Available online at www.sciencedirect.com

SCIENCE DIRECT®

Energy and Buildings 37 (2005) 603–612

ENERGY and BUILDINGS

www.elsevier.com/locate/enbuild

Building design optimization using a convergent pattern search algorithm with adaptive precision simulations

Michael Wetter[a,1,*], Elijah Polak[b]

[a]United Technologies Research Center, East Hartford, CT 06108, USA
[b]Department of Electrical Engineering, University of California at Berkeley, Berkeley, CA 94720, USA

Received 20 January 2004; received in revised form 12 September 2004; accepted 17 September 2004

Abstract

We propose a simulation–precision control algorithm that can be used with a family of derivative free optimization algorithms to solve optimization problems in which the cost function is defined through the solutions of a coupled system of differential algebraic equations (DAEs). Our optimization algorithms use coarse precision approximations to the solutions of the DAE system in the early iterations and progressively increase the precision as the optimization approaches a solution. Such schemes often yield a significant reduction in computation time.

We assume that the cost function is smooth but that it can only be approximated numerically by approximating cost functions that are discontinuous in the design parameters. We show that this situation is typical for many building energy optimization problems. We present a new building energy and daylighting simulation program, which constructs approximations to the cost function that converge uniformly on bounded sets to a smooth function as precision is increased. We prove that for our simulation program, our optimization algorithms construct sequences of iterates with stationary accumulation points. We present numerical experiments in which we minimize the annual energy consumption of an office building for lighting, cooling and heating. In these examples, our precision control algorithm reduces the computation time up to a factor of four.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Optimization; Algorithm implementation; Generalized pattern search; Hooke–Jeeves

1. Introduction

Whole-building energy analysis programs, such as EnergyPlus [7], TRNSYS [14] and DOE-2 [30], use adaptive solvers such as Newton solvers or variable time step integration routines to compute an approximate numerical solution to a complex system of equations including implicit equations, ordinary differential equations and partial differential equations. In adaptive solvers, a change in input data can cause a change in the sequence of solver iterations or a change in the integration mesh, which causes the approximate numerical solution to be discontinuous in the design parameters. Consequently, if in solving an optimization problem, a smooth cost function is evaluated by such programs, it becomes replaced by a numerical approximation that is discontinuous in the design parameter. It is generally accepted in the simulation-based optimization community that if programs with adaptive solvers are used in conjunction with optimization algorithms that require the cost function to be smooth, one needs to compute high precision cost function approximations to prevent the optimization algorithm to fail at a discontinuity. In fact, examples of such failures in building design and control optimization are reported in [28,29].

However, there is usually no benefit in using high precision approximations to the cost function in the early iterations, while far from a minimum, and computation time

* Corresponding author.
E-mail addresses: WetterM@utrc.utc.com (M. Wetter), polak@eecs.berkeley.edu (E. Polak).

[1] This research was performed while the author was at the Simulation Research Group, Building Technologies Department, Environmental Energy Technologies Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA.

0378-7788/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.enbuild.2004.09.005

Nomenclature

$a \in \mathbf{A}$	a is an element of A
$\mathbf{A} \subset \mathbf{B}$	A is a subset of B
$\mathbf{A} \cap \mathbf{B}$	intersection of sets A and B
$\text{card}(\cdot)$	cardinality of a set
e_i	unit vector along the ith coordinate direction
$f(\cdot)$	cost function
$f^*(\cdot,\cdot)$	approximating cost function
n	dimension of the independent parameter
$\mathbb{N}$	$\{0, 1, 2, \ldots\}$
q	dimension of the precision parameter of the DAE solver
$\mathbb{Q}$	set of rational numbers
$\mathbb{Q}_+$	$\{q \in \mathbb{Q} \mid q > 0\}$
$\mathbb{R}$	set of real numbers
$\mathbb{R}^q_+$	$\{x \in \mathbb{R}^q \mid x^i > 0, \quad i \in \{1, \ldots, q\}\}$
$\lfloor s \rfloor$	$\max\{k \in \mathbb{N} \mid k \leq s\}$
t	time
x	independent parameter
$\mathbb{Z}$	$\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
Δ_k	mesh size parameter at kth iteration
$\triangleq$	equal by definition

may in fact become prohibitively expensive if high precision approximations to the cost function are used in all iterations. Thus, we propose a simulation–precision control algorithm that can be used in conjunction with a family of derivative-free optimization algorithms to optimize cost functions of the form $f : \mathbb{R}^n \to \mathbb{R}$ that are defined through the solutions of computationally expensive coupled systems of differential algebraic equations (DAEs). Our optimization algorithms use coarse precision solutions in the early iterations and include a test that progressively increases the precision of the approximating cost function as the optimization approaches a stationary point.

We assume that for obtaining numerical approximations to the solutions of the DAE system, adaptive solvers are used that iterate until a convergence criterion is met. In this situation, the computer code defines approximating cost functions $f^*(\varepsilon, \cdot)$, where $\varepsilon \in \mathbb{R}^q_+$ denotes the precision parameters of the DAE solver.

In [22,27], the authors present a simulation–precision control algorithm that can be used in conjunction with Generalized Pattern Search (GPS) algorithms [3] to control ε during the optimization. GPS algorithms are derivative free optimization algorithms. Examples of GPS algorithms are the Coordinate Search algorithm [20] or the Hooke–Jeeves algorithm [12].

In this paper, we present a new precision control algorithm for GPS algorithms, which yields faster convergence for our building energy optimization problems than the precision control scheme in [22,27]. Under the assumption that $f^*(\varepsilon, \cdot)$ converges to $f(\cdot)$ uniformly on bounded sets, as $\|\varepsilon\| \to 0$, with $f(\cdot)$ being a continuously differentiable function, we prove that our adaptive precision GPS algorithms construct sequences of iterates with stationary accumulation points. For a class of building energy optimization problems, we prove existence and uniqueness of a smooth solution of the DAE system, and hence existence, uniqueness and smoothness of $f(\cdot)$. However, since many existing building simulation programs are built on models that do not satisfy the standard requirements used to establish existence and uniqueness of a smooth solution of the DAE system, we developed the BuildOpt program [26]. BuildOpt is a new detailed thermal building and daylighting simulation program that is built on models that satisfy the smoothness assumptions that are required to prove existence and uniqueness of a smooth solution of the DAE system. We present BuildOpt and use it in the numerical experiments in which we minimize the annual source energy consumption of an office building. Our adaptive precision control algorithm reduces the computation time up to a factor of four compared to the standard Hooke–Jeeves algorithm.

2. Nomenclature

2.1. Conventions

(1) Vectors are always column vectors, and their elements are denoted by superscripts.
(2) Elements of a set or a sequence are denoted by subscripts.
(3) $f(\cdot)$ denotes a function where $(\cdot)$ stands for the undesignated variables. $f(x)$ denotes the value of $f(\cdot)$ for the argument x. $f: A \to B$ indicates that the domain of $f(\cdot)$ is in the space A, and that the image of $f(\cdot)$ is in the space B.
(4) We say that a function $f : \mathbb{R}^n \to \mathbb{R}$ is once (Lipschitz) continuously differentiable if $f(\cdot)$ is defined on $\mathbb{R}^n$, and if $f(\cdot)$ has a (Lipschitz) continuous derivative on $\mathbb{R}^n$.
(5) If $\mathbf{X}$ is a set, we denote by $\partial\mathbf{X}$ its boundary.
(6) The inner product in $\mathbb{R}^n$ is denoted by $\langle\cdot,\cdot\rangle$ and for $x, y \in \mathbb{R}^n$ 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 is defined by $\|x\| \triangleq \langle x, x\rangle^{1/2}$.
(7) If a subsequence $\{x_i\}_{i \in \mathbf{K}} \subset \{x_i\}_{i=0}^{\infty}$ converges to some point x, we write $x_i \to^{\mathbf{K}} x$.

3. Minimization problem

We will consider minimization problems of the form

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

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

with $-\infty \leq l^i < u^i \leq \infty$ for $i \in \{1, \ldots, n\}$. We assume that the cost function is once continuously differentiable and defined as:

$$f(x) \triangleq F(z(x,1)), \tag{2}$$

where $F : \mathbb{R}^m \to \mathbb{R}$ 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 [4] of the form:

$$\dot{z}(x,t) = h(x, z(x,t), \mu), \quad t \in [0, 1], \tag{3a}$$

$$z(x,0) = z_0(x), \tag{3b}$$

$$\gamma(x, z(x,t), \mu) = 0, \tag{3c}$$

where $h : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^m$, $z_0 : \mathbb{R}^n \to \mathbb{R}^m$ and $\gamma : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^l$. The notation $\dot{z}(x,t)$ denotes differentiation with respect to time.

Eq. (3a) describes a DAE system that is typically solved during a thermal building simulation after the spatial domain of wall, floor and ceiling constructions has 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, 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 (e.g., window glass).

To establish existence, uniqueness and differentiability of the solution $z(\cdot, 1)$ of (3), we will make the following assumptions.

Assumption 3.1. *With $\gamma : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^l$ as in (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 $z(\cdot, \cdot) \in \mathbb{R}^m$, Eq. (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 use of the Implicit Function Theorem [21], one can show that Assumption 3.1 implies that the solution of (3c), i.e., the $\mu^*(x, z)$ that satisfies $\gamma(x, z(x, t), \mu^*(x, z)) = 0$, is unique and once continuously differentiable. Therefore, to establish existence, uniqueness and differentiability of $z(\cdot, 1)$, we can reduce the DAE system (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

$$\tilde{h}(x, z(x,t)) \triangleq h(x, z(x,t), \mu^*(x,z)), \tag{4}$$

and write the DAE system (3) in the form

$$\dot{z}(x,t) = \tilde{h}(x, z(x,t)), \quad t \in [0, 1], \tag{5a}$$

$$z(x,0) = z_0(x). \tag{5b}$$

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

Assumption 3.2. *With $\tilde{h} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$ and $z_0 : \mathbb{R}^n \to \mathbb{R}^m$ as in (5), we assume that*

(1) *the initial condition $z_0(\cdot)$ is continuously differentiable; and*
(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:*

$$\|\tilde{h}(x', z') - \tilde{h}(x'', z'')\| \leq K(\|x' - x''\| + \|z' - z''\|), \tag{6a}$$

$$\|\tilde{h}_x(x', z') - \tilde{h}_x(x'', z'')\| \leq K(\|x' - x''\| + \|z' - z''\|), \tag{6b}$$

and

$$\|\tilde{h}_z(x', z') - \tilde{h}_z(x'', z'')\| \leq K(\|x' - x''\| + \|z' - z''\|). \tag{6c}$$

Now we can use the following theorem, which is a special case of Corollary 5.6.9 in Polak [21], to show that $f(\cdot) \triangleq F(z(\cdot, 1))$ is once continuously differentiable.

Theorem 3.3. *Suppose that $F : \mathbb{R}^m \to \mathbb{R}$ is once continuously differentiable on bounded sets, that Assumptions 3.1 and 3.2 are satisfied and that $f : \mathbb{R}^n \to \mathbb{R}$ is defined by $f(x) \triangleq F(z(x,1))$. Then, $f(\cdot)$ is once continuously differentiable on bounded sets.*

We assume that $z(x,t)$ cannot be evaluated exactly, but that it can be approximated by functions $z^*(\varepsilon, x, t)$, with $z^* : \mathbb{R}^q_+ \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^m$, where $\varepsilon \in \mathbb{R}^q_+$ is a vector that contains the precision parameters of the DAE solvers. For example, given a design parameter $x \in \mathbb{R}^n$, $z^*(\varepsilon, x, t)$ is the numerical approximation to the solution $z(x, t)$ of (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 the adaptive DAE solvers.

It is generally accepted in the simulation-based optimization community that, if problem (1) is solved with an optimization algorithm that requires the cost function to be smooth, one needs to compute high-precision approximate solutions $z^*(\varepsilon, x, 1)$. However, for many building design optimization problems, computing a high-precision approximate solution $z^*(\varepsilon, x, 1)$ is computationally expensive. Thus, in our optimization algorithms we use coarse precision approximate solutions $z^*(\varepsilon, x, 1)$ in the early iterations, and progressively decrease the approximation error $|f^*(\varepsilon, x) - f(x)|$ as the optimization approaches a solution. We will assume that the cost function $f(\cdot)$ and its approximating cost functions $\{f^*(\varepsilon, \cdot)\}_{\varepsilon \in \mathbb{R}^q_+}$ have the following properties:

Assumption 3.4.

(1) *There exists an error bound function* $\varphi : \mathbb{R}^q_+ \to \mathbb{R}_+$ *such that for any bounded set* $\mathbf{S} \subset \mathbb{R}^n$, *there exist 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* $\varepsilon \leq \varepsilon_\mathbf{S}$,[2]

$$|f^*(\varepsilon, x) - f(x)| \leq K_\mathbf{S}\varphi(\varepsilon). \tag{7}$$

Furthermore,

$$\lim_{\|\varepsilon\| \to 0} \varphi(\varepsilon) = 0. \tag{8}$$

(2) *The function* $f : \mathbb{R}^n \to \mathbb{R}$ *is once continuously differentiable.*

Note that we allow the functions $\{f^*(\varepsilon, \cdot)\}_{\varepsilon \in \mathbb{R}^q_+}$ to be discontinuous, and that we need to know the function $\varphi(\cdot)$, but not the constant $K_\mathbf{S}$.

Assumption 3.4 ensures that the approximation error can be reduced to an arbitrarily small value. We will also need to make 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 3.5. (**Level set**) *Given a function* $f : \mathbb{R}^n \to \mathbb{R}$ *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*:

$$\mathbf{L}_\alpha(f) \triangleq \{x \in \mathbb{R}^n | f(x) \leq \alpha\}, \tag{9}$$

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

Assumption 3.6. (**Compactness of level sets**) *Let* $\{f^*(\varepsilon, \cdot)\}_{\varepsilon \in \mathbb{R}^q_+}$ *be as in Assumption 3.4, let* $x_0 \in \mathbb{R}^n$ *be the initial iterate, and let* $\varepsilon_0 \in \mathbb{R}^q_+$ *be the initial solver precision parameter. We assume that there exists a compact set* $\mathbf{C} \subset \mathbb{R}^n$ *such that for all* $\varepsilon \in \mathbb{R}^q_+$, *with* $\varepsilon \leq \varepsilon_0$,

$$\mathbf{L}_{f^*(\varepsilon_0, x_0)}(f^*(\varepsilon, \cdot)) \subset \mathbf{C}. \tag{10}$$

4. Computation of approximate solutions of the DAE system

4.1. Existing building simulation programs

Many of today's commercially available building simulation programs, such as EnergyPlus, TRNSYS and DOE-2, are based on models that do not satisfy Assumptions 3.1 and 3.2, and the approximating cost functions $f^*(\varepsilon, \cdot)$, defined by these programs, do not satisfy Assumption 3.4. Numerical experiments [28,29] have shown that cost functions, computed by EnergyPlus, have discontinuities in the order of 2% of the cost function value, which caused various optimization algorithms to fail, sometimes far from a minimum. Furthermore, in many of today's commercially available building simulation programs, the numerical solvers are implemented in a way that makes it impossible to establish error bounds as required by Assumption 3.4. For example, to simulate a thermal zone with daylighting control and purchased heating and cooling, EnergyPlus uses at least 10 precision parameters, most of which are fixed at compile time.

Firstly, in EnergyPlus and in DOE-2, if the window area changes from one iteration to the next, then these programs may use a different window discretization to approximate the daylight illuminance of the window aperture and hence $\tilde{h}(\cdot, \cdot)$ is discontinuous. Secondly, many building simulation programs compute the convective heat flux between a wall and the room air using the function

$$q(x, t) = c\Theta(x, t)|\Theta(x, t)|^{1/3}, \tag{11}$$

with $\Theta(x, t) \triangleq T_w(x, t) - T_a(x, t)$, where $c > 0$ is a constant, $T_w(x, t) \in \mathbb{R}$ is the wall surface temperature, and $T_a(x, t) \in \mathbb{R}$ is the room air temperature. For ease of explanation, suppose that $T_w(x, t) > 0$ and $T_a(x, t) = 0$. Then, $q(x, t) = cT_w(x, t)^{4/3}$ and hence the slope of the partial derivative $\partial q(x, t)/\partial T_w(x, t) = \frac{4}{3}cT_w(x, t)^{1/3}$ goes to infinity as $T_w(x, t) \to 0$ from above. Thus, $\partial q(x, t)/\partial T_w(x, t)$ is not Lipschitz continuous, and consequently, there exists no $K \in [1, \infty)$ such that Eq. (6b) is satisfied.

Therefore, we developed BuildOpt, a new detailed building energy and daylighting simulation program. A detailed description and validation of BuildOpt can be found in [26]. BuildOpt satisfies Assumptions 3.1, 3.2 and 3.4. In BuildOpt, the error of the approximate solutions can be controlled so that $f^*(\varepsilon, \cdot) \to f(\cdot)$ uniformly on bounded sets, as $\varepsilon \to 0$, where $f(\cdot)$ is a once continuously differentiable function.

4.2. BuildOpt building simulation program

BuildOpt uses a DAE system as defined by (3) that satisfies Assumptions 3.1 and 3.2, and it uses the DAE solver DASPK [5] to compute approximate solutions of (3). The DASPK solver uses a variable time-step, variable order Backward-Differentiation Formula [4,11].

To solve the DAE system (3) using DASPK, it is written in the residual form:

$$G(t, v(x, t), \dot{v}(x, t)) = \begin{pmatrix} \dot{z}(x, t) - h(x, z(x, t), \mu^*(x, z)) \\ \gamma(x, z(x, t), \mu^*(x, z)) \end{pmatrix} = 0, \tag{12}$$

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 is the solution of (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

[2] 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 $i \in \{1, \ldots, q\}$.

for$v(x, 0)$ (see [6]).[3] At each time step $t \in [0,1]$, DASPK passes to BuildOpt a $\hat{t} > t$, a $\hat{v}(x, \hat{t})$ and a $\hat{\dot{v}}(x, \hat{t})$, where $\hat{\dot{v}}(x, \hat{t})$ is approximated by backward differences[4] and BuildOpt returns to DASPK the residual vector $G(\hat{t}, \hat{v}(x, \hat{t}), \hat{\dot{v}}(x, \hat{t})) \in \mathbb{R}^{m+l}$. This process is repeated iteratively until all convergence tests in DASPK are satisfied. See [4] for a more detailed description of DASPK. 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 approximates the iteration matrices using finite differences. The linear system of equations that arises in the Newton iterations is solved using a direct method rather than Krylov iterations.[5]

The simulation code that we developed to evaluate $G(\cdot, \cdot, \cdot)$ consists of 30,000 lines or 1.2 MB of C/C++ code. To test the feasibility of our adaptive precision optimization algorithms for solving detailed building optimization problems, we implemented models that are as detailed as the models used in commercial simulation programs.[6] For example, the diffuse solar irradiation is computed using Perez' model [19,18] and the radiation temperature of a cloudy sky is computed using Martin–Berdahl's model [17]. To compute the heat conduction in opaque materials, with possibly composite layers, we use the Galerkin method [23,8] for the spatial discretization, and integrate the spatially discretized equation with respect to time in DASPK, coupled to all other equations. The short-wave radiation through multi-pane windows is computed using a model similar to the one used in the Window 4 program [9]. The daylight illuminance is computed with a model based on view-factors that is similar to the model in the DeLight program of Vartiainen [24].

BuildOpt also differs from other building simulation programs in that it uses various smoothing methods to make the model equations, the table look-ups (used in Perez' model), and the weather data interpolations Lipschitz continuously differentiable as required by Assumption 3.2. Hence, $G(\cdot, \cdot, \cdot)$ is smooth, which allows using a DAE solver like DASPK that makes it possible to control the error of the approximating cost functions.

The thermal simulation model is validated using the ANSI/ASHRAE Standard test procedure 140–2001 [2], and the daylighting simulation is validated using benchmark tests [16,10] produced in the Task 21 of the International Energy Agency (IEA) Solar Heating & Cooling Program. The results of BuildOpt show good agreement with the results of the other validated programs.

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

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

[5] Using Krylov iterations may reduce computation time to solve the linear system of equations, but we did not implement this feature in BuildOpt.

[6] We implemented, however, only the models that we needed for our numerical experiments.

5. Optimization algorithm

We will now present our adaptive precision GPS algorithms that we developed to solve problem (1). The difference between our adaptive precision GPS algorithms and fixed precision GPS algorithms, such as the ones in [3], is that our algorithms have a test that controls the precision of the approximating cost functions. The test causes the optimization algorithms to use coarse approximations to the cost function in the early iterations and to progressively increase the precision of the approximating cost functions as the sequence of iterates approaches a stationary point. Another difference between the GPS algorithms presented here and the ones in [3] and [22] is that the algorithms presented here can be parametrized so that they only accept iterates that reduce the cost sufficiently. A sufficient decrease condition in conjunction with pattern search algorithms has also been used by others, see for example the review [15].

We will first explain the Coordinate Search algorithm with fixed precision cost function evaluations, and then explain our precision control algorithm. We selected the Coordinate Search algorithm because it is the simplest member of the family of GPS algorithms and illustrates best how the precision control can be implemented. In [22], we show a generic model GPS algorithm and the implementation of the Hooke–Jeeves algorithm. For $k \in \mathbb{N}$ let $x_k \in \mathbb{R}^n$ denote the current iterate, let $\Delta_k \in \mathbb{Q}_+$ be a positive number, called the mesh size parameter, and let $\mathcal{L}_k \triangleq \{x_k \pm \Delta_k e_i | i \in \{1, \ldots, n\}\} \cap \mathbf{X}$. If there exists a point $x' \in \mathcal{L}_k$ that satisfies $f^*(\varepsilon, x') - f^*(\varepsilon, x_k) < 0$, then our fixed precision Coordinate Search algorithm sets $x_{k+1} = x'$, $\Delta_{k+1} = \Delta_k$ and replaces k by $k + 1$. Otherwise, it sets $x_{k+1} = x_k$, $\Delta_{k+1} = \Delta_k/2$ and replaces k by $k + 1$.

We will now explain our precision control scheme. Let $\varphi : \mathbb{R}^q_+ \to \mathbb{R}_+$ be as in Assumption 3.4 and let $\zeta \geq 0$ be a constant. Let $\rho : \mathbb{N} \to \mathbb{R}^q_+$, with $\varphi \circ \rho : \mathbb{N} \to \mathbb{R}_+$ strictly monotone decreasing, be a function that is used to assign the precision parameter of the DAE solver, and let $\alpha \in (0, 1)$ be a constant. At the beginning of the optimization, we initialize a counter $N = 1$ and we set $\varepsilon = \rho(N)$. The iterations are as follows. If there exists an $x' \in \mathcal{L}_k$ that satisfies $f^*(\varepsilon, x') - f^*(\varepsilon, x_k) < -\zeta\varphi(\varepsilon)$, we set, as in the fixed precision algorithm, $x_{k+1} = x'$, $\Delta_{k+1} = \Delta_k$ and replace k by $k + 1$. Otherwise, we replace N by $N + 1$ and use tighter precision $\varepsilon = \rho(N)$. If $\varphi(\varepsilon)^\alpha/\Delta_k < \Delta_k$ (for the new ε), we decrease the mesh size parameter by setting $\Delta_{k+1} = \Delta_k/2^m$, where $m \triangleq \arg\min\{m \in \mathbb{N} | \varphi(\varepsilon)^\alpha \geq (\Delta_k/2^m)^2\}$. If $\varphi(\varepsilon)^\alpha/\Delta_k \geq \Delta_k$ (for the new ε), we set $\Delta_{k+1} = \Delta_k$. Thus, we decrease the mesh size parameter only after we sufficiently decreased the error of the approximating cost function, and $\alpha \in (0, 1)$ is used to control how fast the mesh size parameter is decreased. If N is equal to a user-specified limit $N^* \in \mathbb{N}$ the search stops. Hence, for the last iterations, the precision parameter of the DAE solver is $\varepsilon^* = \rho(N^*)$.

The above precision control scheme is different from the one that the authors present in [22,27]. In [22,27], only a simple decrease in cost is required for an iterate to be accepted, and the precision control scheme decreases Δ_k and ε simultaneously whenever $f^*(\varepsilon, x') \geq f^*(\varepsilon, x_k)$ for all $x' \in \mathcal{L}_k$. However, the algorithm in [22,27], when applied to the building design optimization problems that we will present in this paper, searched with a constant ε for many iterations without significantly reducing the cost, and Δ_k was decreased too fast. Thus, to increase the precision faster, we added a sufficient decrease condition, and to prevent Δ_k from becoming too small in early iterations, we increase first the precision of the approximating cost functions, and decrease Δ_k only after the precision of the approximating cost functions has been sufficiently increased. This yields the following algorithm.

Algorithm 5.1.

```
Data    Parameters $\alpha \in (0, 1)$ and $\zeta \geq 0$;
        Initial iterate $x_0 \in \mathbf{X}$;
        Initial mesh size parameter $\Delta_0 \in \mathbb{Q}_+$.
Maps    Function $\varphi : \mathbb{R}^q_+ \to \mathbb{R}_+$ as in Assumption 3.4;
        Function $\rho : \mathbb{N} \to \mathbb{R}^q_+$ (to assign ε),
        such that the composition $\varphi \circ \rho : \mathbb{N} \to \mathbb{R}_+$
        is strictly monotone decreasing.
Step 0  Set $k = 0$, $N = 1$, $\varepsilon = \rho(N)$.
Step 1  Search
        For $i \in \{1, \ldots, n\}$,
          Set $x' = x_k + \Delta_k e_i$.
          If $f^*(\varepsilon, x') - f^*(\varepsilon, x_k) < -\zeta\varphi(\varepsilon)$,
            go to Step 3.
          Set $x' = x_k - \Delta_k e_i$.
          If $f^*(\varepsilon, x') - f^*(\varepsilon, x_k) < -\zeta\varphi(\varepsilon)$,
            go to Step 3.
        end for.
Step 2  No sufficient cost reduction
        Replace $N$ by $N + 1$ and set $\varepsilon = \rho(N)$.
        If $\varphi(\varepsilon)^\alpha/\Delta_k < \Delta_k$,
          set $\Delta_{k+1} = \Delta_k/2^m$, with $m \triangleq \arg\min\{m \in \mathbb{N} | \varphi(\varepsilon)^\alpha \geq (\Delta_k/2^m)^2\}$.
        else
          set $\Delta_{k+1} = \Delta_k$.
        Set $x_{k+1} = x_k$, and go to Step 4.
Step 3  Cost sufficiently reduced
        Set $x_{k+1} = x'$, $\Delta_{k+1} = \Delta_k$, do not change $N$,
        and go to Step 4.
Step 4  Replace $k$ by $k + 1$, and go to Step 1.
```

6. Convergence results

6.1. Unconstrained minimization

We will now establish the convergence properties of Algorithm 5.1 on unconstrained minimization problems, i.e., for $\mathbf{X} = \mathbb{R}^n$. Box-constrained problems are discussed in Section 6.2.

First, we note that all iterates constructed by Algorithm 5.1 belong to the grid

$$\mathbb{M}_k \triangleq \{x_0 + \Delta_k m | m \in \mathbb{Z}^n\}. \tag{13}$$

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

Proposition 6.1. *Any bounded subset of a mesh $\mathbb{M}_k$contains only a finite number of mesh points.*

Proposition 6.2. *Suppose that Assumption 3.6 is satisfied and let $\{\Delta_k\}_{k=0}^\infty \subset \mathbb{Q}_+$be the sequence of mesh size parameters constructed by Algorithm 5.1. Then,* $\liminf_{k\to\infty} \Delta_k = 0$.

Proof. Suppose $\liminf_{k\to\infty} \Delta_k \neq 0$. Then, Step 2 in Algorithm 5.1 can only be executed for a finite number of iterations because in Step 2, N is replaced by $N + 1$, 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 $\mathbb{M}_{k^*} \triangleq \{x_0 + \Delta_{k^*} m | m \in \mathbb{Z}^n\}$.

By Assumption 3.6, there exists a compact set $\mathbf{C}$, such that $\mathbf{L}_{f^*(\varepsilon_0,x_0)}(f^*(\varepsilon,\cdot)) \subset \mathbf{C}$, for all $\varepsilon \in \mathbb{R}^q_+$, with $\varepsilon \leq \varepsilon_0 = \rho(1)$. Hence, it follows from Proposition 6.1 that $\mathbb{M}_{k^*} \cap \mathbf{L}_{f^*(\varepsilon_0,x_0)}(f^*(\varepsilon,\cdot))$ 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 5.1. □

Having shown that $\liminf_{k\to\infty} \Delta_k = 0$, we can introduce the notion of a refining subsequence as used by Audet and Dennis [3].

Definition 6.3 (Refining subsequence). *Consider a sequence $\{x_k\}_{k=0}^\infty$ constructed by Algorithm 5.1. 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 $k \notin \mathbf{K}$.*

We now state that GPS algorithms with adaptive precision function evaluations construct sequences of iterates with stationary accumulation points.

Theorem 6.4 (Convergence to a stationary point). *Suppose that Assumptions 3.4 and 3.6 are satisfied. Let $x^* \in \mathbb{R}^n$ be an accumulation point of the refining subsequence $\{x_k\}_{k\in\mathbf{K}}$ constructed by Algorithm 5.1. Then,*

$$\nabla f(x^*) = 0. \tag{14}$$

Proof. 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 3.6, there exists a compact set $\mathbf{C}$ such that $\mathbf{L}_{f^*(\varepsilon_0,x_0)}(f^*(\varepsilon,\cdot)) \subset \mathbf{C}$, for all $\varepsilon \in \mathbb{R}^q_+$, with $\varepsilon \le \varepsilon_0 = \rho(1)$. Therefore, by Assumption 3.4, there exist 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 $\varepsilon \le \varepsilon_{\mathbf{L}}$, we have $|f^*(\varepsilon, x) - f(x)| \le K_{\mathbf{L}}\varphi(\varepsilon)$.

Next, pick an arbitrary $h \in \{-e_1, +e_1, \ldots, -e_n, +e_n\}$. Because $f(\cdot)$ is continuously differentiable, it follows from the Mean Value Theorem that for all $k \in \mathbf{K}$, there exists a corresponding $s'_k \in (0, 1)$ such that

$$f(x_k + \Delta_k h) - f(x_k) = \langle \nabla f(x_k + s'_k \Delta_k h), \Delta_k h\rangle = \Delta_k df(x_k + s'_k \Delta_k h; h), \tag{15}$$

where $df(\cdot;\cdot)$ denotes the directional derivative [21]. Similarly, there exists an $s''_k \in (0, 1)$ such that

$$f(x_k - \Delta_k h) - f(x_k) = -\Delta_k df(x_k - s''_k \Delta_k h; h). \tag{16}$$

Because $f^*(\varepsilon, x_k + \Delta_k h) - f^*(\varepsilon, x_k) \ge -\zeta\varphi(\varepsilon)$ for all $k \in \mathbf{K}$, it follows from (15) that for all $k \in \mathbf{K}$,

$$\begin{aligned} df(x_k + s'_k \Delta_k h; h) &= \frac{f(x_k + \Delta_k h) - f(x_k)}{\Delta_k} \\ &\ge \frac{f^*(\varepsilon, x_k + \Delta_k h) - f^*(\varepsilon, x_k)}{\Delta_k} \\ &\quad - 2K_{\mathbf{L}}\frac{\varphi(\varepsilon)}{\Delta_k} \ge -\zeta\frac{\varphi(\varepsilon)}{\Delta_k} - 2K_{\mathbf{L}}\frac{\varphi(\varepsilon)}{\Delta_k} \\ &= -(\zeta + 2K_{\mathbf{L}})\frac{\varphi(\varepsilon)}{\Delta_k}. \end{aligned} \tag{17}$$

It follows similarly from (16) that for all $k \in \mathbf{K}$,

$$\begin{aligned} -df(x_k - s''_k \Delta_k h; h) &= \frac{f(x_k - \Delta_k h) - f(x_k)}{\Delta_k} \\ &\ge \frac{f^*(\varepsilon, x_k - \Delta_k h) - f^*(\varepsilon, x_k)}{\Delta_k} \\ &\quad - 2K_{\mathbf{L}}\frac{\varphi(\varepsilon)}{\Delta_k} \ge -\zeta\frac{\varphi(\varepsilon)}{\Delta_k} - 2K_{\mathbf{L}}\frac{\varphi(\varepsilon)}{\Delta_k} \\ &= -(\zeta + 2K_{\mathbf{L}})\frac{\varphi(\varepsilon)}{\Delta_k}. \end{aligned} \tag{18}$$

Since by Proposition 6.2, $\Delta_k \to^{\mathbf{K}} 0$, it follows from the constructions in Algorithm 5.1 that $\varphi(\varepsilon)/\Delta_k \to^{\mathbf{K}} 0$. Hence, it follows from (17) that $df(x_k + s'_k \Delta_k h; h) \ge 0$ and from (18) that $df(x_k - s''_k \Delta_k h; h) \le 0$, for all $k \in \mathbf{K}$. Because $(x_k + s'_k \Delta_k h) \to^{\mathbf{K}} x^*$ and $(x_k - s''_k \Delta_k h) \to^{\mathbf{K}} x^*$, it follows from the continuity of $\nabla f(\cdot)$ that $df(x^*; h) = 0$. Since $h \in \{-e_1, +e_1, \ldots, -e_n, +e_n\}$ is arbitrary, we have $\nabla f(x^*) = 0$. □

6.2. Box-constrained minimization

We will now extend the convergence results to box-constrained problems, i.e., for $\mathbf{X}$ as defined in (1b). The case with linear constraints is discussed in [3,22].

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

Definition 6.5 (**Tangent and normal cone**).

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

$$\mathbf{T}_{\mathbf{X}}(x^*) \triangleq \{\mu(x - x^*) | \mu \ge 0, \quad x \in \mathbf{X}\}. \tag{19a}$$

(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*

$$\mathbf{N}_{\mathbf{X}}(x^*) \triangleq \{\nu \in \mathbb{R}^n | \forall t \in \mathbf{T}_{\mathbf{X}}(x^*), \quad \langle \nu, t\rangle \le 0\}. \tag{19b}$$

We have the following theorem.

Theorem 6.6 (**Convergence to a feasible stationary point**).
Suppose that Assumptions 3.4 and 3.6 are satisfied. Let $x^* \in \mathbf{X}$ be an accumulation point of the refining subsequence $\{x_k\}_{k\in\mathbf{K}}$ constructed by Algorithm 5.1 in solving problem (1). Then,

$$\langle \nabla f(x^*), t\rangle \ge 0, \quad \forall t \in \mathbf{T}_{\mathbf{X}}(x^*), \tag{20a}$$

and

$$-\nabla f(x^*) \in \mathbf{N}_{\mathbf{X}}(x^*). \tag{20b}$$

Proof. If x^* is in the interior of $\mathbf{X}$, then the result reduces to Theorem 6.4.

Let $x^* \in \partial\mathbf{X}$. There exists a set of search directions $\mathcal{H}(x^*) \subset \{-e_1, +e_1, \ldots, -e_n, +e_n\}$ such that $\mathbf{T}_{\mathbf{X}}(x^*) = \{\sum_{i=1}^{\operatorname{card}\mathcal{H}(x)} \alpha^i h_i | h_i \in \mathcal{H}(x^*), \alpha^i \ge 0, i \in \{1, \ldots, \operatorname{card}\ \mathcal{H}(x^*)\}\}$. Because $x_k \to^{\mathbf{K}} x^*$, it follows from (17) that there exists an infinite subset $\mathbf{K}' \subset \mathbf{K}$ for which $df(x_k + \Delta_k s'_k h; h) \ge 0$, with $s'_k \in (0, 1)$, for all $k \in \mathbf{K}'$ and for all $h \in \mathcal{H}(x^*)$. Thus, it follows from the continuity of $\nabla f(\cdot)$ that $\langle \nabla f(x^*), t\rangle \ge 0$, for all $t \in \mathbf{T}_{\mathbf{X}}(x^*)$. It follows directly that $\langle -\nabla f(x^*), t\rangle \le 0$, for all $t \in \mathbf{T}_{\mathbf{X}}(x^*)$, which shows that $-\nabla f(x^*) \in \mathbf{N}_{\mathbf{X}}(x^*)$. □

7. Numerical experiments

We will now describe the performance of our precision control algorithm used in conjunction with the Hooke–Jeeves optimization algorithm. The optimizations were done using the GenOpt$^{(R)}$ 2.0.0 optimization program [25].

We minimized the annual source energy consumption of the office rooms shown in Fig. 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.

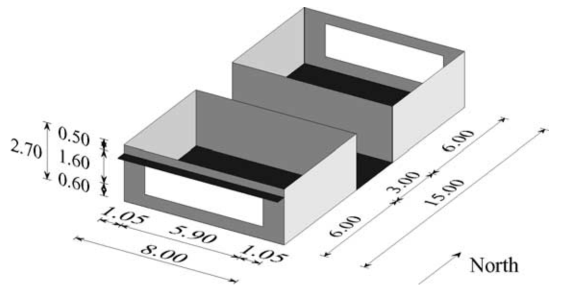

Fig. 1. Thermal zones used for computing the buildings annual source energy consumption.

The south window has a shading overhang. The north and south zones have daylighting controls with an illuminance setpoint of 500 lux 3 m from the window. We used TMY2 weather data for Houston Intercontinental, TX.

The annual source energy consumption is

$$f(x) \triangleq \frac{Q_h(x)}{\eta_h} + \frac{Q_c(x)}{\eta_c} + 3E_l(x), \tag{21}$$

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 plant efficiencies that relate the zone load to the primary energy consumption for heating and cooling generation, including electricity consumption for fans and pumps [13]. The electricity consumption is multiplied by three to convert site electricity to source fuel energy consumption.

There are five independent variables, normalized so that $0 \leq x^i \leq 1$ for all $i \in \{1, \ldots, 5\}$. The components x^1 and x^2 linearly scale the width of the north and south facing window, respectively, from 4 m to 7.8 m. The component x^3 linearly scales the width of the window overhang from 0.1 m to 1.0 m. The components x^4 and x^5 linearly scale the shading control setpoints. For the north window, the setpoint is varied from 100 W/m^2 to 200 W/m^2 and for the south window it is varied from 100 W/m^2 to 600 W/m^2. For the initial iterate, we set $x_0^i = 0.5$ for all $i \in \{1, \ldots, 5\}$.

We solved the optimization problem with fixed precision and with adaptive precision cost function evaluations. In the optimization with fixed precision cost function evaluations, we set $\varepsilon = 10^{-5}$ and we allowed the mesh size parameter Δ_k to be decreased four times before the optimization stops.

For the optimization with adaptive precision cost function evaluations, we defined $\rho : \mathbb{N} \to \mathbb{R}_+$ as $\rho(N) = 10^{-N}$ and increased the precision four times. Thus, $\varepsilon = \rho(1) = 10^{-1}$ for the first iterations, and $\varepsilon = \rho(5) = 10^{-5}$ for the last iterations. Present day DAE solvers, including 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 $z^*(\varepsilon, \cdot, 1)$ 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 (7).)

Table 1
Normalized computation time required to solve the optimization problem with Algorithm 5.1

	$\zeta = 0$	$\zeta = 10^{-8}$	$\zeta = 10^{-6}$	$\zeta = 10^{-4}$	$\zeta = 10^{-2}$	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

The last column shows for each α the smallest Δ_k used in the search.

In Table 1, we show the values that we selected for the algorithm parameters $\alpha \in (0, 1)$ and $\zeta \geq 0$, the corresponding computation time and in the last column the smallest mesh size parameter Δ_{k^*}. A computation time of 1 corresponds to 5.5 days of computing on a 2.2 GHz AMD processor running Linux with the 2.4.18–3 kernel.

Note that in Algorithm 5.1, 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 up to a factor of four. 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 source energy consumption by 4.6% or 9.4 kWh/(m^{2}a). The 4.6% reduction is small, but not representative since considerably higher reductions have been reported in other building design optimizations, see for example [1,29]. How big the reduction is depends on how good the initial design is and how sensitive the energy consumption is with respect to changes in the design parameter, which differs from one situation to another. Furthermore, the purpose of this paper is to present a new technique for building design optimization, rather than to assess the typical savings that can be obtained by optimization.

We will now describe how the optimizations 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 kth iterate $x_k \in \mathbb{R}^n$ to the minimizer $x^* \triangleq \arg\min_{x \in \mathbf{X}} f(x)$ be defined as $d(x_k) \triangleq \|x_k - x^*\| / \|x_0 - x^*\|$, where $x_0 \in \mathbb{R}^n$ is the initial iterate. Fig. 2 shows the cost function value and the distance to the minimizer as a function of the computation time. The horizontal axis is in logarithmic scale for better display of the early iterations. 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-hand side graph, we can see that even for such coarse a

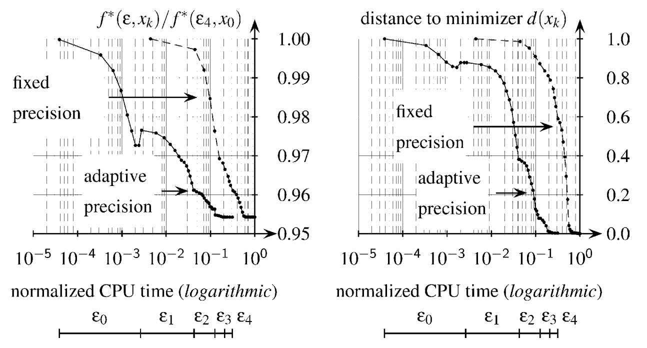

Fig. 2. Normalized cost function value (left-hand side graph) and distance to the minimizer (right-hand side graph) as a function of the normalized CPU time in logarithmic scale. Below the graphs we show the intervals for which the precision ε has been kept constant. For the adaptive precision optimization, we used $\zeta = 10^{-4}$ and $\alpha = 1/6$. For better display of the early iterations, the time scale is in logarithmic scale.

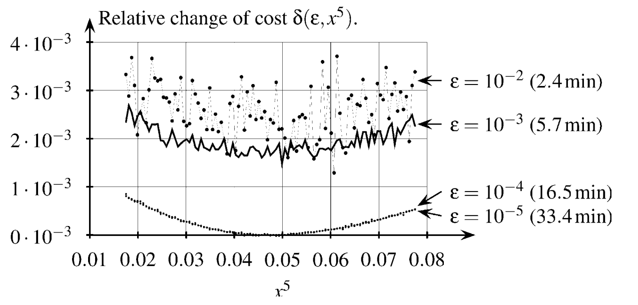

Fig. 3. Relative change of cost along the coordinate direction e_5 for different precision parameters ε. For better visibility of the data series, the support points are connected by lines. In brackets, we show the computation time required for one cost function evaluation with the corresponding precision parameter ε. The component $x^5 \in [0, 1]$ scales the setpoint of the shading device for the south facing window.

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.

In Fig. 3 we show how the functions $\{f^*(\varepsilon, \cdot)\}_{\varepsilon \in \mathbb{R}^q_+}$ converge to a smooth function. The figure shows the relative change of the cost function value $\delta(\varepsilon, x^5)$, defined as

$$\delta(\varepsilon, x^5) \triangleq \frac{f^*(\varepsilon, x^* + (x^5 - (x^*)^5)e_5) - f^*(10^{-5}, x^*)}{f^*(10^{-5}, x^*)}, \qquad (22)$$

where $x^* \triangleq \arg\min_{x \in \mathbf{X}} f(x)$. For the subspace spanned by the coordinate vector e_5, the figure shows how the discontinuities in $f^*(\varepsilon, \cdot)$ vanish as $||\varepsilon|| \to 0$, that $f^*(\varepsilon, \cdot)$ converges to a smooth function, and how much the computation time for one cost function evaluation increases as ε is decreased. The difference between the functions $f^*(10^{-4}, \cdot)$ and $f^*(10^{-5}, \cdot)$ is too small to be visible in Fig. 3.

8. Conclusion

Building energy and daylighting simulation programs construct discontinuous approximations to a usually continuously differentiable cost function. This can cause optimization methods to fail far from an optimal solution. In such cases, the economic potential that optimization offers is not realized. To eliminate this problem, one needs to use high precision approximations to the cost function, which may cause the computation time to be prohibitively long if used for all iterations.

We have shown that detailed thermal building and daylighting simulation programs can be written so that they compute for a large class of optimization problems approximating cost functions that converge to a once continuously differentiable function as precision is increased.

We have presented a precision control algorithm that uses low-cost, coarse precision approximations to the cost function when far from a solution, with the precision progressively increased as a solution is approached. We have proven that our optimization algorithm constructs a sequence of iterates with stationary accumulation point even though the cost function is approximated by a family of discontinuous functions.

In the presented numerical experiments, our precision control scheme reduces the computation time up to a factor of four compared to the standard Hooke–Jeeves algorithm.

Acknowledgments

This research was supported 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, and by the National Science Foundation under Grant No. ECS-9900985.

References

[1] Mohammad S., Al-Homoud Optimum thermal design of office buildings, International Journal of Energy Research 21 (1997) 941–957.
[2] ASHRAE. ANSI/ASHRAE Standard 140-2001, Standard method of test for the evaluation of building energy analysis computer programs, 2001.
[3] C. Audet, J.E. Dennis Jr., Analysis of generalized pattern searches, SIAM Journal on Optimization 13 (3) (2003) 889–903.
[4] K.E. Brenan, S.L. Campbell, L.R. Petzold, Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations, North-Holland, 1989.
[5] Peter N. Brown, Alan C. Hindmarsh, Linda R. Petzold, Using Krylov methods in the solution of large-scale differential-algebraic systems, SIAM Journal on Scientific Computing 15 (1994) 1467–1488.
[6] Peter N. Brown, Alan C. Hindmarsh, Linda R. Petzold, Consistent initial condition calculation for differential-algebraic systems, SIAM Journal on Scientific Computing 19 (5) (1998) 1495–1512, September.
[7] Drury B. Crawley, Linda K. Lawrie, Frederick C. Winkelmann, Walter F. Buhl, Y. Joe Huang, Curtis O. Pedersen, Richard K. Strand, Richard J. Liesen, Daniel E. Fisher, Michael J. Witte, Jason Glazer, EnergyPlus: creating a new-generation building energy simulation program, Energy and Buildings 33 (4) (2001) 443–457.
[8] Lawrence C. Evans, Partial Differential Equations, American Mathematical Society, 1998.

[9] E.U. Finlayson, D.K. Arasteh, C. Huizenga, M.D. Rubin, M.S. Reilly, WINDOW 4.0: Documentation of Calculation Procedures, Technical Report LBL-33943, Lawrence Berkeley National Laboratory, Berkeley, CA, USA, July 1993.

[10] M. Fontoynont, P. Laforgue, R. Mitanchey, M. Aizlewood, J. Butt, W. Carroll, R. Hitchcock, H. Erhorn, J. De Boer, M. Dirksmöller, L. Michel, B. Paule, J.L. Scartezzini, M. Bodart, G. Roy, 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, November 1999.

[11] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II. Springer Series in Computational Mathematics, second edition, Springer–Verlag, Berlin, 1996.

[12] R. Hooke, T.A. Jeeves, 'Direct search' solution of numerical and statistical problems, Journal of the Association for Computing Machinery 8 (2) (1961) 212–229.

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

[14] S.A. Klein, J.A. Duffie, W.A. Beckman, TRNSYS–a transient simulation program, ASHRAE Transactions 82 (1) (1976) 623–633.

[15] Tamara G. Kolda, Robert Michael Lewis, Virginia Torczon, Optimization by direct search: new perspectives on some classical and modern methods, SIAM Review 45 (3) (2003) 385–482.

[16] P. Laforgue, IEA SHC Task 21/ECBCS Annex 29, Daylight in buildings, Subtask C1: draft report of Genelux simulations and other software results, Technical Report T21/C1/97-10, International Energy Agency, October 1997.

[17] Marlo Martin, Paul Berdahl, Characteristics of infrared sky radiation in the United States, Solar Energy 33 (1984) 321–336.

[18] Richard Perez, Pierre Ineichen, Robert Seals, Joseph Michalsky, Ronald Stewart, Modeling daylight availability and irradiance components from direct and global irradiance, Solar Energy 44 (5) (1990) 271–289.

[19] Richard Perez, Robert Seals, Pierre Ineichen, Ronald Stewart, David Menicucci, A new simplified version of the Perez diffuse irradiance model for tilted surfaces, Solar Energy 39 (3) (1987) 221–231.

[20] Elijah Polak, Computational Methods in Optimization; a Unified Approach, Volume 77 of Mathematics in Science and Engineering, Academic Press, New York, 1971.

[21] Elijah Polak, Optimization, Algorithms and Consistent Approximations Volume 124 of Applied Mathematical Sciences, Springer–Verlag, 1997.

[22] Elijah Polak, Michael Wetter, Generalized Pattern Search Algorithms with Adaptive Precision Function Evaluations, Technical Report LBNL-52629, Lawrence Berkeley National Laboratory, Berkeley, CA, 2003.

[23] Gilbert Strang, George J. Fix, An Analysis of the Finite Element Method, Prentice-Hall, Inc., 1973.

[24] E. Vartiainen, Daylight modelling with the simulation tool DeLight, Technical Report TKK-F-A799, Helsinki University of Technology, Finland, Department of Engineering Physics and Mathematics, June 2000.

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

[26] M. Wetter, Simulation-Based Building Energy Optimization, Ph.D. thesis, University of California at Berkeley, 2004.

[27] M. Wetter, E. Polak, A convergent optimization method using pattern search algorithms with adaptive precision simulation, in: G. Augenbroe, J. Hensen (Ed.), Proceedings of the 8-th IBPSA Conference, vol. III, Eindhoven, NL, August 2003, pp. 1393–1400.

[28] M. Wetter, J. Wright, Comparison of a generalized pattern search and a genetic algorithm optimization method, in: G. Augenbroe, J. Hensen (Eds.), Proceedings of the 8th IBPSA Conference, vol. III, Eindhoven, NL, August 2003, pp. 1401–1408.

[29] Michael Wetter, Jonathan Wright, A comparison of deterministic and probabilistic optimization algorithms for nonsmooth simulation-based optimization, Building and Environment 39 (8) (2004) 989–999, August.

[30] F.C. Winkelmann, B.E. Birsdall, W.F. Buhl, K.L. Ellington, A.E. Erdem, J.J. Hirsch, S. Gates, DOE-2 supplement, version 2. 1E, Technical Report LBL-34947, Lawrence Berkeley National Laboratory, Berkeley, CA, USA, November 1993.