-1}_{win}(x,t) - T^{2N}_{win}(x,t) \right) \\ & + h^0_{con}(x,t) \left(T_{air}(x,t) - T^{2N}_{win}(x,t) \right) + h^0_{IR}(x,t) \left(T^*(x,t) - T^{2N}_{win}(x,t) \right) \\ & + \frac{S^N(x,t)}{2}, \end{aligned} \tag{A.4.113a}$$

where $H_{IR}(\cdot,\cdot)$ is the heat gain due to infrared irradiation from the zone, introduced in (A.4.41f), and $T^*(\cdot,\cdot)$ is the radiation temperature introduced in (A.4.45a).

Solution of the Heat Balance Equations The equations (A.4.111a), (A.4.112a) and (A.4.113a) are a coupled system of nonlinear equations with $2N$ unknowns. We will linearize this system around $T_0 \triangleq (15 + 273.16)$ Kelvin.

Cuzzillo and Pagni (1998) showed that conduction through stagnant air is the dominant heat transport mechanism for air or argon gaps up to 10 mm thickness. Furthermore, Fig. A.11 shows that $U\left(T^1_{win}(x,t), T^2_{win}(x,t)\right)$ is not sensitive to the temperature difference between the panes, and hence convection contributes little to the heat transfer through the gap. Thus, we set the conductance of the gas gap to the constant value

$$U^1 = \frac{k^1}{u^1}. \tag{A.4.114}$$

Now we can reformulate (A.4.111a), (A.4.112a), and (A.4.113a) as

$$A(x,t)\, T_{win}(x,t) = Q(x,t), \tag{A.4.115}$$

where $A \in \mathbb{R}^{2N\times 2N}$, $Q \in \mathbb{R}^{2N}$ and $T_{win} \in \mathbb{R}^{2N}$ is the vector of unknown variables. The non-zero elements of the matrix $A(x,t)$ are

$$A^{1,1}(x,t) = K^1 + h^1_{con}(x,t) + h^1_{IR}(t), \tag{A.4.116a}$$

$$A^{1,2} = -K^1, \tag{A.4.116b}$$

and with $i \in \{2,4,\ldots,2N-2\}$,

$$A^{i,i-1} = -K^{i/2}, \tag{A.4.116c}$$

$$A^{i,i} = K^{i/2} + 4\frac{\sigma}{R_{rad}^{i/2}} T_0^3 + U_0^{i/2}, \tag{A.4.116d}$$

$$A^{i,i+1} = -4\frac{\sigma}{R_{rad}^{i/2}} T_0^3 - U_0^{i/2}, \tag{A.4.116e}$$

and with $i \in \{3,5,\ldots,2N-1\}$,

$$A^{i,i-1} = -4\frac{\sigma}{R_{rad}^{(i-1)/2}} T_0^3 - U_0^{(i-1)/2}, \tag{A.4.116f}$$

$$A^{i,i} = K^{(i+1)/2} + 4\frac{\sigma}{R_{rad}^{(i-1)/2}} T_0^3 + U_0^{(i-1)/2}, \tag{A.4.116g}$$

$$A^{i,i+1} = -K^{(i+1)/2}, \tag{A.4.116h}$$

and

$$A^{2N,2N-1} = -K^N, \tag{A.4.116i}$$

$$A^{2N,2N}(x,t) = K^N + h_{con}^0(x,t) + h_{IR}^0(x,t). \tag{A.4.116j}$$

The vector $Q \in \mathbb{R}^{2N}$ has components

$$Q^1(x,t) = \frac{S^1(x,t)}{2} + h^1_{con}(x,t)\,T_{out}(t) + h^1_{IR}(t)\,T_{env}(t), \tag{A.4.117a}$$

$$Q^i(x,t) = \frac{S^{i/2}(x,t)}{2}, \qquad i \in \{2,4,\ldots,2N-2\}, \tag{A.4.117b}$$

$$Q^i(x,t) = \frac{S^{(i+1)/2}(x,t)}{2}, \qquad i \in \{3,5,\ldots,2N-1\}, \tag{A.4.117c}$$

$$Q^{2N}(x,t) = \frac{S^N(x,t)}{2} + h^0_{con}(x,t)\,T_{air}(x,t) + h^0_{IR}(x,t)\,T^*(x,t) + H_{IR}(x,t). \tag{A.4.117d}$$

The linear system of equations (A.4.115) is solved as a band diagonal linear system of equations using LU-decomposition. Even though only a tridiagonal system needs to be solved, we use LU-decomposition because of higher robustness of the algorithm. The algorithm for solving tridiagonal systems can fail algorithmically even for nonsingular matrices (see Press et al. (1992)).

Module Description

Parameter

Variable	Description
N	number of panes
$\{k^i_{glass}\}_{i\in\mathbf{N}}$	glass thermal conductivity
$\{l^i\}_{i\in\mathbf{N}}$	glass thickness
$\{k^i\}_{i=1}^{N-1}$	gas thermal conductivity
$\{u^i\}_{i=1}^{N-1}$	gas gap width
$\{A^{i,j}_{\rightarrow}\}_{i\in\mathbf{I},\, j\in\mathbf{N}}$	solar absorbtance for irradiation from outside (no shading device)
$\{A^{\cap,j}_{\leftarrow}\}_{j\in\mathbf{N}}$	solar absorbtance for hemispherical irradiation from inside

Continued on next column.

Parameter (continued)

Variable	Description
$\{A^{i,j}_{\tau^1,\rightarrow}\}_{i\in\mathbf{I},\, j\in\mathbf{N}}$	solar absorbtance for irradiation from outside (exterior shading device)
$\{A^{i,j}_{\tau^N,\rightarrow}\}_{i\in\mathbf{I},\, j\in\mathbf{N}}$	solar absorbtance for irradiation from outside (interior shading device)
$\{T^{i,1-\tau,N}\}_{i\in\mathbf{I},\, j\in\mathbf{N}}$	solar transmittance for irradiation from outside (exterior shading device)

Continued on next page.

Parameter (continued)

Variable	Description
$\{T^{i,1,N}\}_{i\in\mathbf{I},\, j\in\mathbf{N}}$	solar transmittance for irradiation from outside (no shading device)
$\{T^{i,1,N+\tau}\}_{i\in\mathbf{I},\, j\in\mathbf{N}}$	solar transmittance for irradiation from outside (interior shading device)

Input

Variable	Description
$T_{out}(t)$	outside temperature
$T_{env}(t)$	environment temperature for infrared radiation, as introduced in (A.4.44b)
$T_{air}(x,t)$	room-side air temperature
$T^*(x,t)$	room radiation temperature, as introduced in (A.4.45a))
$\phi(x,t)$	solar incidence angle

Continued on next page.

Input (continued)

Variable	Description
$H^1_{dir,til}\big(\phi(x,t)\big)$	direct solar irradiation on tilted surface
$H^1_{dif,til}\big(\phi(x,t)\big)$	diffuse solar irradiation on tilted surface
$H^0_{sw}(x,t)$	short wave room irradiation on window from room-side
$H_{IR}(x,t)$	infrared heat gain due to internal gains from the room

Output

Variable	Description
$\{T^i_{win}(x,t)\}^{2N}_{i=1}$	window glass surface temperatures

Algorithm – Initialization Constant Values

1: **for all** $i \in \mathbf{N}$ **do**
2: $\quad K^i \overset{\text{(A.4.111b)}}{\longleftarrow} k^i_{glass},\ l^i;$
3: **end for**
4: **for all** $i \in \{1, 2, \ldots, N-1\}$ **do**
5: $\quad U^i_0 \overset{\text{(A.4.114)}}{\longleftarrow} k^i,\ u^i;$
6: **end for**
7: **for all** $j \in \mathbf{N}$ **do**
8: $\quad \{a_{A^j_{\rightarrow},i}\}^5_{i=0} \overset{\text{(A.4.110a)}}{\longleftarrow} A^{\triangleleft,j}_{\rightarrow};$
9: $\quad \{a_{A^j_{\tau^1,\rightarrow},i}\}^5_{i=0} \overset{\text{(A.4.110a)}}{\longleftarrow} A^{\triangleleft,j}_{\tau^1,\rightarrow};$
10: $\quad \{a_{A^j_{\tau^N,\rightarrow},i}\}^5_{i=0} \overset{\text{(A.4.110a)}}{\longleftarrow} A^{\triangleleft,j}_{\tau^N,\rightarrow};$
11: **end for**
12: $\{a_{T^{\triangleleft,1-\tau,N},i}\}^5_{i=0} \overset{\text{(A.4.110a)}}{\longleftarrow} T^{\triangleleft,1-\tau,N};$
13: $\{a_{T^{\triangleleft,1,N},i}\}^5_{i=0} \overset{\text{(A.4.110a)}}{\longleftarrow} T^{\triangleleft,1,N};$
14: $\{a_{T^{\triangleleft,1,N+\tau},i}\}^5_{i=0} \overset{\text{(A.4.110a)}}{\longleftarrow} T^{\triangleleft,1,N+\tau};$
15: $\{a_{A^{\triangleleft,\tau^N}_{\rightarrow},i}\}^5_{i=0} \overset{\text{(A.4.110a)}}{\longleftarrow} A^{\triangleleft,\tau^N}_{\rightarrow};$

Algorithm – Initialization Time-Dependent Values

1: **for all** $i \in \mathbf{N}$ **do**
2: $\quad S^i(x,t) \overset{\text{(A.4.111c)}}{\longleftarrow} H^1_{dir,til}\big(\phi(x,t)\big),\ \{a_{A^{\triangleleft,\tau^N}_{\rightarrow},i}\}^5_{i=0},\ \phi(x,t),$
$\quad H^1_{dif,til}\big(\phi(x,t)\big),\ A^{\cap,i}_{\rightarrow},\ H^0_{sw}(x,t),\ A^{\cap,i}_{\leftarrow},\ H^0_{IR}(x,t);$
3: **end for**

Algorithm – Heat Balance

1: compute $Q(x,t)$ using (A.4.117);
2: compute $A(x,t)$ using (A.4.116);
3: $T_{win}(x,t) \overset{\text{(A.4.115)}}{\longleftarrow} A(x,t),\ Q(x,t);$

A.4.3.6 Window Shading Control

We will now present the model for the window shading control. The window shading control laws that are implemented in BuildOpt are (a) shading always off, (b) shading always on, and (c) shading on if the sum of direct and diffuse solar irradiation per window unit area exceeds a user-specified set point, which is denoted by H_{set}. All shading control laws can be deactivated by specifying a seasonal, weekly, and daily schedule.

For the schedule law (c), the control signal is

$$y_{sc}(x,t) = \widetilde{\mathrm{H}}\left(\frac{H_{dir,til}(x,t) + H_{dif,til}(t) - H_{set}}{H_{set}}\right). \tag{A.4.118}$$

If $0 < y_{sc}(x,t) < 1$, a fraction $y_{sc}(x,t)$ of the window area is assumed to have the shading device activated, and a fraction of $1 - y_{sc}(x,t)$ is assumed to have no shading device activated. This makes the computation once Lipschitz continuously differentiable in x and in t.

A.4.3.7 Partially Shaded Window and Window with Partially Deployed Shading Device

In order to make the equations of the window heat balance model once Lipschitz continuously differentiable, we allow the window shading device to be partially deployed rather than taking only 0 or 1 as permissible values. Let $y_{sc}(x,t) \in [0,\ 1]$ denote the control signal of the window shading device, with $y_{sc}(x,t) = 0$ if the shading device is

not activated, as introduced in (A.4.118). Similarly, let $r(x,t) \in [0,\ 1]$ denote the fraction of the window area that is shaded by a window overhang, with $r(x,t) = 0$ if the window is unshaded, as introduced in (A.4.109b). We compute the window heat balance for four different states, namely for

1: shading device activated, window completely shaded,

2: shading device deactivated, window completely shaded,

3: shading device activated, window completely unshaded, and

4: shading device deactivated, window completely unshaded.

We approximate the temperature of a partially shaded window with partially activated shading device as

$$\begin{aligned} T_{win}(x,t) \;=\; & y_{sc}(x,t)\, r(x,t)\, T_{win,1}(x,t) \\ & + \big(1 - y_{sc}(x,t)\big)\, r(x,t)\, T_{win,2}(x,t) \\ & + y_{sc}(x,t)\, \big(1 - r(x,t)\big)\, T_{win,3}(x,t) \\ & + \big(1 - y_{sc}(x,t)\big)\, \big(1 - r(x,t)\big)\, T_{win,4}(x,t), \end{aligned} \tag{A.4.119}$$

where $T_{win,i}(x,t)$, with $i \in \{1, 2, 3, 4\}$, denotes the window surface temperature of the four states introduced above.

A.4.3.8 Short Wave Radiation from the Outside to the Room

For windows with an exterior shading device, the transmitted solar radiation from the outside to the room is

$$\begin{aligned} H_{dir,r}(x,t) &= H^1_{dir,til}\big(\phi(x,t)\big)\,\big(1-r(x,t)\big) && \text{(A.4.120a)}\\ &\quad \big(y_{sc}(x,t)\,T^{\lhd,1-\tau,N}\big(\phi(x,t),x\big)+(1-y_{sc}(x,t))\,T^{\lhd,1,N}\big(\phi(x,t)\big)\big)\,, \\ H_{dif,r}(t) &= H^1_{dif,til}(t)\,\big(y_{sc}(x,t)\,T^{\cap,1-\tau,N}(x)+\big(1-y_{sc}(x,t)\big)\,T^{\cap,1,N}\big), && \text{(A.4.120b)} \end{aligned}$$

where $r(x,t)$ is the fraction of the window that is shaded by the overhang, which is introduced in (A.4.109b), and $y_{sc}(x,t)$ is the shading control signal, which is introduced in (A.4.118).

For windows with interior shading device, $T^{\lhd,1-\tau,N}(\cdot,\cdot)$ need to be replaced with $T^{\lhd,1,N+\tau}(\cdot,\cdot)$ and $T^{\cap,1-\tau,N}(\cdot)$ need to be replaced with $T^{\cap,1,N+\tau}(\cdot)$.

For windows with no shading device, $y_{sc}(x,t)$ is equal to zero for all x and for all t.

A.4.3.9 Energy Balance for the Room Air

The room air is assumed to be completely mixed. The time rate of change of the room air temperature is proportional to the sum of the convective heat transfer with the room enclosure $Q_{con}(x,t)$, the convective heat gains $Q_{cog}(t)$, the heat input from the air conditioning system $Q_{sys}(x,t)$, and the heat transfer due to air infiltration or due to

increased conductance, such as through a window frame, $Q_U(x,t)$. Hence,

$$C_{roo}\,\frac{dT_{air}}{dt}(x,t) = Q_{con}(x,t) + Q_{cog}(t) + Q_{sys}(x,t) + Q_U(x,t), \tag{A.4.121a}$$

where C_{roo} is the heat capacity of the room. The thermal capacity of the room is

$$C_{roo} = \gamma \rho_{air} V_{roo}\, c_p, \tag{A.4.121b}$$

where $\gamma \geq 1$ is a constant that can be used to take the thermal capacity of the room's furniture into account. The convective heat exchange with the room enclosure is

$$Q_{con}(x,t) = \sum_{n=1}^{N_{sur}} h_{con}(x,t) A^n(x) \left(T^n_{sur}(x,t) - T_{air}(x,t)\right), \tag{A.4.121c}$$

where $T^n_{sur}(x,t)$ is the surface temperature of opaque constructions or of windows, $h_{con}(x,t)$ is the convective heat transfer coefficient introduced in Section A.4.3.1 on page 169, and the sum is over all surfaces of the room enclosure. The convective heat gain $Q_{cog}(t)$ is specified by a time schedule as described in Section A.4.2.6 on page 166. The computation of the heat input from the HVAC system is described in the next section. Increased heat conduction, such as through a window frame, can be taken into account by specifying for each construction a conductance U^n. This conductance can also be used to model a constant air flow between adjacent rooms and outside air infiltration. The additional

heat transfer is

$$Q_U(x,t) = \sum_{n=1}^{N_{sur}} U^n A^n(x) \left(T^n_{ext}(x,t) - T_{air}(x,t)\right), \tag{A.4.121d}$$

where for exterior constructions, $T^n_{ext}(x,t)$ is the outside air temperature, and for interior constructions, $T^n_{ext}(x,t)$ is the adjacent zone's air temperature.

A.4.3.10 HVAC Control Scheme

Since we do not model a detailed HVAC system, we do not implement a controller as it is found in HVAC systems. Instead of implementing a real controller, we compute the HVAC heating or cooling power $Q_{sys}(x,t)$ using a model based on Euler integration. Let $T_h(t)$ denote the heating set point temperature, let $T_c(t)$ denote the cooling set point temperature, with $T_c(t) \leq T_h(t)$ for all $t \geq 0$, and let Δt be a fixed time-interval, which we define as $\Delta t \triangleq 15\,\text{min}$. To simplify the notation, we introduce

$$Q_{no,sys}(x,t) \triangleq Q_{con}(x,t) + Q_{cog}(t) + Q_U(x,t), \tag{A.4.122}$$

where $Q_{con}(x,t)$ are the convective heat flux from the room enclosure surfaces to the room air, as defined in (A.4.121c), $Q_{cog}(t)$ are the convective heat gains due to internal loads, which are specified by a time schedule as described in Section A.4.2.6 on page 166 and $Q_U(x,t)$ are the heat gains due to increased conduction or air infiltration, as defined in (A.4.121d).

First, we discuss the computation of the heating power. At any $t \geq 0$, the heating power is computed using a first order approximation so that $T_{air}(x, t+\Delta t) \approx T_h(t)$. Using the Explicit Euler integration method, we can write

$$T_{air}(x, t+\Delta t) = T_{air}(x,t) + \frac{dT_{air}(x,t)}{dt}\Delta t, \tag{A.4.123a}$$

from which follows that

$$\frac{dT_{air}(x,t)}{dt} = \frac{T_h(t) - T_{air}(x,t)}{\Delta t}. \tag{A.4.123b}$$

Substituting (A.4.121a) in (A.4.123b) and solving for the heating power yields

$$\widetilde{Q}_{sys,h}(x,t) = C_{roo}\frac{T_h(t) - T_{air}(x,t)}{\Delta t} - Q_{no,sys}(x,t). \tag{A.4.123c}$$

Equation (A.4.123c) can yield a negative heating power. Therefore, we compute the heating power as

$$Q_{sys,h}(x,t) = \widetilde{\max}(0,\ \widetilde{Q}_{sys,h}(x,t)). \tag{A.4.123d}$$

For the cooling case, we obtain similarly

$$Q_{sys,c}(x,t) = \widetilde{\min}\left(0,\ C_{roo}\frac{T_c(t+\Delta t) - T_{air}(x,t)}{\Delta t(x)} - Q_{no,sys}(x,t)\right). \tag{A.4.123e}$$

The system power is computed as

$$Q_{sys}(x,t) = Q_{sys,h}(x,t) + Q_{sys,c}(x,t). \tag{A.4.123f}$$

This defines $Q_{sys}(\cdot,\cdot)$ as a once Lipschitz continuously differentiable function of x and t.

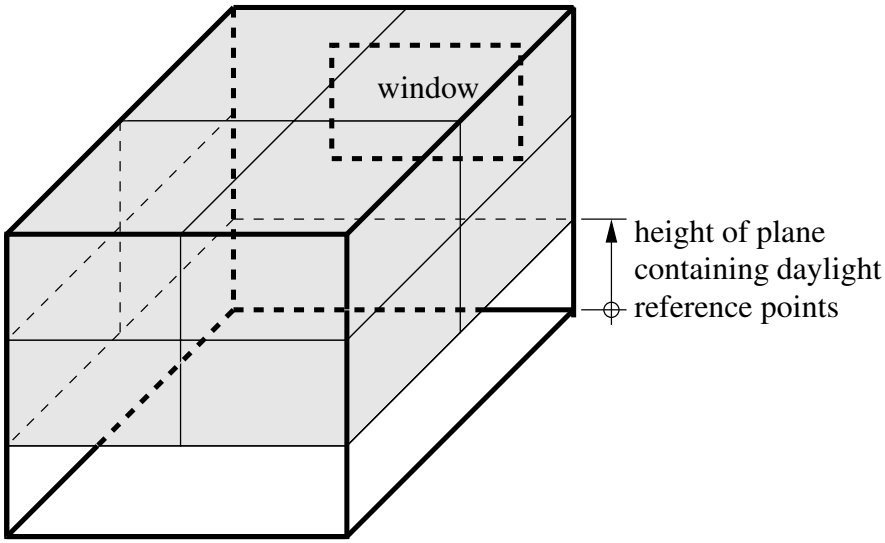


Figure A.12: Location of patches (displayed in gray) on the walls and the ceiling used to compute the illuminance at the daylight reference points due to room internal reflections.

A.4.4 Daylighting and Electric Lighting

A.4.4.1 Introduction

For each thermal zone, the model for daylighting and electric lighting computes the horizontal illuminance at two user-specified reference points at each time step. The interior light transfer model is similar to the model used in the daylighting simulation program DeLight developed by Vartiainen (2000). Despite its simplicity, DeLight's average simulation error is only $2\% - 3\%$ at illuminace levels of $300 - 500$lx when compared to year-round illuminance and irradiance measurements (Vartiainen, 2000). Considerable errors are possible when direct sunlight enters the room. However, usually most of the direct sunlight is in excess of the recommended workplace illuminance. Therefore, for lighting energy calculations, accurate modeling of situations where no direct sunlight

enters the room is more important.

We describe our model for only one reference point, which we denote by P_r. The computations for the other reference point are identical. To reduce computation time, both reference points are required to be at the same height.

The model requires a rectangular closed room with surfaces that are perpendicular to each other as shown in Fig. A.12. One of the walls must contain one window. To compute the illuminance at the reference points due to room internal reflections, the walls and the ceiling are divided into four rectangular surfaces of equal area, which we will call *patches*. All patches are located right above the plane that contains the daylight reference points.

The model is based on the assumptions that the reflections at the room surfaces and at the ground outside the building are isotropic, and that obstructions inside or outside the building do not affect the daylight illuminance at the reference points. Only first reflections in the room are computed, multiple reflections in the room are neglected. This implies that reflections at the floor and at the wall that contains the window are neglected.

Direct radiation enters the room in only one direction (which is a function of time). Thus, modeling advanced daylighting devices such as specular shading devices or light shelves is not possible (but in principle, the model could be extended for such cases).

Our model differs from DeLight in the computation of the angular dependency of the

window's solar transmittance: DeLight uses the approximation of Riviero (Bryan and Clear, 1981), whereas we use the more detailed window model described in Section A.4.3.3 which is based on the algorithm used in the WINDOW 4 program (Arasteh et al., 1994; Finlayson et al., 1993).

To reduce the computation time, we use less patches than DeLight. Using bigger patches increases the error in the approximate computation of the illuminance of the enclosing surfaces, in particular for patches that are close to the window. To reduce this approximation error, we average the view factors and the angular dependency of the window transmittance over each patch. This averaging is not done in the DeLight program. Since the averaging must be done only once per simulation, it is computationally cheap.

The horizontal illuminance at the surface element located at the reference point, which is denoted by dA_r, is computed as the sum of the direct (beam) illuminance $E_{dir,dA_r}(x,t)$, the diffuse illuminance $E_{dif,dA_r}(x,t)$, and the illuminance due to reflections at the walls and the ceiling $E_{ref,dA_r}(x,t)$.

Throughout this section, let A_p denote the receiving surface, let ΔA_p denote a patch of A_p, let dA_p denote an infinitesimal small area of ΔA_p and let A_s denote the radiating surface.

We will now begin the model description.

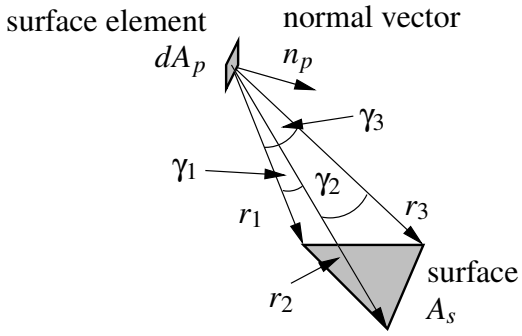


Figure A.13: Nomenclature used for computing the view factor using (A.4.124b) for the case of $N = 3$.

A.4.4.2 View Factor

We will now describe how the model computes the view factors between dA_p and A_s, which is (see for example Holman (1997))

$$F_{dA_p - A_s} = \int_{A_s} \frac{\cos(\theta_p)\cos(\theta_s)}{\pi S^2}\, dA_s, \tag{A.4.124a}$$

where θ_p and θ_s are the angles between the line that connects dA_p and dA_s and the respective surface outward normal vector, and S is the distance between dA_p and dA_s. The method is analytical and based on contour integration. It is valid for the case of A_s being a surface of an arbitrarily oriented planar polygon with an arbitrary number of vertices, and dA_p being an infinitesimal small surface element oriented so that dA_p can see A_s from one side only. A derivation of the method can be found in Hottel and Sarofim (1967), Baum et al. (1989) and Durand et al. (1999).

Using the notation defined in Fig. A.13 and defining the auxiliary variable $r_{N+1} \triangleq r_1$,

the view factor is

$$F_{dA_p - A_s} = \frac{1}{2\pi} \left\langle n_p,\ \sum_{n=1}^{N} \gamma_n \frac{r_n \times r_{n+1}}{|\, r_n \times r_{n+1}\,|} \right\rangle, \tag{A.4.124b}$$

where $r_n \times r_{n+1}$ denotes the cross product and γ_n is the angle between r_n and r_{n+1}, which is

$$\gamma_n = \arccos\left(\frac{\langle r_n,\ r_{n+1} \rangle}{|r_n|\,|r_{n+1}|} \right). \tag{A.4.124c}$$

A.4.4.3 Window Transmittance for Visible Radiation

The window transmittance for visible radiation is computed as described in Section A.4.3.3. If the window construction has an exterior or an interior shading device, then we compute the window transmittance separately for the window with activated and with deactivated shading device.

Let $y_{sc}(x,t)$ denote the control signal for the shading device, computed by (A.4.118), and let $T_{sa}^{\triangleleft}(\cdot)$ and $T_{ds}^{\triangleleft}(\cdot)$ denote the window transmittance for the window with activated and deactivated shading device, respectively, obtained using the 5-th order polynomial fit described in (A.4.110a). Using these polynomials, we compute the window transmittance for the incidence angle $\phi_i(x,t)$ as

$$T^{\triangleleft}\big(\phi_i(x,t);x,t\big) = y_{sc}(x,t)\, T_{sa}^{\triangleleft}\big(\phi_i(x,t)\big) + \big(1 - y_{sc}(x,t)\big)\, T_{ds}^{\triangleleft}\big(\phi_i(x,t)\big). \tag{A.4.125}$$

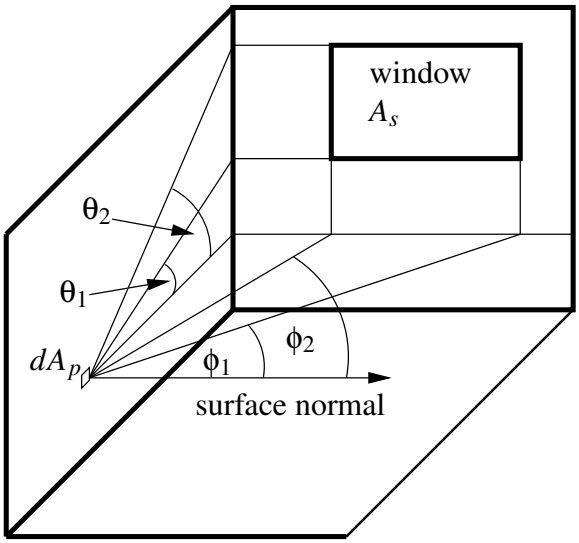


Figure A.14: Altitude and azimuth angle of the window edges as seen from a surface element dA_p.

A.4.4.4 Direct Illuminance

Using the notation shown in Fig. A.14, the sun is visible through a window with no shading overhang from a surface element dA_p if the solar zenith angle $\theta_s(t)$, as defined in Fig. A.3 on page 144, satisfies

$$\theta_1(x) < \frac{\pi}{2} - \theta_s(t) < \theta_2(x), \tag{A.4.126a}$$

and if, in addition, the azimuth of the sun relative to the normal vector of dA_p satisfies

$$\phi_1(x) < \Delta\phi(x,t) < \phi_2(x), \tag{A.4.126b}$$

where $\Delta\phi(x,t) = \phi_s(t) - \phi_p(x,t)$ is as in (A.4.106). If both conditions (A.4.126a) and (A.4.126b) are satisfied, then the direct illuminance at a *horizontal* surface element dA_p is

$$\widehat{E}_{dir,dA_p}(x,t) = T^{\triangleleft}\big(\phi_i(x,t);x,t\big)\, G_{dir,hor}(t), \tag{A.4.126c}$$

where $G_{dir,hor}(t) = G_{glo,hor}(t) - G_{dif,hor}(t)$ is the direct horizontal illuminance outside the building, $\phi_i(x,t)$ is the solar beam incidence angle on the window surface, and $T^{\triangleleft}\big(\phi_i(x,t);x,t\big)$ is the window's visible transmittance, defined in (A.4.125).

If conditions (A.4.126a) and (A.4.126b) are both satisfied, then the direct illuminance at a *vertical* surface element dA_p is

$$\widehat{E}_{dir,dA_p}(x,t) = T^{\triangleleft}\big(\phi_i(x,t);x,t\big)\, G_{dir,nor}(t)\cos\big(\phi_{i,dA_p}(x,t)\big), \tag{A.4.126d}$$

where $G_{dir,nor}(t)$ is the direct normal illuminance outside the building and $\phi_{i,dA_p}(x,t)$ is the incidence angle of the solar beam on the surface element dA_p.

The conditions (A.4.126a) and (A.4.126b) cause $\widehat{E}_{dir,dA_p}(\cdot,\cdot)$ to be a non-differentiable function of x and t. Furthermore, they do not take into account that a window may be partially shaded by an external shading device, such as an overhang. To make the computation once Lipschitz continuously differentiable in x and t and to take the shading due to a window overhang into account, we reformulate the computations as follows. First,

we reformulate the conditions (A.4.126a) and (A.4.126b) by defining, for $N \triangleq 100$,

$$\delta_\theta(x) \triangleq \frac{|\theta_2(x)-\theta_1(x)|}{N}, \tag{A.4.126e}$$

$$f_{dA_p,\theta}(x,t) \triangleq \widetilde{\mathrm{H}}\left(\frac{\pi}{2}-\theta_s(t)-\theta_1(x)-\delta_\theta(x);\delta_\theta(x)\right) \widetilde{\mathrm{H}}\left(\theta_2(x)-\frac{\pi}{2}+\theta_s(t)-\delta_\theta(x);\delta_\theta(x)\right), \tag{A.4.126f}$$

$$\delta_\phi(x) \triangleq \frac{|\phi_2(x)-\phi_1(x)|}{N}, \tag{A.4.126g}$$

$$f_{dA_p,\phi}(x,t) \triangleq \widetilde{\mathrm{H}}\left(\Delta\phi(x,t)-\phi_1(x)-\delta_\phi(x);\delta_\phi(x)\right) \widetilde{\mathrm{H}}\left(\phi_2(x)-\Delta\phi(x,t)-\delta_\phi(x);\delta_\phi(x)\right), \tag{A.4.126h}$$

$$f_{dA_p,sol}(x,t) \triangleq f_{dA_p,\theta}(x,t)\, f_{dA_p,\phi}(x,t). \tag{A.4.126i}$$

This defines $f_{dA_p,sol}(\cdot,\cdot)$ as a once Lipschitz continuously differentiable function of x and t. If the window is partially shaded by an overhang, we multiply the direct illuminance by one minus the fraction of the window that is shaded by the overhang, i.e., by $(1-r(x,t))$. The fraction $r(x,t)$ is obtained from (A.4.109b). Thus, for any $\theta_s(t)$ and for any $\Delta\phi(x,t)$, we compute for a *horizontal* surface element

$$E_{dir,dA_p}(x,t) = f_{dA_p,sol}(x,t)\left(1-r(x,t)\right) T^{\triangleleft}\left(\phi_i(x,t);x,t\right) G_{dir,hor}(t) \tag{A.4.126j}$$

and for a *vertical* surface element

$$E_{dir,dA_p}(x,t) = f_{dA_p,sol}(x,t)\left(1 - r(x,t)\right)$$
$$T^{\triangleleft}\left(\phi_i(x,t);x,t\right) G_{dir,nor}(t) \cos\left(\phi_{i,dA_p}(x,t)\right). \quad \text{(A.4.126k)}$$

To compute the direct illuminance on a surface patch $E_{dir,\Delta A_p}(\cdot,\cdot)$, we place 49 surface elements equidistantly on the patch so that each corner contains one surface element $dA_{p,k}$, and all other surface elements are located equidistantly on a rectangular mesh between those corner elements. The direct illuminance on the patch is then approximated as

$$E_{dir,\Delta A_p}(x,t) \approx \frac{1}{49}\sum_{k=1}^{49} E_{dir,dA_{p,k}}(x,t). \quad \text{(A.4.126l)}$$

A.4.4.5 Diffuse Illuminance

First, we describe how the diffuse illuminance on a patch ΔA_p, i.e., a rectangle with horizontal base line, is computed. Since ground and sky have different luminance, we need to know for every element dA_p of ΔA_p what fraction of the window, as seen from dA_p, shows the ground and what fraction shows the sky. Let $f_{dA_p,gro}$ denote the fraction that shows the ground.

As shown in Fig. A.15, let $z_l(x)$ and $z_u(x)$ be the height coordinates of the window, and let z_{min} and z_{max} be the height coordinates of the patch ΔA_p. We will assume that the horizon is horizontal and that the view to the horizon is unobstructed. Then, for an

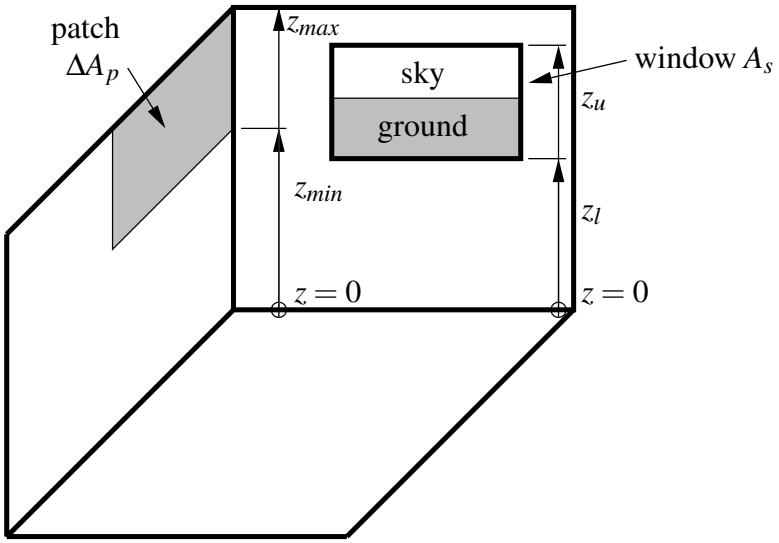


Figure A.15: Nomenclature used in computing $f_{dA_p,gro}(z;x)$ according to (A.4.127b).

infinitesimal small area dA_p at height z, the fraction of the view through the window that shows the ground is

$$\widehat{f}_{dA_p,gro}(z;x) = \begin{cases} 0, & \text{if } z < z_l(x), \\ \frac{z-z_l(x)}{z_u(x)-z_l(x)}, & \text{if } z_l(x) \le z \le z_u(x), \\ 1, & \text{if } z > z_u(x). \end{cases} \tag{A.4.127a}$$

Since $\widehat{f}_{dA_p,gro}(z;\cdot)$ is not continuously differentiable, we replace it by

$$f_{dA_p,gro}(z;x) = \widetilde{p}\left(\frac{z-z_l(x)}{z_u(x)-z_l(x)}\right). \tag{A.4.127b}$$

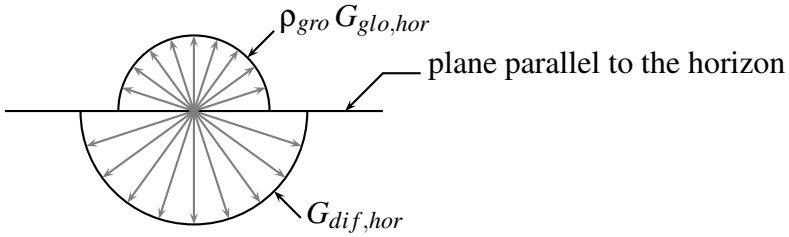


Figure A.16: Spherical distribution of the diffuse illuminance.

Thus, averaged over ΔA_p, we have

$$f_{\Delta A_p,gro}(x) = \frac{1}{z_{max} - z_{min}} \int_{z_{min}}^{z_{max}} f_{dA_p,gro}(z;x)\, dz. \tag{A.4.127c}$$

We approximate the integral by the sum

$$f_{\Delta A_p,gro}(x) \approx \frac{1}{N+1} \sum_{n=0}^{N} f_{dA_p,gro}\left(z_{min} + \frac{n}{N}(z_{max} - z_{min}); x\right), \tag{A.4.127d}$$

with $N \triangleq 20$.

We will assume that the ground reflectance is diffuse and hence that the illuminance distribution is as shown in Fig. A.16. Therefore, the diffuse illuminance $G_{dif,\Delta A_p}(x,t)$ at the surface ΔA_p is

$$G_{dif,\Delta A_p}(x,t) = f_{\Delta A_p,gro}(x)\, \rho_{gro}\, G_{glo,hor}(t) + \left(1 - f_{\Delta A_p,gro}(x)\right) G_{dif,hor}(t), \tag{A.4.127e}$$

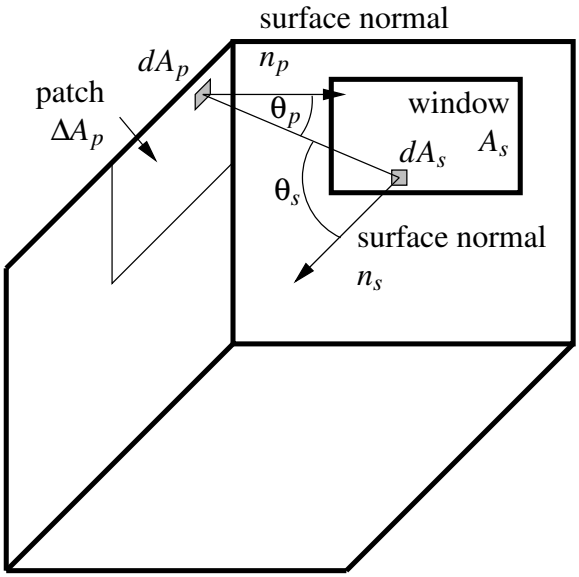


Figure A.17: Nomenclature used for computing the diffuse illuminance on ΔA_p.

where $G_{glo,hor}(t)$ and $G_{dif,hor}(t)$ are the global and diffuse horizontal illuminance available from the TMY2 weather data file and ρ_{gro} is the ground reflectance. We use interpolated weather data as described in Section A.4.2.4 on page 159.

To compute the illuminance on a surface ΔA_p, located *inside* the building, consider the configuration shown in Fig. A.17. As seen from ΔA_p, the illuminance of the aperture A_s is $G_{dif,\Delta A_p}(x,t)$, defined in (A.4.127e). The window attenuates each light beam that is incident on an element dA_s and that travels to an element dA_p by the factor $T^{\triangleleft}(\theta_s;x,t)$, defined in (A.4.125). Therefore, the diffuse illuminance on ΔA_p can be written as

$$E_{dif,\Delta A_p}(x,t) = \frac{G_{dif,\Delta A_p}(x,t)}{\Delta A_p} \int_{\Delta A_p} \int_{A_s(x)} T^{\triangleleft}(\theta_s;x,t) \frac{\cos(\theta_p)\cos(\theta_s)}{\pi S^2} dA_s\, dA_p. \tag{A.4.127f}$$

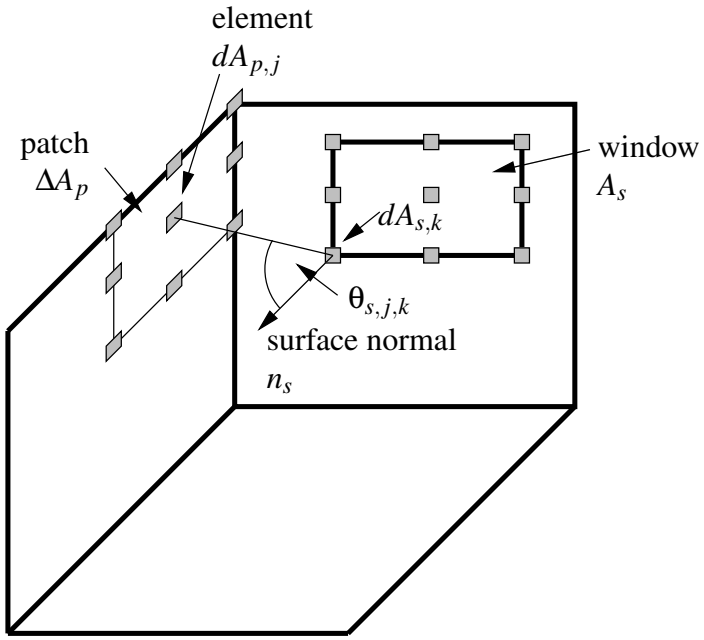


Figure A.18: Location of surface elements $dA_{p,j}$ and $dA_{s,k}$ used for approximating the window transmittance and the integral in (A.4.127g). For clarity, not all 49 surface elements have been shown.

This double integral cannot be solved analytically.

In evaluating $T^{\triangleleft}(\theta_s;x,t)$ in (A.4.127f), DeLight assumes that all beams are parallel to the line that connects the center of ΔA_p with the center of A_s. DeLight further approximates (A.4.127f) by replacing ΔA_p with the infinitesimal small area dA_p, located in the center of ΔA_p. Since we use patches that are bigger than the ones used in DeLight, we do not replace ΔA_p with one element dA_p only. We average the window transmittance

and approximate the view factor integral over ΔA_p as follows:

$$\begin{aligned} E_{dif,\Delta A_p}(x,t) &= \frac{G_{dif,\Delta A_p}(x,t)}{\Delta A_p} \int_{\Delta A_p} \int_{A_s(x)} T^{\triangleleft}(\theta_s;x,t) \frac{\cos(\theta_p)\cos(\theta_s)}{\pi S^2} \, dA_s \, dA_p \\ &\approx \frac{G_{dif,\Delta A_p}(x,t)}{\Delta A_p} \int_{\Delta A_p} \frac{1}{K} \left(\sum_{k=1}^{K} T^{\triangleleft}\big(\theta_{s,k}(x);x,t\big) \right) \int_{A_s(x)} \frac{\cos(\theta_p)\cos(\theta_s)}{\pi S^2} \, dA_s \, dA_p \\ &= \frac{G_{dif,\Delta A_p}(x,t)}{\Delta A_p} \int_{\Delta A_p} \frac{1}{K} \left(\sum_{k=1}^{K} T^{\triangleleft}\big(\theta_{s,k}(x);x,t\big) \right) F_{dA_p - A_s}(x) \, dA_p \\ &\approx \frac{G_{dif,\Delta A_p}(x,t)}{J\,K} \left(\sum_{j=1}^{J} \left(\sum_{k=1}^{K} T^{\triangleleft}\big(\theta_{s,j,k}(x);x,t\big) \right) F_{dA_{p,j} - A_s}(x) \right), \end{aligned} \tag{A.4.127g}$$

where we defined

$$F_{dA_{p,j} - A_s}(x) \triangleq \int_{A_s(x)} \frac{\cos(\theta_{p,j})\cos(\theta_s)}{\pi S^2} \, dA_s, \tag{A.4.127h}$$

which we approximate by (A.4.124b). In (A.4.127g), we set $J = K = 49$ and defined the angles $\theta_{s,j,k}(x)$, with $j \in \{1,\ldots,J\}$ and $k \in \{1,\ldots,K\}$, as follows: Given an element $dA_{p,j}$ on ΔA_p and an element $dA_{s,k}$ on A_s, located as indicated by the gray elements in Fig. A.18, the angle $\theta_{s,j,k}(x)$ is the incidence angle that a ray originating in $dA_{p,j}$ has at $dA_{s,k}$. In particular, if $n_s(x)$ is the normal vector of $A_s(x)$, and $r_{dA_{p,j}}$ and $r_{dA_{s,k}}(x)$ are the coordinate vectors of $dA_{p,j}$ and $dA_{s,k}$, respectively, then $\theta_{s,j,k}(x)$ is, for $j \in \{1,\ldots,J\}$ and $k \in \{1,\ldots,K\}$,

$$\theta_{s,j,k}(x) = \arccos\left(\frac{\left\langle r_{dA_{p,j}} - r_{dA_{s,k}}(x), n_s(x) \right\rangle}{\left| r_{dA_{p,j}} - r_{dA_{s,k}}(x) \right| \, |n_s(x)|} \right). \tag{A.4.127i}$$

In (A.4.127g), $T^{\triangleleft}\left(\theta_{s,j,k}(x);x,t\right)$ is a function of time only because of the window shading device control signal. Thus, we use (A.4.125) to decompose the computationally expensive double sum in (A.4.127g) into two different double sums which are both independent of time. We define

$$\widetilde{T}_{sa}(x) \triangleq \frac{1}{JK}\left(\sum_{j=1}^{J}\left(\sum_{k=1}^{K} T_{sa}^{\triangleleft}\left(\theta_{s,j,k}(x)\right)\right) F_{dA_{p,j}-A_s}(x)\right), \tag{A.4.128}$$

$$\widetilde{T}_{ds}(x) \triangleq \frac{1}{JK}\left(\sum_{j=1}^{J}\left(\sum_{k=1}^{K} T_{ds}^{\triangleleft}\left(\theta_{s,j,k}(x)\right)\right) F_{dA_{p,j}-A_s}(x)\right), \tag{A.4.129}$$

and rewrite (A.4.127g) as

$$E_{dif,\Delta A_p}(x,t) \approx G_{dif,\Delta A_p}(x,t)\left(y_{sc}(x,t)\,\widetilde{T}_{sa}(x) + \left(1-y_{sc}(x,t)\right)\widetilde{T}_{ds}(x)\right). \tag{A.4.130}$$

In (A.4.128) and (A.4.129), the computationally expensive sum is time-independent, and hence, must be evaluated only once in each simulation.

A.4.4.6 Luminous Flux Density at the Room Surfaces

After having computed the illuminance on the wall and the ceiling surface elements, we compute the surface's luminous flux density per unit area, which we denote by $G_s(x,t)$. Assuming diffuse reflections, we obtain at each patch ΔA_p,

$$G_{s,\Delta A_p}(x,t) = \rho_p\left(E_{dir,\Delta A_p}(x,t) + E_{dif,\Delta A_p}(x,t)\right), \tag{A.4.131}$$

where ρ_p denotes the surface reflectivity of A_p.

A.4.4.7 Illuminance at the Daylight Reference Point due to Reflections

Let dA_r denote the surface element located at the reference point. The illuminance at dA_r due to reflections is

$$E_{ref,dA_r}(x,t) = \sum_{i=1}^{n_{sur}} \sum_{p=1}^{n_i} G_{s,\Delta A_{p,i}}(x,t)\, F_{dA_r - \Delta A_{p,i}}, \tag{A.4.132a}$$

where $n_{sur} \triangleq 4$ is the number of surfaces from which daylight can reach dA_r after *one* reflection inside the room, $n_i \triangleq 4$ is the number of patches located on the i-th surface, and $F_{dA_r - \Delta A_{p,i}}$ is the view factor from dA_r to $\Delta A_{p,i}$ computed using (A.4.124b).

A.4.4.8 Horizontal Illuminance at the Daylight Reference Point

The horizontal daylight illuminance at the daylight reference point can now be computed as

$$E_{day}(x,t) = E_{dir,dA_r}(x,t) + E_{dif,dA_r}(x,t) + E_{ref,dA_r}(x,t). \tag{A.4.132b}$$

A.4.4.9 Daylighting Control

We will now describe the daylighting control which models a lighting system with continuous dimming of the electrical light.

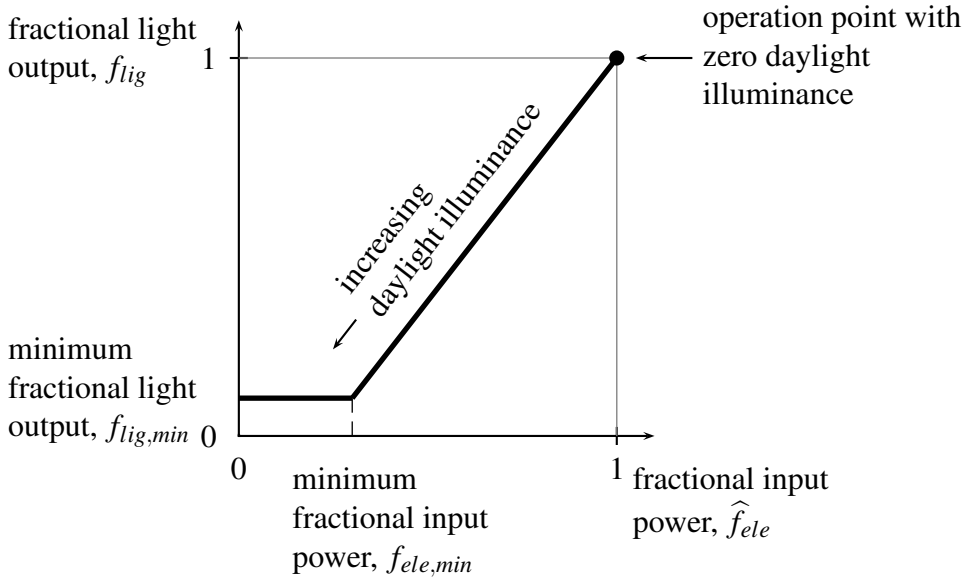


Figure A.19: Power/light curve for continuous dimming according to (A.4.134).

Fractional Light Output Let $r \in \{1, 2\}$ denote the number of the daylight reference point and let $E_{req,r}$ denote the required illuminance at the r-th reference point. Then, for the r-th daylight reference point, the required fractional light output is

$$f_{lig,r}(x,t) = \widetilde{\max}\left(\frac{E_{req,r} - E_{day,r}(x,t)}{E_{req,r}},\ 0\right). \tag{A.4.133}$$

Lighting Control We assume that the lights are controlled using continuous dimming between full fractional light output and a user-specified minimal fractional light output as shown in Fig. A.19. The fractional input power is

$$\widehat{f}_{ele,r}(x,t) = f_{ele,min} + \max\left(0,\ \left(f_{lig,r}(x,t) - f_{lig,min}\right)\frac{1 - f_{ele,min}}{1 - f_{lig,min}}\right), \tag{A.4.134}$$

where $f_{ele,min}$ and $f_{lig,min}$ are user-specified values. Values for $f_{ele,min}$ and $f_{lig,min}$ obtained in field measurements by Rubinstein (1999) are $f_{ele,min} \approx 0.3$ and $f_{lig,min} \approx 0.1$.

To make (A.4.134) once Lipschitz continuously differentiable in x and t, we rewrite it as

$$f_{ele,r}(x,t) = f_{ele,min} + \widetilde{\max}\left(0,\ \left(f_{lig,r}(x,t) - f_{lig,min}\right)\frac{1-f_{ele,min}}{1-f_{lig,min}}\right). \tag{A.4.135}$$

Lighting Power Users can specify what fraction of the room light is controlled by each reference point. Let $f_{Z,r}$ denote this fraction for the r-th reference point. Then, the fractional lighting power for the whole room is

$$M_p(x,t) = \sum_{r=1}^{n_r} f_{ele,r}(x,t)\, f_{Z,r} + \left(1 - \sum_{r=1}^{n_r} f_{Z,r}\right), \tag{A.4.136}$$

where $n_r \triangleq 2$ is the number of daylight reference points. The last term accounts for the fraction of the zone that does not have lighting control. Thus, if $P_{lig,nom}$ denotes the maximum lighting electricity power, then the lighting lighting electricity power is

$$P_{lig}(x,t) = M_p(x,t)\, P_{lig,nom}. \tag{A.4.137}$$

Module Description

We list only parameters and inputs that are required for the daylighting and electric lighting simulation. Parameters such as the room geometry and the surfaces' reflectivity are not listed here since they have already been specified in other models.

Parameter

Variable	Description
r_1	coordinates of 1-st reference point
r_2	coordinates of 2-nd reference point. (Both reference points must be at the same height above ground.)
$\{E_{req,r}\}_{r=1}^{2}$	required illuminance for each reference point

Continued on next column.

Parameter (continued)

Variable	Description
$f_{ele,min}$	minimal fractional (electrical) input power (see Fig. A.19)
$f_{lig,min}$	minimal fractional light output (see Fig. A.19)
$\{f_{Z,r}\}_{r=1}^{2}$	fraction of zone light controlled by r-th reference point

Continued on next page.

Parameter (continued)

Variable	Description
$T_{sa}^{\sphericalangle}(\cdot)$	function for angular dependency of window transmittance, window with activated shading device
$T_{ds}^{\sphericalangle}(\cdot)$	function for angular dependency of window transmittance, window with deactivated shading device

Input

Variable	Description
$G_{glo,hor}(t)$	global horizontal illuminance
$G_{dif,hor}(t)$	diffuse horizontal illuminance
$\theta_s(t)$	solar zenith angle
$\phi_s(t)$	solar azimuth
$y_{sc}(t)$	control signal for shading device

Output

Variable	Description
$M_p(x,t)$	fractional lighting power of the whole zone
$\{E_{day,r}(x,t)\}_{r=1}^{2}$	horizontal daylight illuminance at the daylight reference points
$\{f_{lig,r}(x,t)\}_{r=1}^{2}$	required fractional light output for each reference point
$\{f_{ele,r}(x,t)\}_{r=1}^{2}$	fractional (electric) input power for light controlled by r-th reference point

A.5 Implementation of the Models and the DAE Solver

BuildOpt consists of two parts. The first part, which we will call the *simulation model generator*, parses a text input file with the detailed description of the building geometry, the building materials and the expected occupancy behavior and then generates a simulation model for the particular building.

The second part of BuildOpt, to which our simulation model generator was linked, is the commercial DAE solver DASPK (Brenan et al., 1989; Brown et al., 1994, 1998).

The total size of BuildOpt is 38,000 of C/C++ and Fortran code, of which 30,000 lines (1.2 MB) of C/C++ code represent the simulation model generator and 8,000 lines (0.3 MB) of Fortran 77 code represent the commercial solver DASPK.

The simulation model generator is object-oriented and written in the C++ language (Stroustrup, 2000) using generic algorithms and data structures of the Standard Template Library (STL) (Musser et al., 2001; Stepanov and Lee, 1995).

We believe that using object-oriented rather than procedural models significantly reduced the development time, even though it took us about one man year to develop the 30,000 lines of C++ code. If we were to include the code of the simulation model generator in this documentation using single space formating, the document would grow by 600 pages. The simulation model generator is documented using the doxygen program developed by Dimitri van Heesch.[8]

We obtained good numerical results with the DASPK solver. However, we believe

[8] `http://www.doxygen.org/`

that the computational time could be reduced if we invest more development time for formulating the DAE system so that its Jacobian matrix is band diagonal, which seems to be a doable task but not a trivial task in view of the geometrical complexity of multi-zone buildings. A further reduction in computational time may be achieved if we exploit the sparsity of the DAE system.

If models for HVAC components are added, it may be interesting and necessary to test how numerical solvers that detect state events perform.

A.6 Compiling and Linking BuildOpt

We compiled BuildOpt on a Linux computer with an AMD Athlon processor running RedHat 9.0 with the kernel 2.4.20-8. We used the compiler gcc, version 3.2.2. To compile the source code files, we used the commands

```
g++ -c -static -o src.o src.cpp
```

```
g77 -c -fno-underscoring src.o src.f
```

where `src` is the name of the source file. To link the object files, we used the command

```
g++ -static -o buildoptsimkernel OBJECTLIST (continued on next line)
```

```
  -lfrtbegin -lg2c -lc -lm -lgcc -lstdc++
```

where `OBJECTLIST` is a list of all object files.

Appendix B

BuildOpt – Validation

B.1 Thermal Model

B.1.1 Introduction

To validate BuildOpt's thermal models, we used the ANSI/ASHRAE Standard 140-2001 (ASHRAE, 2001). This standard has been developed to identify and to diagnose differences in predictions for whole building energy simulation software that may possibly be caused by software errors. The test consists of a series of specified test cases that progress systematically from the simple to the relatively realistic case. Output values, such as annual energy consumptions, peak loads, annual minimum, average and maximum room air temperatures, and some hourly data are compared to the results of other building energy simulation programs, namely to ESP, BLAST, DOE2.1D, SUNCODE, SERIRES, S3PAS, TRNSYS and TASE.

We used the so-called "basic test cases" to validate BuildOpt. The basic test cases test the ability to model such combined effects as thermal mass, solar gains, window-shading devices, internally generated heat, outside air infiltration, sunspaces and thermostat control.

B.1.2 Specification of the Test Cases

We will now present a brief overview of the basic test cases. We refer the reader to ASHRAE (2001) for a more detailed description.

There are three series of basic cases:

- The low mass basic tests (Cases 600 through 650), which use a light weight building envelope.
- The high mass basic tests (Cases 900 through 960), which use heavy weight walls, a heavy weight floor and include an additional building configuration with a sunspace.
- The free-float basic tests (Cases with ending "FF"), which have no heating or cooling system.

Tab. B.1.2 lists all the base cases. For BuildOpt, only the cases in non-italic font were simulated. Case 630 and Case 930 were not simulated because BuildOpt has no model for vertical shading devices. Case 650 and Case 950 were not simulated because BuildOpt has no model for night ventilation.

Fig. B.1 shows the building configuration used for the Cases 600, 610, 640, 650, 900, 930, 940 and 950, Fig. B.2 shows the building configuration used for the Cases 620, 630, 920 and 930, and Fig. B.3 shows the building configuration with sunspace used for the Case 960. The roof and the external shading devices are not shown.

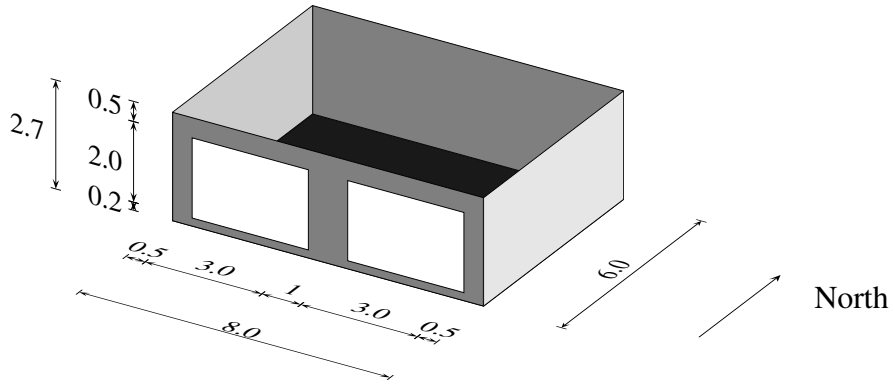


Figure B.1: Isometric view of building with south windows. The roof is not shown.

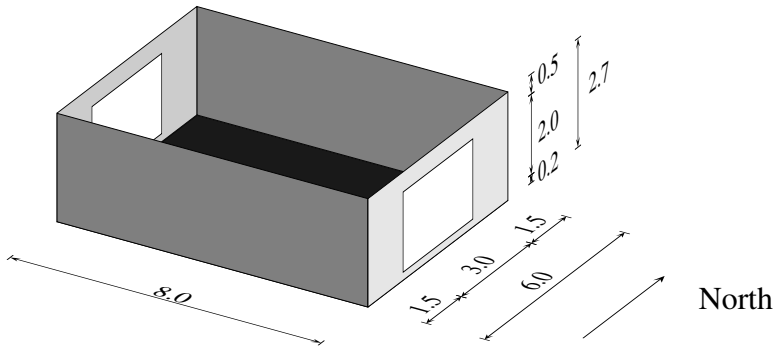


Figure B.2: Isometric view of building with west and east windows. The roof is not shown.

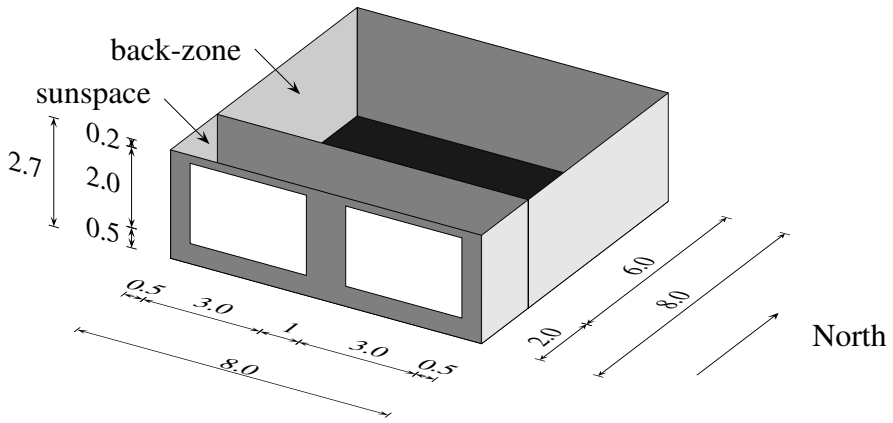


Figure B.3: Isometric view of building with sunspace. The roof is not shown.

Case	H, C, V °C	Mass	INTGEN W	INFILTR 1/h	INT IR EMIT	EXT IR EMIT	INT SW ABSORPT	EXT SW ABSORPT	Glass m^2	ORIENT	SHADE	COMMENTS
600	20, 27	L	200	0.5	0.9	0.9	0.6	0.6	12	S	no	Case 600 tests south solar transmission.
610	20, 27	L	200	0.5	0.9	0.9	0.6	0.6	12	S	1.0 mH	Cases 610, 600 test south overhang.
620	20, 27	L	200	0.5	0.9	0.9	0.6	0.6	6, 6	E, W	no	Cases 620, 600 test east and west solar transmittance/incidence.
630	*20, 27*	*L*	*200*	*0.5*	*0.9*	*0.9*	*0.6*	*0.6*	*6, 6*	*E, W*	*1.0 mHV*	*Cases 630, 620 test east and west overhangs and fins.*
640	SETBACK	L	200	0.5	0.9	0.9	0.6	0.6	12	S	no	Cases 640, 600 test night setback.
650	*27, V*	*L*	*200*	*0.5*	*0.9*	*0.9*	*0.6*	*0.6*	*12*	*S*	*no*	*Case 650, 600 test venting.*
900	20, 27	H	200	0.5	0.9	0.9	0.6	0.6	12	S	no	Cases 900, 600 test thermal mass and solar interaction.
910	20, 27	H	200	0.5	0.9	0.9	0.6	0.6	12	S	1.0 mH	Cases 910, 900 test south overhang/mass interaction.
920	20, 27	H	200	0.5	0.9	0.9	0.6	0.6	6, 6	E, W	no	Cases 920, 900 test east and west transmittance/mass interaction.
930	*20, 27*	*H*	*200*	*0.5*	*0.9*	*0.9*	*0.6*	*0.6*	*6, 6*	*E, W*	*1.0 mHV*	*Cases 930, 920 test east and west shading/mass interaction.*
940	SETBACK	H	200	0.5	0.9	0.9	0.6	0.6	12	S	no	Cases 940, 900 test setback/mass interaction.
950	*27, V*	*H*	*200*	*0.5*	*0.9*	*0.9*	*0.6*	*0.6*	*12*	*S*	*no*	*Cases 950, 900 test venting/mass interaction.*
960	2ZONE, SS	See specification in Test Procedures										Case 960 tests passive solar/interzonal heat transfer.
600FF 900FF *650FF* *950FF*	NONE NONE *NONE, V* *NONE, V*	These cases, labeled FF (interior temperatures free-float), are exactly the same as the corresponding non-FF cases except there are no mechanical heating or cooling systems.										

Table B.1: Description of base cases, reproduced from ASHRAE (2001). (For BuildOpt, only the cases in non-italic font were simulated.)

B.1.3 Modeling Notes

We will now present the modeling notes in the format defined in ASHRAE (2001).

STANDARD 140 OUTPUT FORM - MODELING NOTES

INSTRUCTIONS: See Annex A2.

SOFTWARE: BuildOpt

VERSION: 1.0.1

DOCUMENT BELOW THE MODELING METHODS USED IF ALTERNATIVE MODELING METHODS OR ALGORITHMS ARE AVAILABLE IN THE SOFTWARE BEING TESTED.

Continued on next page.

Continued from previous page.

Simulated Effect:

Complexity of the physical models.

Optional Settings or Modeling Capabilities:

`MODELCOMPLEXITY`*, any integer between 0 and 4 (inclusive). If 0 is selected, then the coarsest models will be used, and if 4 is selected, the most detailed models will be used.*

Setting or Capability Used:

```
MODELCOMPLEXITY, 4;
```

Physical Meaning of Option Used:

Perez's model is used to compute the diffuse solar irradiation. The convective heat transfer coefficients at room-side surfaces are computed as a function of the temperature difference.

Continued on next page.

Continued from previous page.

Simulated Effect:

Heat conduction in opaque materials.

Optional Settings or Modeling Capabilities:

`ELEMENTS`, *any non-zero natural number.*

Setting or Capability Used:

```
ELEMENTS,
  10;
```

Physical Meaning of Option Used:

The number of elements used in the Galerkin method is equal to 10 *for a reference material of* 20 cm *concrete. For other materials, the number of elements is adjusted based on a Fourier number similarity.*

Continued on next page.

Continued from previous page.

Simulated Effect:

Settings of differential algebraic equation solver.

Optional Settings or Modeling Capabilities:

See BuildOpt documentation.

Setting or Capability Used:

```
DASPK,
 1E-4,     ! Relative tolerance.
 1E-4,     ! Absolute tolerance.
  600,     ! HMAX, maximum step size for time integration.
 10,       ! H0, initial step size for time integration.
 -1,       ! Maximum order of integration scheme
           ! (-1: use default).
 Y_d,      ! Method to compute consistent initial conditions.
           ! If Y_d, then Y_d is given, Y_a and Y'_d
           ! are computed.
           ! If Y', then Y is computed.
 INCLUDE,  ! Flag to include algebraic variables in
           ! the error test.
```

Continued on next page.

Continued from previous page.

```
10,      ! MXNIT, maximum number of Newton iterations
         ! for IV computation.
100,     ! MXNJ,  maximum number of Jacobian
         ! evaluations for IV computation.
-1,      ! MXNH, maximum number of values of the
         ! artificial step size parameter H in
         ! the IV computation (-1: use default).
         ! Only used if Y_d is selected above.
OFF,     ! Line search flag (ON, OFF).
-1,      ! EPINIT, swing factor in the Newton
         ! iteration convergence test.
0,       ! Extra printing in IV computation (0: off).
3600;    ! Report interval in seconds.
```

B.1.4 Results

B.1.4.1 Annual Loads and Peak Loads

For the high mass test cases, the annual cooling load and the annual peak cooling are on the lower side compared to the results of the other tested programs, in particular for Case 900 and Case 940. For these cases, the annual cooling load and the annual peak cooling are about 15% lower than the ones of the other tested programs (see Fig. B.9, Fig. B.11, Fig. B.17 and Fig. B.19). For the other test cases, BuildOpt's results agree well with the results of the other tested programs.

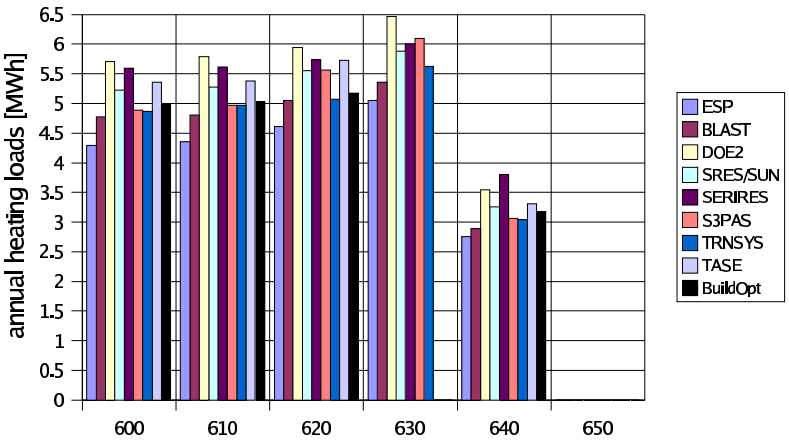


Figure B.4: Annual heating loads for low mass buildings. Case 630 was not simulated with BuildOpt because it has no model for vertical shading devices. In Case 650, the heating system is switched off.

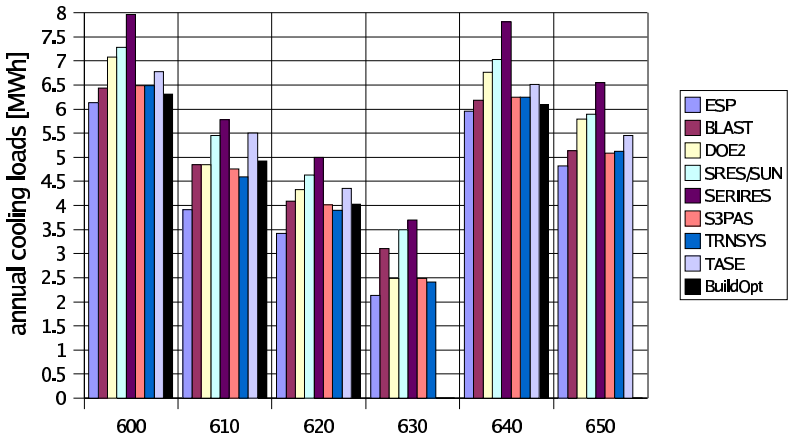


Figure B.5: Annual sensible cooling loads for low mass buildings. Case 630 was not simulated with BuildOpt because it has no model for vertical shading devices. Case 650 was not simulated with BuildOpt because it has no model for time-scheduled ventilation.

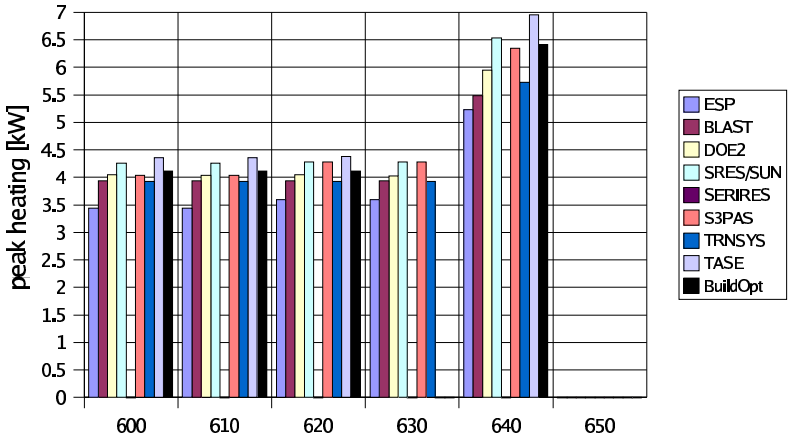


Figure B.6: Peak heating loads for low mass buildings. Case 630 was not simulated with BuildOpt because it has no model for vertical shading devices. In Case 650, the heating system is switched off.

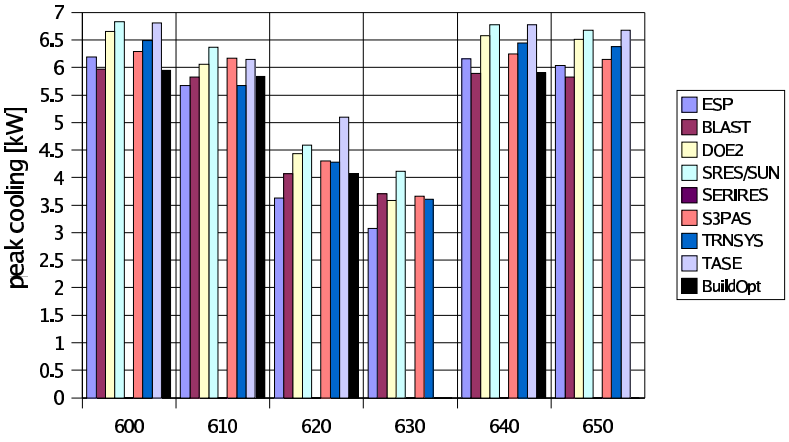


Figure B.7: Annual peak sensible cooling loads for low mass buildings. Case 630 was not simulated with BuildOpt because it has no model for vertical shading devices. Case 650 was not simulated with BuildOpt because it has no model for time-scheduled ventilation.

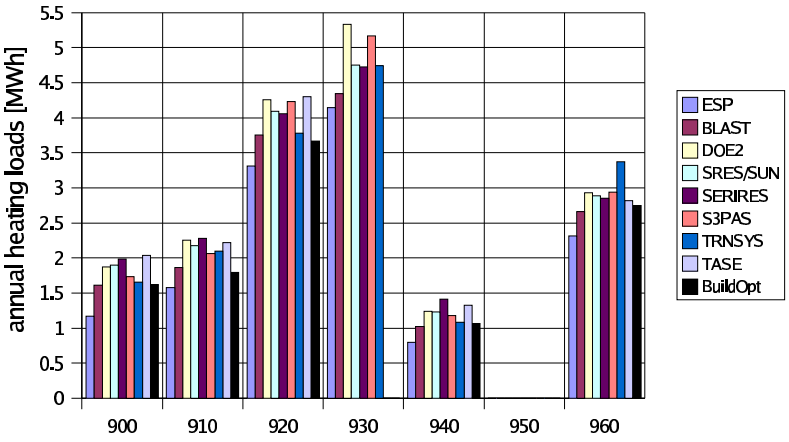


Figure B.8: Annual heating loads for high mass buildings. Case 930 was not simulated with BuildOpt because it has no model for vertical shading devices. In Case 950, the heating system is switched off.

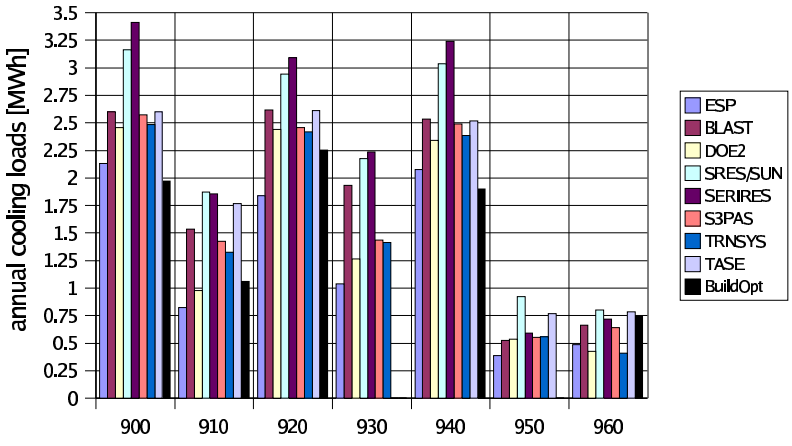


Figure B.9: Annual sensible cooling loads for high mass buildings. Case 930 was not simulated with BuildOpt because it has no model for vertical shading devices. Case 950 was not simulated with BuildOpt because it has no model for time-scheduled ventilation.

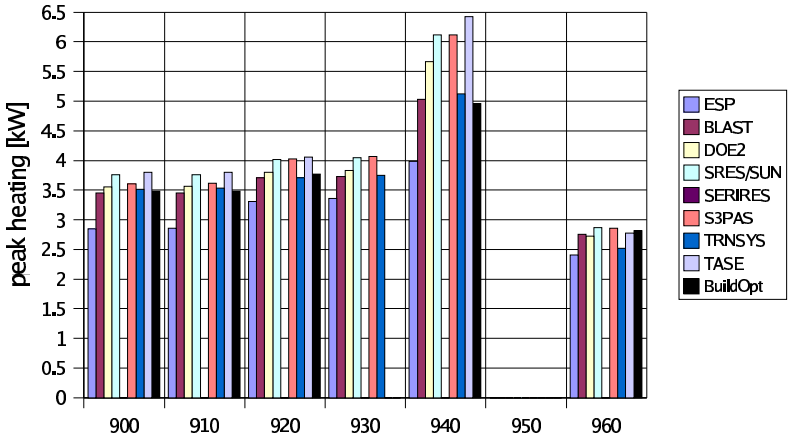


Figure B.10: Peak heating loads for high mass buildings. Case 930 was not simulated with BuildOpt because it has no model for vertical shading devices. In Case 950, the heating system is switched off.

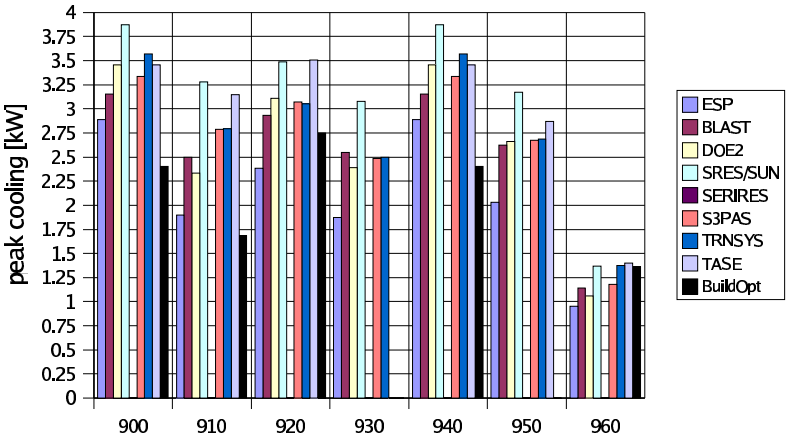


Figure B.11: Annual peak sensible cooling loads for high mass buildings. Case 930 was not simulated with BuildOpt because it has no model for vertical shading devices. Case 950 was not simulated with BuildOpt because it has no model for time-scheduled ventilation.

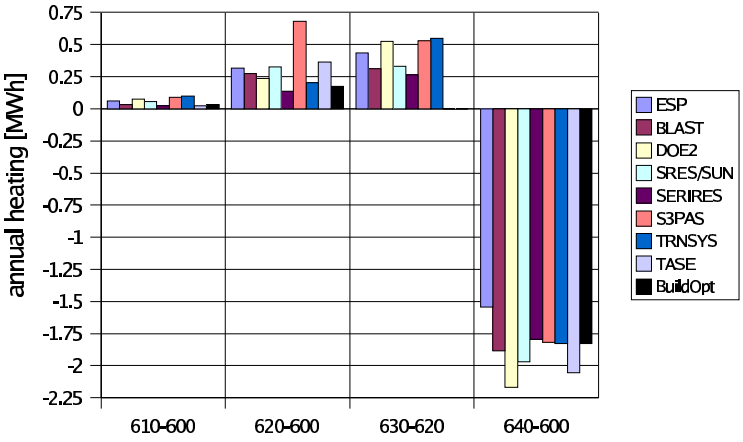


Figure B.12: Sensitivity of annual heating load for low mass buildings. Case 630 was not simulated with BuildOpt because it has no model for vertical shading devices.

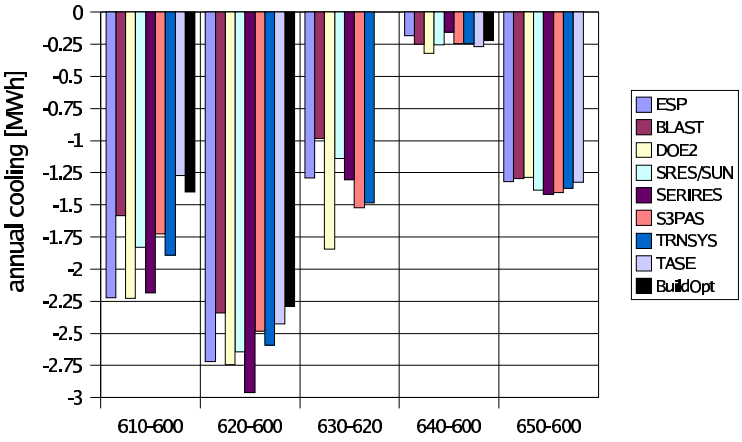


Figure B.13: Sensitivity of annual sensible cooling load for low mass buildings. Case 630 was not simulated with BuildOpt because it has no model for vertical shading devices. Case 650 was not simulated with BuildOpt because it has no model for time-scheduled ventilation.

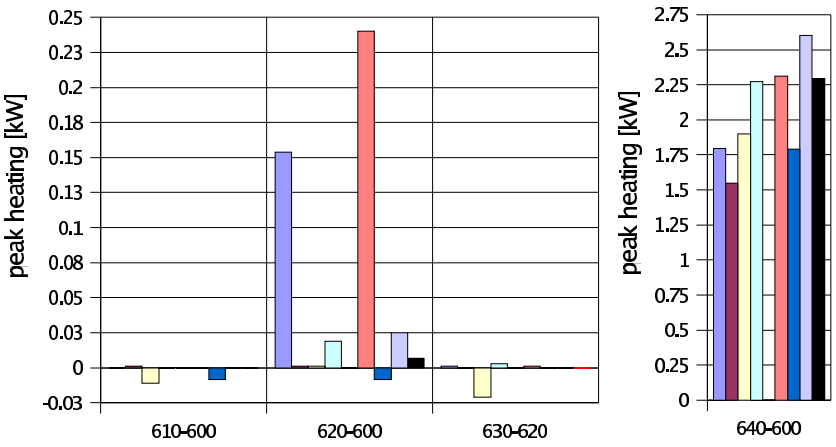


Figure B.14: Sensitivity of peak heating load for low mass buildings. Case 630 was not simulated with BuildOpt because it has no model for vertical shading devices.

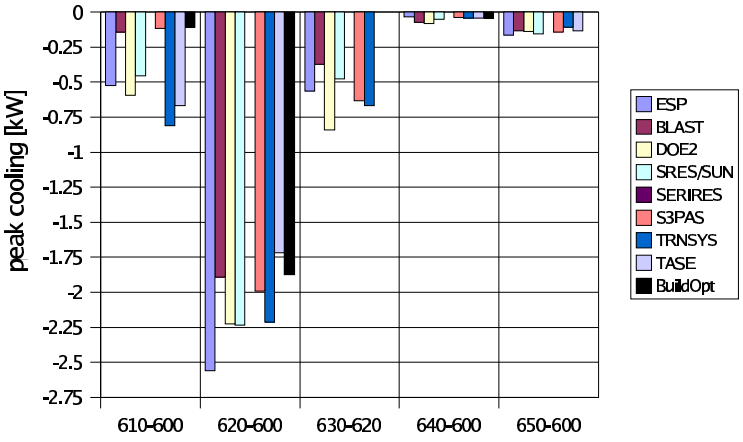


Figure B.15: Sensitivity of peak sensible cooling load for low mass buildings. Case 630 was not simulated with BuildOpt because it has no model for vertical shading devices. Case 650 was not simulated with BuildOpt because it has no model for time-scheduled ventilation.

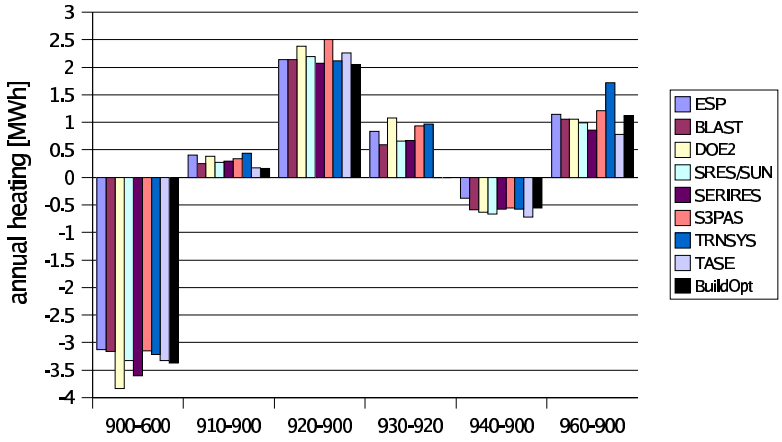


Figure B.16: Sensitivity of annual heating load for high mass buildings. Case 930 was not simulated with BuildOpt because it has no model for vertical shading devices.

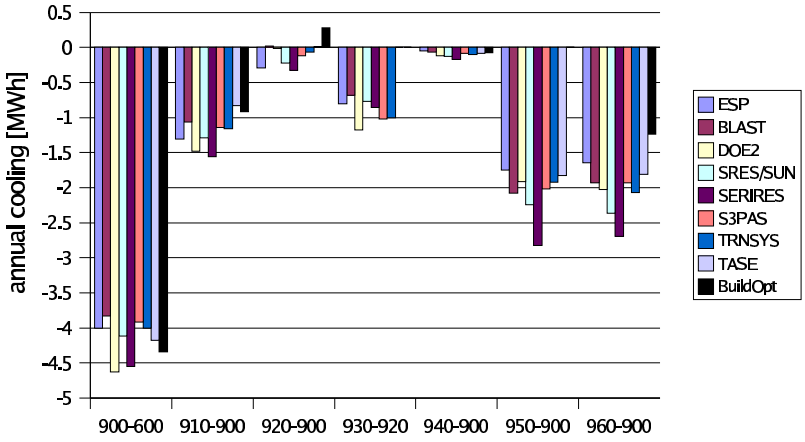


Figure B.17: Sensitivity of annual sensible cooling load for high mass buildings. Case 930 was not simulated with BuildOpt because it has no model for vertical shading devices. Case 950 was not simulated with BuildOpt because it has no model for time-scheduled ventilation.

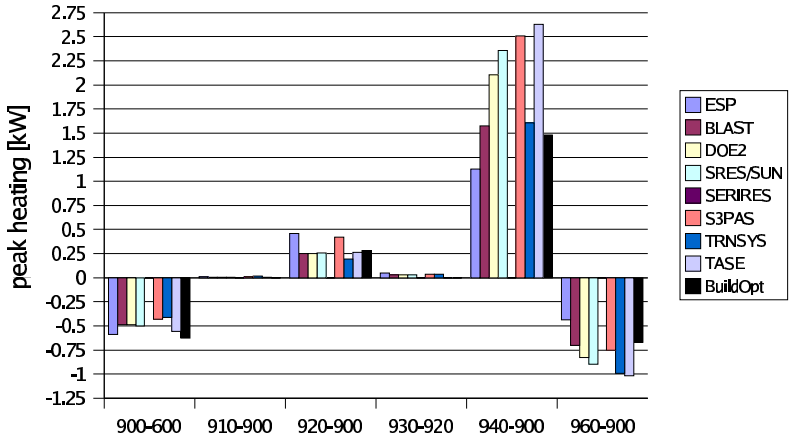


Figure B.18: Sensitivity of peak heating load for high mass buildings. Case 930 was not simulated with BuildOpt because it has no model for vertical shading devices.

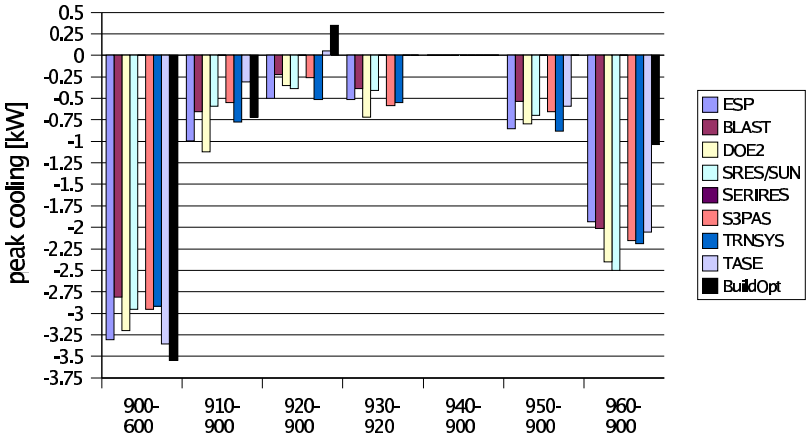


Figure B.19: Sensitivity of peak sensible cooling load for high mass buildings. Case 930 was not simulated with BuildOpt because it has no model for vertical shading devices. Case 950 was not simulated with BuildOpt because it has no model for time-scheduled ventilation.

B.1.4.2 Indoor Air Temperatures

The minimum indoor air temperature computed by BuildOpt is for the low mass test cases in the range of the results of the other tested programs, but it is for the high mass test cases 1 K lower. The average indoor air temperature computed by BuildOpt is within the range of the results of the other tested programs. The maximum indoor air temperature computed by BuildOpt is for the high mass test cases 3 K lower and for the low mass test cases 4 K lower than the results of the other tested programs. Consequently, for Case 900FF, there are fewer hours with a high indoor air temperature compared to the other tested programs, but the number of hours with low indoor air temperature coincides well with the results of the other tested programs (see Fig. B.23). On January 4, the hourly free float temperature is in the range of results of the other tested programs for both, the low mass test case (see Fig. B.24) and the high mass test case (see Fig. B.25).

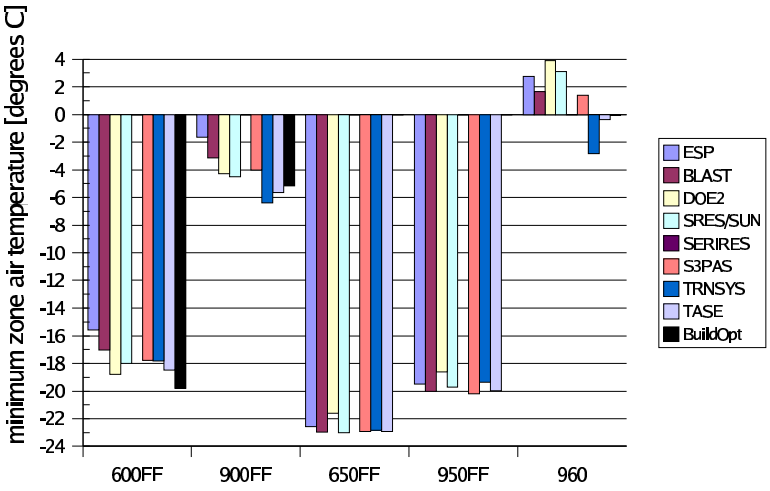


Figure B.20: Minimum hourly annual temperature for free-float test cases. Case 650FF was not simulated with BuildOpt because it has no model for time-scheduled ventilation. Case 950FF was not simulated with BuildOpt because it has no model for time-scheduled ventilation.

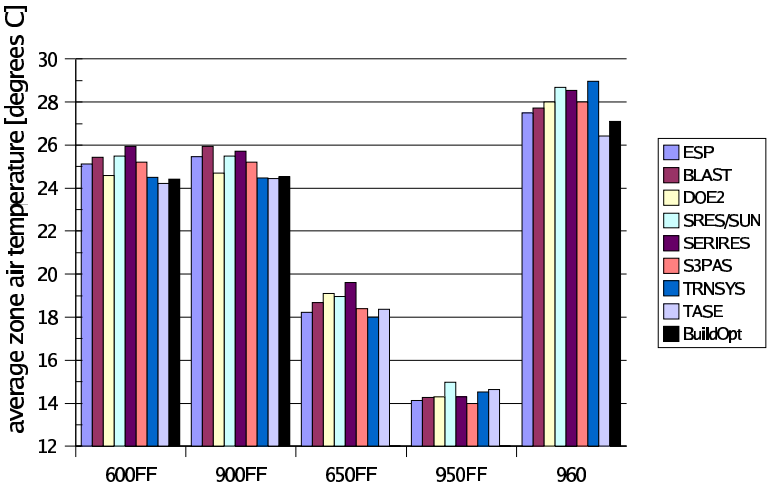


Figure B.21: Average hourly annual temperature for free-float test cases. Case 650FF was not simulated with BuildOpt because it has no model for time-scheduled ventilation. Case 950FF was not simulated with BuildOpt because it has no model for time-scheduled ventilation.

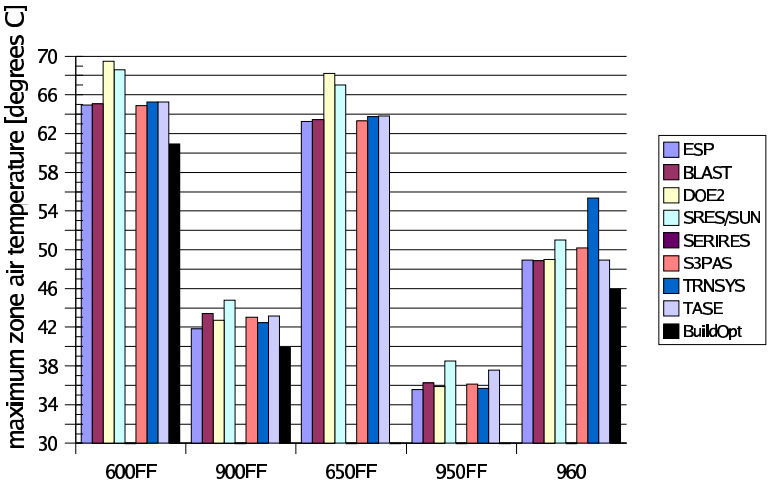


Figure B.22: Maximum hourly annual temperature for free-float test cases. Case 650FF was not simulated with BuildOpt because it has no model for time-scheduled ventilation. Case 950FF was not simulated with BuildOpt because it has no model for time-scheduled ventilation.

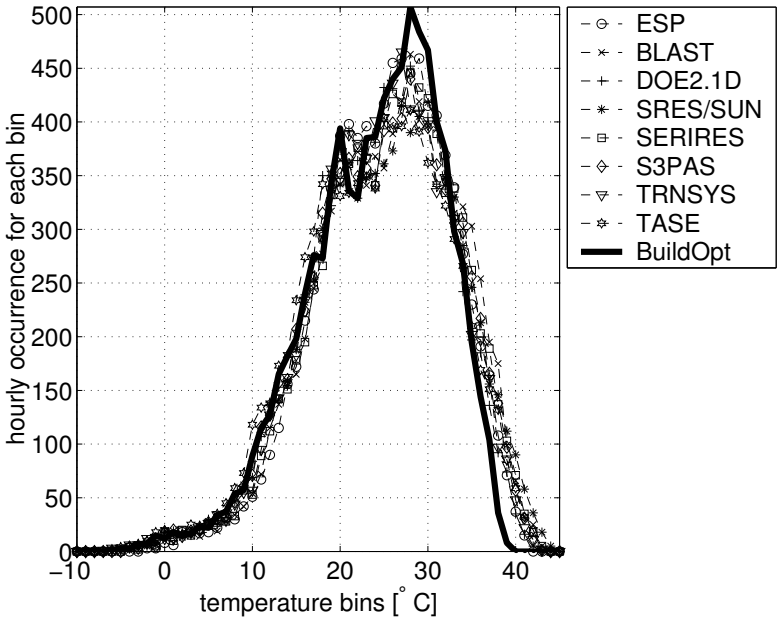


Figure B.23: Annual hourly temperature frequency for each 1°C bin for Case 900FF.

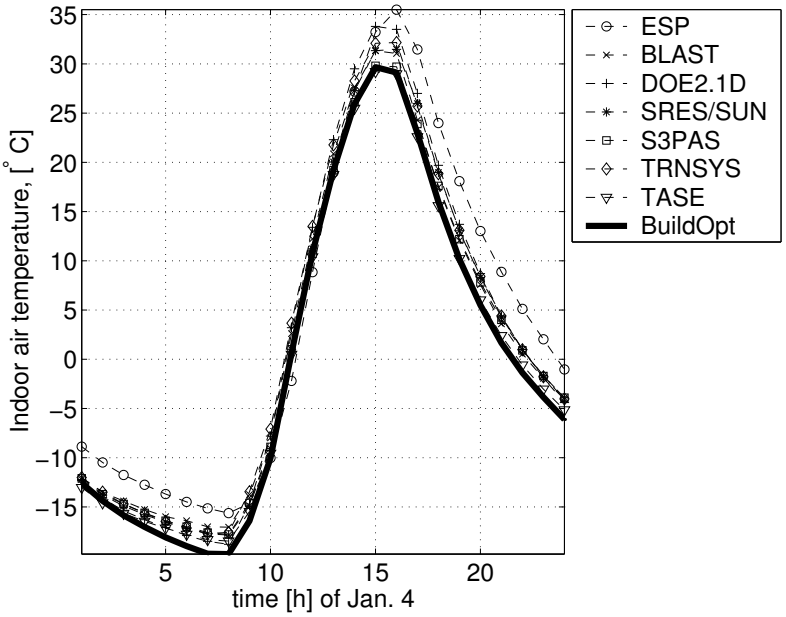


Figure B.24: Hourly free float temperatures on January 4 for low mass building (Case 600FF).

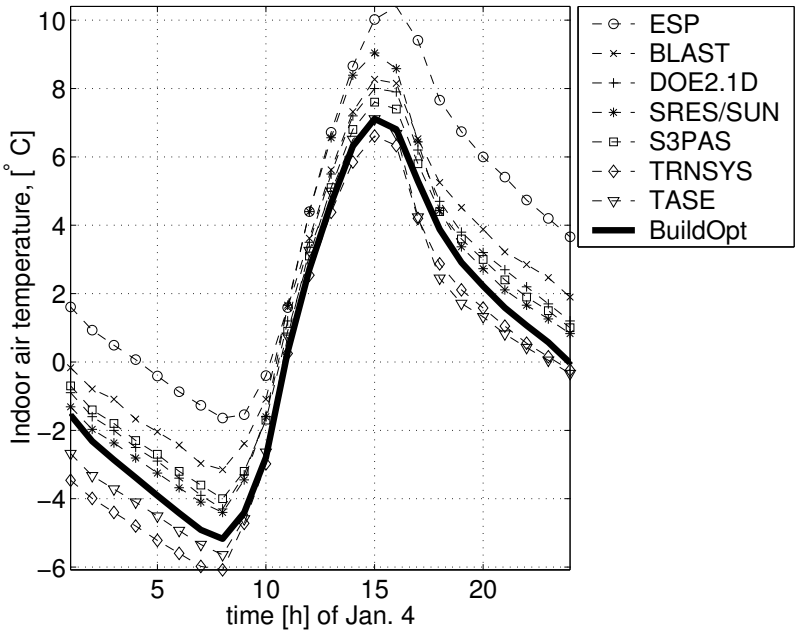


Figure B.25: Hourly free float temperatures on January 4 for heavy mass building (Case 900FF).

B.1.4.3 Incident and Transmitted Solar Radiation

The annual incident and transmitted solar radiation computed by BuildOpt coincides with the results of the other tested programs. However, on March 5 and July 27, which are cloudy days, BuildOpt's hourly incident solar radiation on a south surface is shifted by 1 hour compared to the results of the other tested programs (see Fig. B.29 and Fig. B.30). The values of BuildOpt are nearly symmetric to 12 : 00, whereas the values of the other tested programs are nearly symmetric to 13 : 00. For this building site, on March 5, 1990, the sun is highest at 12 : 11, and on July 27, 1990, it is highest at 12 : 06 (neglecting daylight savings time).[1] Thus, the solar radiation should nearly be symmetric to 12 : 00, as computed by BuildOpt, rather than to 13 : 00, as computed by the other tested programs.

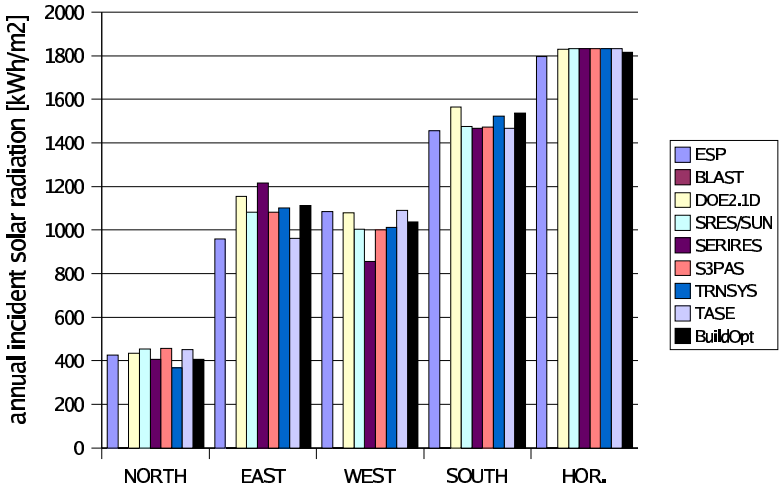


Figure B.26: Annual incident solar radiation.

[1] See, for example, `http://aa.usno.navy.mil/data/docs/RS_OneDay.html`.

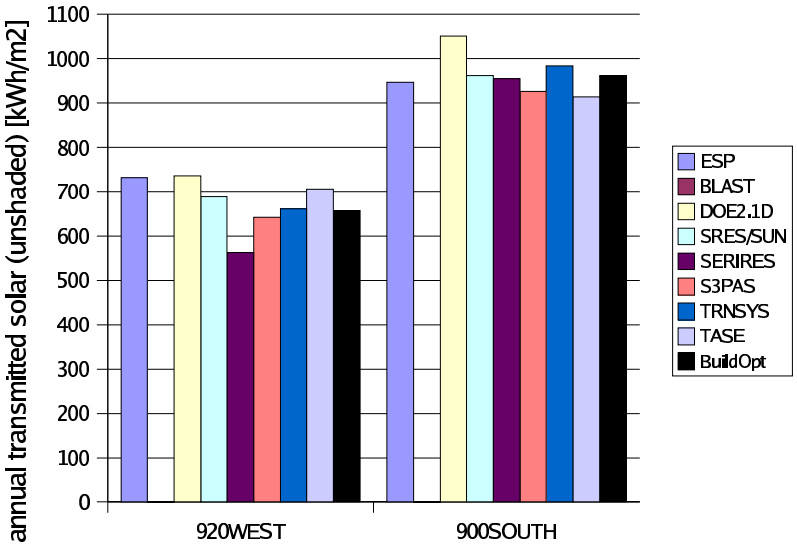


Figure B.27: Annual transmitted solar radiation with unshaded windows.

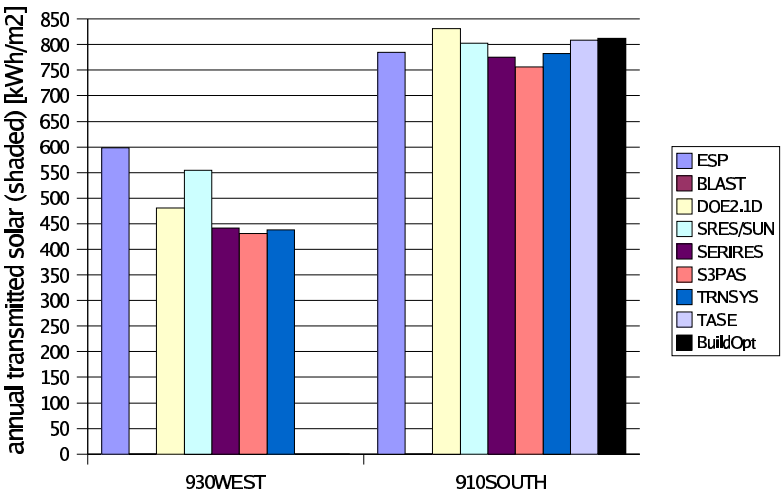


Figure B.28: Annual transmitted solar radiation with shaded windows. Case 930 was not simulated with BuildOpt because it has no model for vertical shading devices.

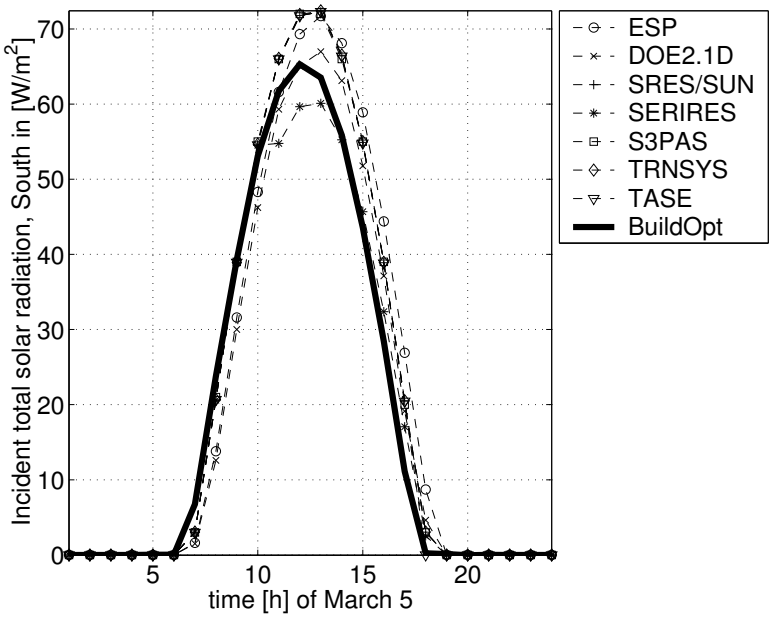


Figure B.29: Hourly incident solar radiation on a cloudy day (March 5) on the south facing surface.

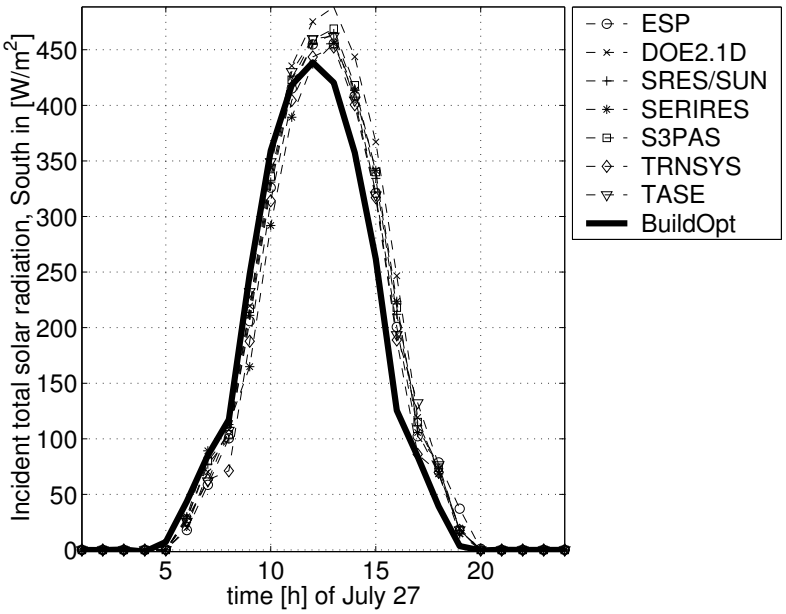


Figure B.30: Hourly incident solar radiation on a clear day (July 27) on the south facing surface.

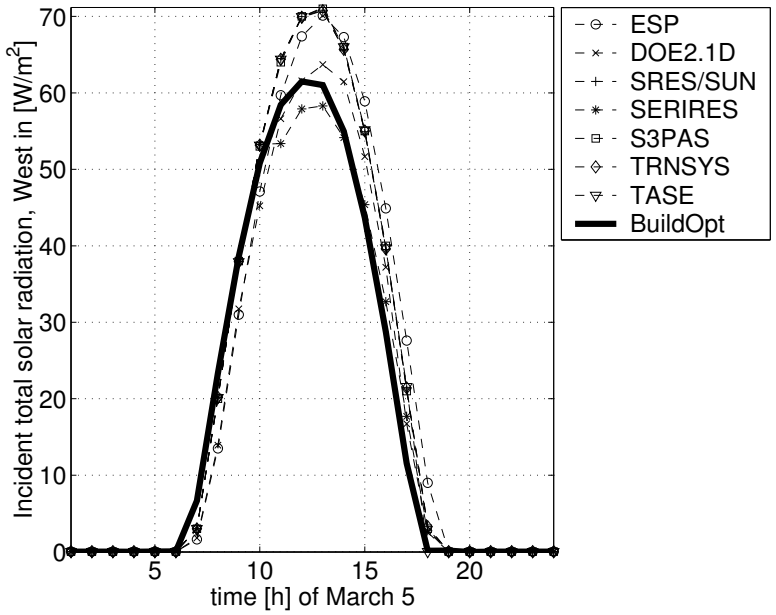


Figure B.31: Hourly incident solar radiation on a cloudy day (March 5) on the west facing surface.

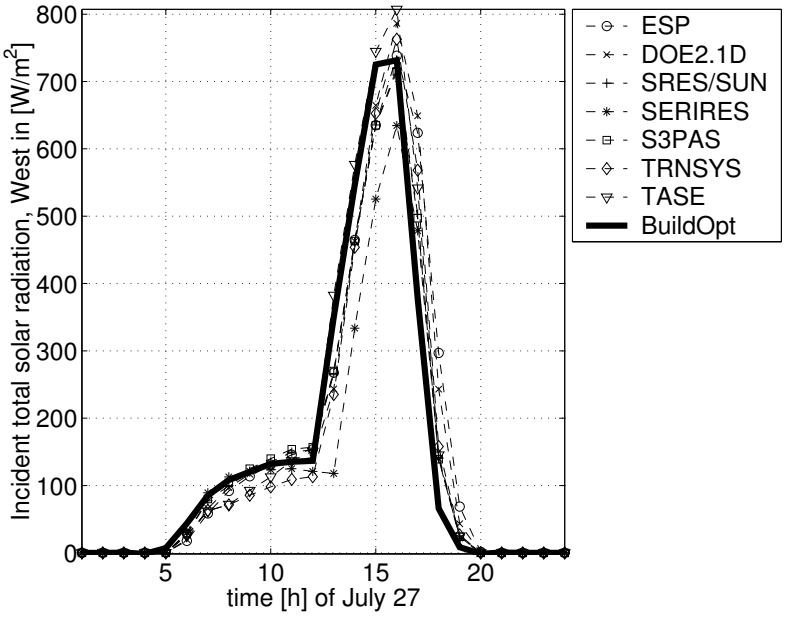


Figure B.32: Hourly incident solar radiation on a clear day (July 27) on the west facing surface.

B.1.4.4 Solar Transmissivity and Shading Coefficients

The solar transmissivity and the shading coefficients computed by BuildOpt coincide with the results of the other tested programs.

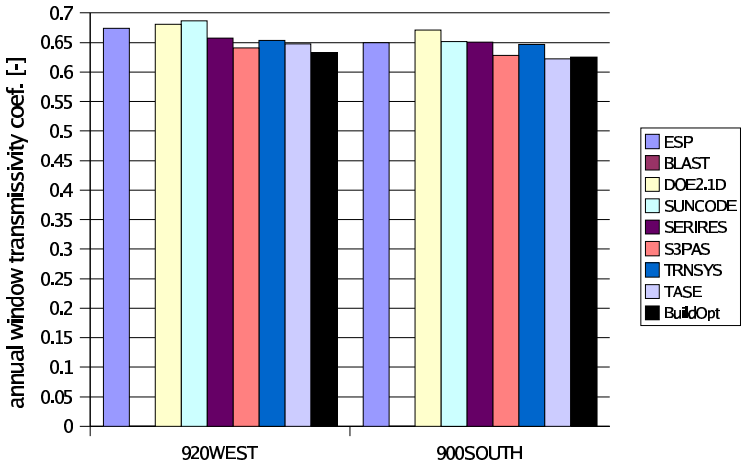


Figure B.33: Annual transmissivity coefficient of windows. This is the ratio between the unshaded transmitted solar radiation and the incident solar radiation.

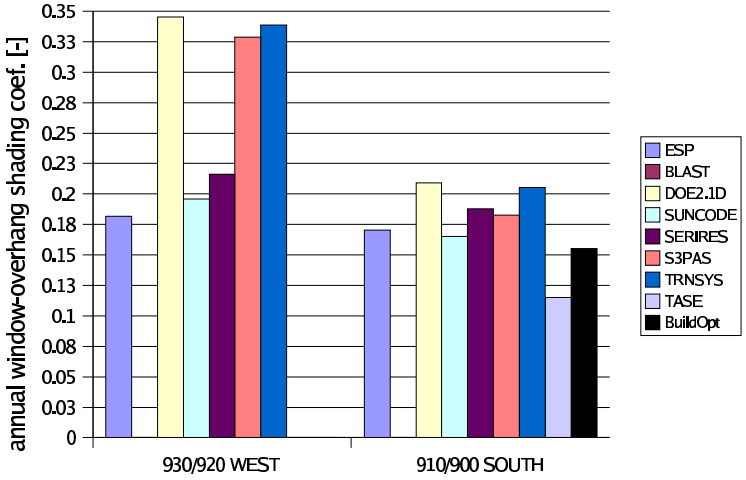


Figure B.34: Annual overhang and fin shading coefficients. The shading coefficient is defined as one minus the ratio between the shaded transmitted solar radiation and the unshaded transmitted solar radiation. Case 930 was not simulated with BuildOpt because it has no model for vertical shading devices.

B.1.4.5 Hourly Heating and Cooling Power

The hourly heating and cooling power coincide with the results of the other tested programs.

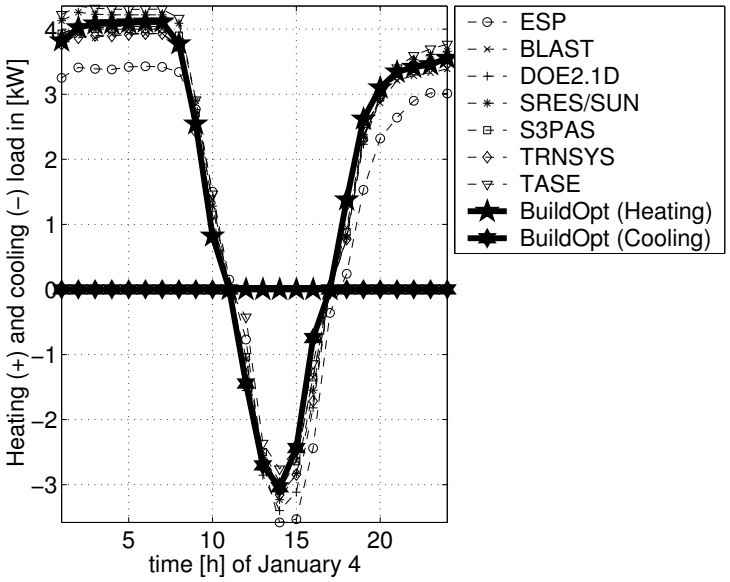


Figure B.35: Hourly heating and cooling power on January 4 for low mass building (Case 600).

B.1.5 Conclusions

BuildOpt's results coincide for most tests with the results of the other tested programs. The largest differences are for the high mass test cases. For these tests, the annual cooling load and the annual peak cooling computed by BuildOpt is low compared to the values of the other tested programs. Also, for the free floating cases, the maximum and the minimum indoor air temperature computed by BuildOpt are low. However, the differences are relatively small, and we do not think that they are caused by a modeling error or a programming error.

B.2 Daylighting Model

B.2.1 Introduction

For the validation of the daylighting model, we used the benchmark tests from Laforgue (1997) and Fontoynont et al. (1999) that have been produced in the Task 21 of the International Energy Agency (IEA) Solar Heating & Cooling Program. These benchmark tests compare the values of the daylight factors for rectangular room configurations with increasing complexity. (The daylight factors are defined as the ratio of the illuminance on indoor surfaces at specific locations to the simultaneous horizontal illuminance on the roof.) The here presented daylight factors were obtained either by measurement, by using analytical computations, or by using different commercially available detailed daylighting simulation programs.

All test cases have a room aperture with no glass. Hence, for the daylight validation, we set the light transmittance of the window to one for all incidence angles. However, we remind that if a room has a window, then BuildOpt uses the same window model for the daylight transmittance and for the solar transmittance, with possibly different input data. This window model was validated in Section B.1.

Due to the model simplifications in BuildOpt's daylighting model, we used only the results for side-lit rooms with an isotropic sky, and we compared only daylight factors

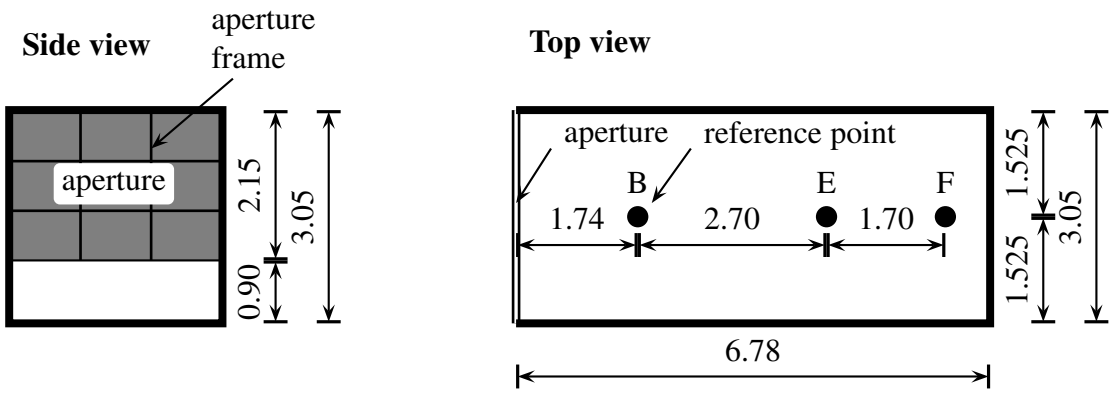


Figure B.36: LESO scale model. The gray area in the left-hand figure shows the location of the aperture, and the black dots in the right-hand figure show the location of the daylight reference points.

on horizontal surfaces.

B.2.2 Specification of the Test Cases

B.2.2.1 LESO Scale Model

For the first set of test cases, we used data from the scale model of the Laboratoire d'Energy Solaire (LESO) of the Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland. Fig. B.36 shows the scale model with the location of the daylight reference points that we used in our validation. All daylight reference points are 0.75 m above the floor. Fontoynont et al. (1999) report that the frame of the aperture was not simulated with any of the software. For the room and ground reflectance, we used the reflectance values shown in Tab. B.2.[2]

[2]Laforgue (1997) lists in the test specification for cases (a) and (b) reflectance for all room surfaces and for the ground of 0.02, in contradiction to the values listed in the result tables, which are zero for all room surfaces and for the ground. The difference between the results is small.

	case (a)	case (b)	case (c)	case (d)
Ceiling	0	0	0.83	0.83
Walls	0	0	0.60	0.60
Floor	0	0	0.28	0.28
Ground	0	0.29	0	0.29

Table B.2: Reflectance values for the LESO scale model.

B.2.2.2 CSTB/ECAD Scale Model

For the second set of test cases, we used data from the scale model of the Centre Scientifique du Bâtiment (CSTB), France. Fig. B.37 shows the scale model and the location of the reference points that we used in our validation. We used the reflectance values shown in Tab. B.3.[3]

	case (a) and (b)	case (c) and (d)
Ceiling	0	0.70
Walls	0	0.50
Floor	0	0.30
Ground	0	0.30

Table B.3: Reflectance values for the CSTB scale model.

For case (a), the daylight reference points were located on the floor and for case (b), they were located 0.8 m above the floor. For case (c), the daylight reference points were for Genelux and BuildOpt on the floor, for SUPERLITE 0.08 m above the floor, and at an unspecified height for the CSTB sim results. For case (d), the daylight reference points were 0.8 m above the floor for all tabulated values.

[3]The ground reflectance is not specified in Laforgue (1997). However, from the daylight factors that are tabulated in Laforgue (1997), one can conclude that for cases (a) and (b), the ground reflectance must be zero. For cases (c) and (d), the results in Laforgue (1997) were obtained by using a ground reflectance of 0.30 (Carroll, 2003). This value accounted for the fact that the 1:10 scale model was placed on a 1 m high table, and hence, the view factor to the sky was as in a room on the 4-th floor. Consequently, an observer that looks downward from the ceiling may see part of the illuminated sky.

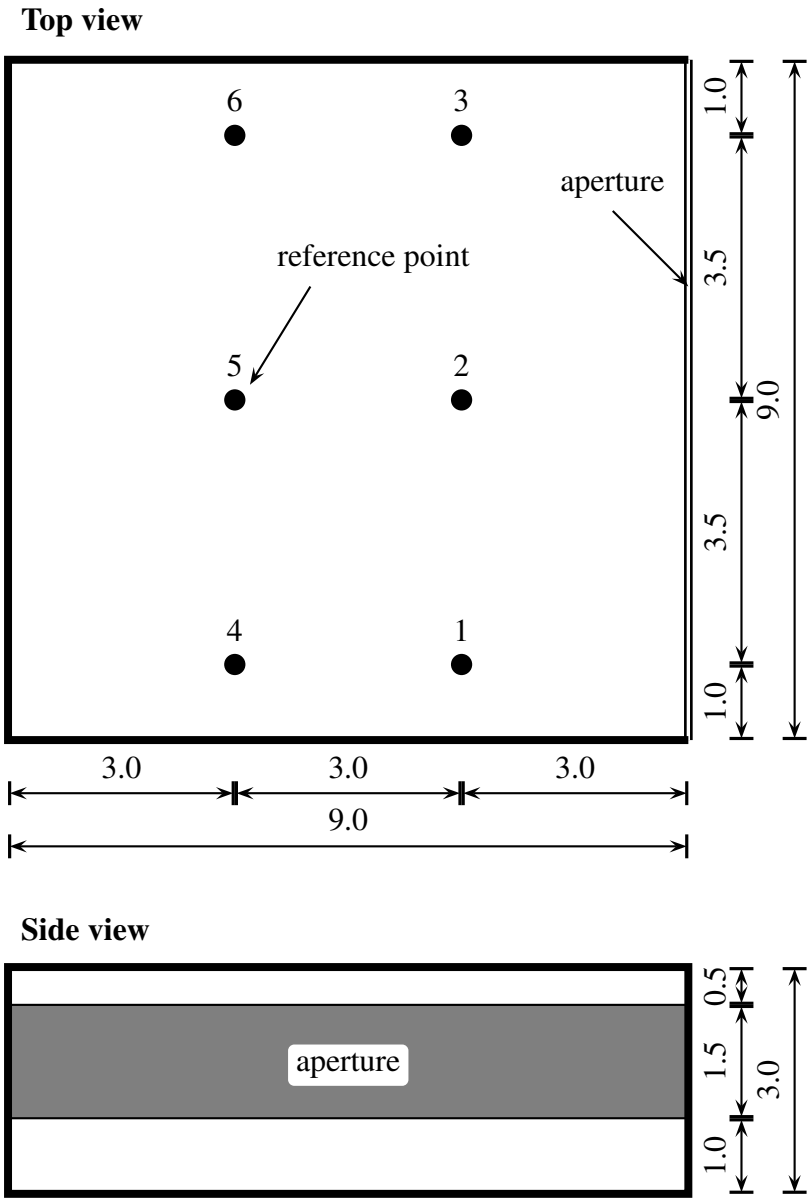


Figure B.37: CSTB scale model. The black dots in the top figure show the location of the daylight reference points, and the gray area in the bottom figure shows the location of the aperture.

B.2.3 Results

B.2.3.1 LESO Scale Model

Tab. B.4 and Fig. B.38 show the daylight factors for all benchmark tests for the LESO scale model that we used in the BuildOpt validation. For the cases (a) and (b) in which all room surfaces are non-reflecting, BuildOpt's daylight factors are within the range of the other results at the reference points E and F which are at a distant of 4.44 m and 6.14 m from the aperture. For the reference point B, which is at a distant of 1.74 m from the aperture, the daylight factor computed by BuildOpt is about 3.5% higher in absolute value than the values listed in Laforgue (1997). For the cases (c) and (d), which used a room with reflecting surfaces, the daylight factors at all reference points are within the range of the other tabulated results. For the daylight reference points E and F, which are far away from the aperture, the daylight factors computed by BuildOpt are on the lower side compared to the other tabulated results. This is because with increasing distance between daylight reference point and aperture, more of the daylight that arrives at the reference point underwent multiple reflections. However, BuildOpt computes only the first reflection, and consequently, it underestimates the daylight illuminance for locations far away from the aperture.

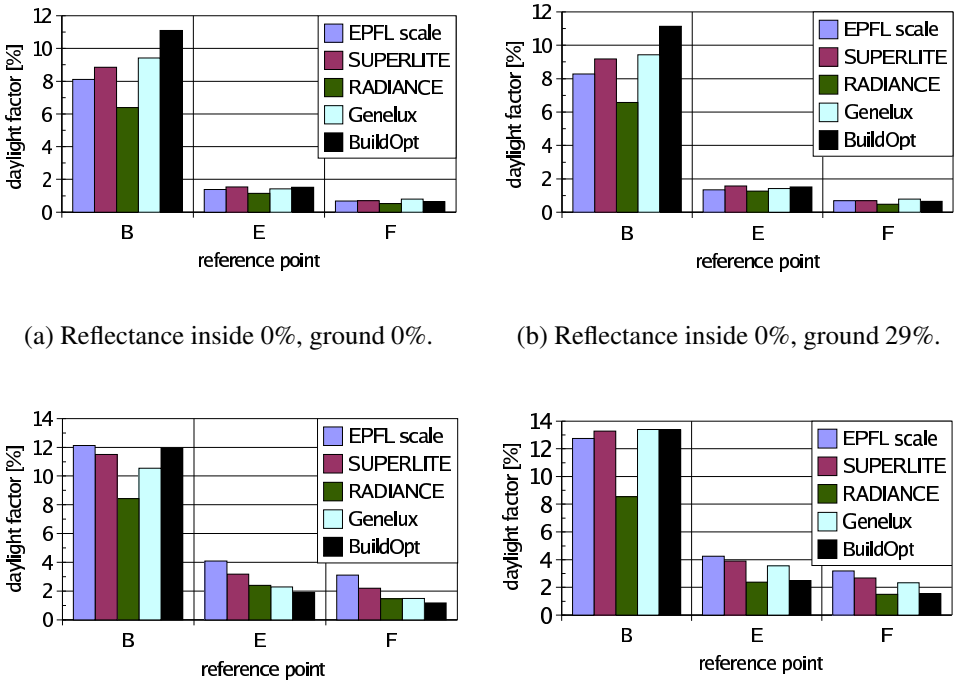
Case (a)					
Point	EPFL scale	SUPERLITE	RADIANCE	Genelux	BuildOpt
B	8.11	8.86	6.39	9.42	11.09
E	1.39	1.53	1.14	1.43	1.51
F	0.68	0.69	0.53	0.80	0.65

Case (b)					
Point	EPFL scale	SUPERLITE	RADIANCE	Genelux	BuildOpt
B	8.27	9.17	6.57	9.42	11.13
E	1.34	1.57	1.26	1.43	1.53
F	0.69	0.70	0.48	0.80	0.66

Case (c)					
Point	EPFL scale	SUPERLITE	RADIANCE	Genelux	BuildOpt
B	12.13	11.5	8.44	10.54	11.96
E	4.09	3.17	2.40	2.29	1.90
F	3.12	2.21	1.46	1.48	1.18

Case (d)					
Point	EPFL scale	SUPERLITE	RADIANCE	Genelux	BuildOpt
B	12.74	13.27	8.55	13.4	13.37
E	4.24	3.90	2.38	3.56	2.50
F	3.18	2.69	1.49	2.34	1.55

Table B.4: Daylight factors in [%] for the LESO scale model with an isotropic sky.



(a) Reflectance inside 0%, ground 0%.

(b) Reflectance inside 0%, ground 29%.

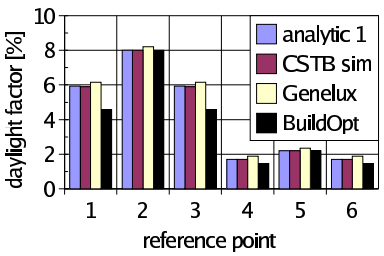
(c) Reflectance inside non-zero, ground 0%.

(d) Reflectance inside non-zero, ground 29%.

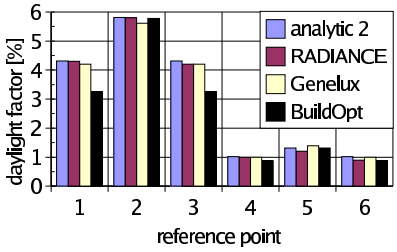
Figure B.38: Daylight factors for the LESO scale model with an isotropic sky.

B.2.3.2 CSTB/ECAD Scale Model

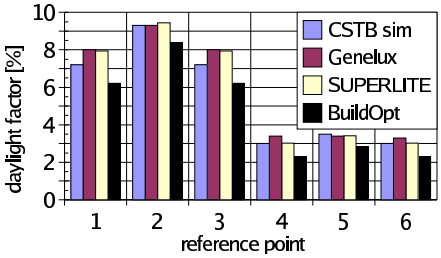
Tab. B.5 and Fig. B.39 show the daylight factors for all benchmark tests for the LESO scale model. For the cases (a) and (b) which used a room with black surfaces, BuildOpt's results are in the range of all other results except at the reference points 1 and 3, at which the daylight factor computed by BuildOpt is 1% to 2% lower in absolute value. For cases (c) and (d) that used a room with gray surfaces, BuildOpt underestimates the daylight factor by 2% in absolute value for the reference points that are close to the aperture. The accuracy is higher for reference points that are farther away from the aperture. In general, BuildOpt tends to underestimate the daylight factors because it does not account for multiple reflectances.



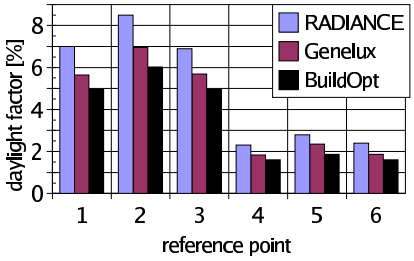
(a) Reflectance inside 0%, ground 0%, reference points on floor.



(b) Reflectance inside 0%, ground 0%, reference points 0.8 m above floor.



(c) Reflectance inside non-zero, ground 30%, reference points on floor.



(d) Reflectance inside non-zero, ground 30%, reference points 0.8 m above floor.

Figure B.39: Daylight factors for the CSTB/ECAD scale model with an isotropic sky.

Case (a)				
Point	analytic 1	CSTB sim	Genelux	BuildOpt
1	5.94	5.90	6.16	4.57
2	8.01	8.00	8.21	8.00
3	5.94	5.90	6.16	4.57
4	1.71	1.70	1.89	1.47
5	2.21	2.20	2.35	2.20
6	1.71	1.70	1.89	1.47

Case (b)				
Point	analytic 2	RADIANCE	Genelux	BuildOpt
1	4.32	4.30	4.20	3.26
2	5.81	5.80	5.62	5.79
3	4.32	4.20	4.20	3.26
4	1.02	1.00	1.00	0.87
5	1.32	1.20	1.39	1.32
6	1.02	0.90	1.00	0.87

Case (c)				
Point	CSTB sim	Genelux	SUPERLITE	BuildOpt
1	7.20	8.00	7.93	6.22
2	9.30	9.30	9.44	8.38
3	7.20	8.00	7.93	6.22
4	3.00	3.40	3.03	2.30
5	3.50	3.40	3.42	2.83
6	3.00	3.30	3.03	2.30

Case (d)			
Point	RADIANCE	Genelux	BuildOpt
1	7.00	5.64	4.96
2	8.50	6.95	6.03
3	6.90	5.69	4.96
4	2.30	1.84	1.61
5	2.80	2.35	1.87
6	2.40	1.87	1.61

Table B.5: Daylight factors in [%] for the CSTB/ECAD scale model with an isotropic sky.

B.2.4 Conclusions

The daylight factors computed by BuildOpt coincide for most test cases with the daylight factors that were obtained with other, substantially more detailed, commercial daylighting programs or by using analytical formulas.

The largest differences between BuildOpt's results and the values tabulated in the IEA reports from Laforgue (1997) and Fontoynont et al. (1999) are for rooms with black surfaces at reference points that are close to a wall. For these cases, BuildOpt underestimates the daylight factor by 2% in absolute value.

For the more realistic cases in which the room has reflecting surfaces, the daylight factors computed by BuildOpt are either within the range of the tabulated values, or within an absolute difference of 1%.

For rooms with reflecting surfaces, BuildOpt tends to underestimate the daylight factors at locations far away from the aperture because it does not take into account multiple reflections.