Black-box optimization of lighting simulation in architectural design*

Alberto Costa†, Giacomo Nannicini, Thomas Schroepfer, and Thomas Wortmann

Abstract This paper deals with an application of optimization in architectural design. Formally, we consider the problem of optimizing a function that can only be evaluated through an expensive oracle. We assume that the analytical expression of the function is unknown and first-order information is not available. This situation frequently occurs when each function evaluation relies on the output of a complex and time-consuming simulation. In the literature, this is called a *black-box* optimization problem with costly evaluation. This paper presents a black-box problem from architectural design: we aim to find the values of the design variables that yield optimal lighting conditions inside a building. The building façade is described as a parametric model whose parameters are the design variables. We tackle this problem by adapting the Radial Basis Function (RBF) method originally proposed by Gutmann (2001). Experiments indicate that our open-source implementation is competitive with commercial software for black-box optimization, and that it can be a valuable decision-support tool for complex problems requiring time-consuming simulations. The usefulness of this approach goes beyond the specific application in architectural design.

1 Introduction

We consider the problem of optimizing an unknown function given as an oracle. We assume that the oracle is expensive to evaluate, so that estimating partial derivatives

A. Costa, G. Nannicini, T. Wortmann, T. Schroepfer
Singapore University of Technology and Design, 138682 Singapore.
e-mail: {costa,nannicini,thomas.schroepfer}@sutd.edu.sg,thomas_wortmann@mymail.sutd.edu.sg

* Supported by IDC grant IDG21300102.

† Corresponding author.

by finite differences is impractical. In the literature, such a problem is known as a *black-box* optimization problem with costly evaluation.

Black-box optimization finds many applications. Our main motivation stems from architectural design. In recent years, simulation has been increasingly employed as a decision-support tool in this field. As a case study, we optimize a building façade in terms of daylighting and glare to achieve a more sustainable design. The façade is described as a parametric model whose parameters are the decision variables. Daylight simulation that is based on site-specific climatic data can, over a prescribed period, predict brightness inside the building as well as glare. This quantitative information serves as a performance measure, i.e., an objective function. We assume that the desired characteristics can be combined into a single objective function that may include multiple terms, for instance penalties for violating constraints based on the simulated visual comfort of building's occupants. Our goal is to optimize this objective function, and thus to find optimal values for the parametric model of the design. However, each run of the simulation takes considerable time, depending on the desired accuracy. Thus, we want to find values of the decision variables (design parameters) that achieve a close to optimal value for the objective function, while performing only a small number of function evaluations to limit computing times.

Current architectural design practice often relies on experience and best practices rather than on optimization techniques. Better-informed decisions in the design stage of a building can lead to e.g. better environmental performance. Reliable solutions for these types of design problems are crucial for the development of sustainable architecture and highly liveable future cities. Even if optimum efficiency may not always be attainable in practice, optimization techniques provide invaluable information to the architect and thus create new design possibilities. One reason for today's absence of optimization techniques in the design process is the time-intensity of simulations. This constraint is exacerbated by the type of optimization algorithms that are usually proposed for architectural design, that is, genetic and swarm-based algorithms, e.g., [3, 17, 18]. Such algorithms converge on a solution only after performing many simulation runs. The RBF method described in this paper aims to provide much faster convergence, while offering flexibility for designers in defining the performance measure, i.e., the objective function.

A common approach to black-box problems is to use several evaluations of the oracle to build a model of the objective function. This model is also called *response surface* or *surrogate* model. Examples of this approach are the Radial Basis Function (RBF) method of [6] (see also [19]) and the kriging-based Efficient Global Optimization method (EGO) of [11]. Despite the appealing theoretical properties of EGO, empirical evidence suggests that the RBF method is more effective on engineering problems [9]. The RBF method has many potential applications in engineering design beyond the case study presented here, in particular for optimizing the performance of complex physical devices such as engines, see e.g., [1, 7].

The rest of this paper is organized as follows. Section 2 presents a case study of optimal decision-making for complex problems in architectural design. In particular, we introduce a method to provide mathematical models to assess daylighting quality based on the designers preferences. Section 3 provides an overview of the RBF method for black-box optimization, discusses possible extensions, and describes our open-source implementation. A brief computational evaluation shows that our implementation is competitive with commercial applications from engineering design, and orders of magnitude more efficient than genetic algorithms, which are commonly used in architectural design.

2 Application to architectural design

The conflicting objectives in architectural design and complexity of the simulations mandate a black-box approach to architectural optimization. Accordingly, the literature on optimization in the architectural design is dominated by discussions of genetic algorithms, e.g., [13, 15]. Other approaches to optimization in the architectural field, such as particle swarm and ant colony optimization, usually are swarm-based, e.g., [12]. Genetic algorithms are also the most likely technique to be employed by architecture practitioners and students. Grasshopper, a free plug-in for the popular 3D-modelling application Rhinoceros™, includes an easy-to-use evolutionary solver named Galapagos. However, in practice, evolutionary and swarm-based approaches require the evaluation of many design candidates via simulation, which likely is one of the reasons why their application in design practice has remained limited. The RBF method presented in this paper conserves the advantages of a black-box approach while greatly reducing the required number of design candidate evaluations, see Section 3.4.

Several aspects make the RBF method especially promising for applications in architectural design. First, being a black-box method, it requires no knowledge about the internal structure of the optimization problem. Although typically in the literature only specific aspects of a building design are optimized, even these smaller problems are characterised by multiple, competing objectives. For example, in optimizing a load-bearing structure, the designer often aims at minimizing both structural displacement and weight, see [12, 13]. Similarly, in façade design, opening sizes have to balance daylight and shading requirements to avoid heat gains and glare. The case study presented in this paper is located in a tropical climate, and aims at maximizing daylight while minimizing glare. Combining these objectives with the complex physics being simulated, we obtain an objective function such that the relationship between inputs and outputs is not analytically available.

The RBF method has two other important benefits over evolutionary or swarm algorithms that are especially important for architectural design. First, the surrogate model created by the RBF method approximates the design space implied by the parametric model and the performance criteria. The surrogate model thus allows the approximate evaluation of different design scenarios around the optimum without additional simulations. Design theorist Roy Woodbury conceives of architectural design as design space exploration and highlights the need for computational tools that support this exploration [23]. From the perspective of design as exploration, the

RBF method is a tool to approximately but quickly evaluate the sensitivity of the performance measure with respect to the design parameters. A discussion on how to compute a confidence level of this sensitivity analysis is given in Section 3.3.1. Second, one can use the RBF method with different objective functions without repeating time-intensive simulations. This is because in many situations, such as daylight simulation, one can save the (complex) output data of a simulation run to allow the evaluation of a different objective function relying on the already available output data. In this way, we quickly obtain a model of the modified performance criterion, and can hot-start the optimization method to converge much faster than restarting from scratch. These two aspects at least partially address an important criticism of optimization methods in the architectural design field: optimization has to rely on pre-defined performance criteria and is thus difficult to integrate into complex design processes with shifting performance criteria. This difficulty was succinctly formulated by design theorist Horst Rittel in his famous chapter “Planning Problems are wicked Problems” [20]:

> The methods of Operations Research [. . .] become operational, however, only after the most important decisions have already been made, i.e., after the problem has already been tamed. Take an optimization model. Here the inputs needed include the definition of the solution space, the system of constraints, and the performance measure as a function of the planning and contextual variables. But setting up and constraining the solution space and constructing the measure of performance is the wicked part of the problem. Very likely it is more essential than the remaining steps of searching for a solution which is optimal relative to the measure of performance and the constraint system.

The RBF method attempts to alleviate the “wickedness” of architectural design problems by allowing the approximate exploration of the design space, and, perhaps more importantly, the changing of performance criteria with significantly reduced effort.

As a case study for the integration of the RBF method into architectural design processes, the method is employed in the design of a mixed-use high-density church building in Singapore. The church is designed with a performative façade that modulates daylighting conditions on the interior due to the differentiated inclinations of its disk-shaped louvers, see Figure 1. The value of the angles of the small, disk-shaped louvers are the design variables. We associate one design variable with the opening angle of a group of louvers, instead of individual louvers. This simplification serves three purposes: it reduces the dimension of the optimization problem, guarantees a more unified visual effect of the façade, and standardizes the façade elements for easier construction. We further reduce the optimization problem complexity with two techniques from systems architecture, namely decomposition and integration [14]. For an individual room inside the building, lighting conditions depend only on the part of the façade adjacent to that room. Hence, we can decompose the problem by considering each room separately, and find the optimal configuration of the corresponding disk-shaped louvers using the RBF method. Finally, we integrate these partial solutions to obtain a global optimal configuration for the façade as a whole. Accordingly, in the following we consider only one individual room.

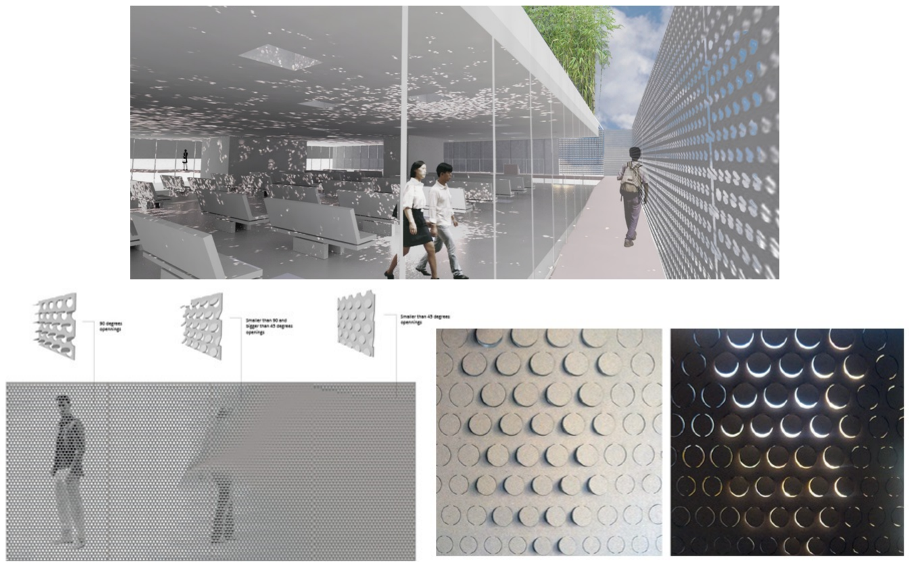
90 degrees opennings
Smaller than 90 and bigger than 45 degrees opennings
Smaller than 45 degrees opennings

Fig. 1 Façade of the church building with disk-shaped louvers.

Each of our design variables represents the opening angle of a group of louvers, and takes values between 0° (louvers closed) and 180° (louvers fully open). Daylighting is simulated with DIVA, a plugin for the 3D-modelling software Rhinoceros™ 3D [10]. DIVA yields illuminance values inside the building. Typically, running such a daylighting simulation is very time consuming and may take up to several hours. To define an optimization problem we need an objective function. In other words, given an output of the lighting simulator, how can we assess its quality? The quality of given lighting conditions is assessed in terms of two metrics calculated in DIVA. The first one is the Useful Daylight Illuminance (UDI). UDI is defined as [16]:

> the annual occurrence of illuminances across the work plane that are within a range considered "useful" by occupants. The range considered "useful" is based on a survey of reports of occupant preferences and behaviour in daylit offices with user operated shading devices. Daylight illuminances in the range 100-300 lux are considered effective either as the sole source of illumination or in conjunction with artificial lighting. Daylight illuminances in the range 300 to around 3,000 lux are often perceived either as desirable or at least tolerable.

We compute UDI values through a time-consuming simulation. The output is represented as false-color map.

The second metric is glare. An example of the glare values over one year, with time slots of one hour, is provided in Figure 2. The glare values are in the range $[0,1]$, and values greater than or equal to 0.45 are considered intolerable.

We define our objective function in terms of UDI and glare. Let s be the output of a lighting simulation. Let $U(s)$ be the value of UDI normalized to be in the range $[0,1]$, $G(s)$ be a value in $[0,1]$ expressing the average quality of the glare values over

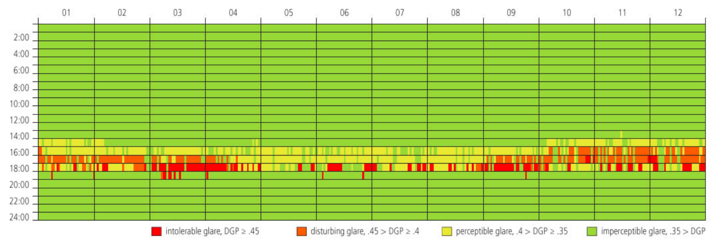

Fig. 2 Example of the output of a simulation for glare on a room over one year.

one year, and $T(s)$ equal to 1 if there are too many large (intolerable) glare values, and 0 otherwise. The objective function we want to maximize is expressed as:

$$f(s) = \alpha U(s) + (1-\alpha)G(s) - \alpha(1-\alpha)|U(s) - G(s)| - T(s), \quad \alpha \in (0,1), \quad (1)$$

where the term $\alpha(1-\alpha)|U(s)-G(s)|$ is used to penalize solutions presenting good values of UDI and bad values of glare, or vice-versa. The parameter α is used to decide the relative weights of glare and UDI for the considered room, and this decision is taken by the architects. The values of $f(s)$ are in the range $[-1,1]$, and negative solutions are assumed infeasible. Notice that $f(s)$ is not defined in terms of the design parameters (i.e., the angles of the disk-shaped louvers), but it can be computed from the output of the simulations.

Our goal is to optimize the design parameters (opening angles of the disk-shaped louvers) to achieve an optimal value for this objective function. Note that a designer can change the relative importance of brightness versus glare, for example because the function of the room has changed, and build a new model of the objective function and hot-start the optimization process without re-running the simulations. The designer can thus explore different design scenarios with a minimal need for time-intensive computations. This flexibility is key in the architectural design process, which typically involves not only changes to the solution but also changes to the design objectives [20].

3 The Radial Basis Function method

The problem discussed above can formally be cast into the following general form:

$$\min f(x), \text{ subject to } x \in [x^L, x^U], \quad (2)$$

where $f : \mathbb{R}^n \to \mathbb{R}$, and $x^L, x^U \in \mathbb{R}^n$ are vectors of lower and upper bounds on the decision variables that define the design space. The analytical expression for f is unknown and function values are only available through an oracle. We assume that f is continuous with respect to all variables. The method we discuss can be extended to take into account integer variables and additional (explicit) constraints to enforce known relationships between the decision variables, but we keep this simplified formulation for ease of exposition. Our implementation, however, can handle integer variables.

The main idea of the RBF method proposed in [6] is to use radial basis functions (i.e., real-valued functions whose value only depends on the distance from a center x_i, that is, functions of the form $\phi(\|x-x_i\|)$) to build a surrogate model that interpolates the known points. The next evaluation point is chosen by selecting a target objective function value f_k^*, and finding the point in the design space that minimizes the "bumpiness" of the resulting interpolant, if we were to add an interpolation node at the chosen target level. Thanks to the properties of RBFs, we can find an analytical expression for a measure of bumpiness. For space reasons, here we provide a brief overview of the method, skipping most of the details. The interested reader can refer to [6, 9, 19].

Let $\Omega := \{x \in [x^L, x^U]\} \subset \mathbb{R}^n$. Given k distinct points $x_1, \ldots, x_k \in \Omega$, the RBF interpolant s_k is defined as:

$$s_k(x) := \sum_{i=1}^{k} \lambda_i(\|x-x_i\|)^3 + ax + b, \tag{3}$$

where $\lambda_1, \ldots, \lambda_k \in \mathbb{R}$ are the coefficients of the RBFs, $\|\cdot\|$ is the Euclidean norm, $a \in \mathbb{R}^n, b \in \mathbb{R}$. In the expression above we are using the RBF $\phi : \mathbb{R}_+ \to \mathbb{R}, \phi(r) = r^3$. In this paper we only discuss the cubic RBF $\phi(r) = r^3$, which gives the best performance in our computational experiments, but in general other choices of the RBF are possible (e.g., thin-plate splines). The parameters λ, a, b of the function $s_k(x)$ that interpolates the k points $(x_i, f(x_i)), \forall i \in \{1, \ldots, k\}$ can be determined by solving a linear system (see e.g., [6]). In the following, we will keep referring to s_k as our surrogate model of f after k function evaluations. An overview of the algorithm is given in Figure 3.

This methods finds a solution within ε of the global optimum, provided that the unknown function is sufficiently smooth, the algorithm is executed for a large enough number of iterations, and the choice of target values f_k^* satisfies a technical condition [6].

We still need to address the following questions:

- How do we choose the initial points needed to create the first interpolant?
- How do we choose the next point to evaluate through the oracle using a target function value?

3.1 Selection of the initial points

The initial sample points must be linearly independent to guarantee existence of an interpolant. A commonly used strategy [6, 8, 9] is to choose corner points of the box Ω. Taking all the 2^n corner points is impractical if n is large (remember that oracle is evaluated at each of these points). A commonly used strategy is to use a

```
input  : oracle for $f$, domain $[x^L, x^U]$, maximum # evaluations $n_{\max}$
output: best solution found within $n_{\max}$ evaluations

evaluate $f$ at $k_0$ starting points $x_1 \dots x_{k0}$;
$i \longleftarrow \arg\min\{f(x_i), \forall i \in \{1, \dots, k_0\}\}$;
$(x^*, f^*) \longleftarrow (x_i, f(x_i))$;
$k \longleftarrow k_0$;
while $k < n_{\max}$ do
    compute the interpolant $s_k(x)$ using the $k$ points evaluated so far;
    choose a target value $f_k^*$ and select the next evaluation point $x_{k+1}$;
    evaluate $f(x_{k+1})$ through the oracle;
    if $f(x_{k+1}) < f^*$ then
        $x^* \longleftarrow x_{k+1}$;
        $f^* \longleftarrow f(x_{k+1})$;
    end
    $k \longleftarrow k + 1$;
end
return $(x^*, f^*)$
```

Fig. 3 The RBF method.

Latin Hypercube experimental design of size $n+1$; this is our default strategy in practice. Other choices are possible, see e.g. [9]. Points sampled according to these strategies may not be feasible if there are integer variables and/or additional explicit constraints. [9] suggests sampling more points than strictly necessary (i.e., $> n+1$), and picking the first $n+1$ feasible ones, rounding integer variables if necessary. In practice, feasibility in this case must not be too difficult to obtain, otherwise solving the initial problem (2) is essentially hopeless.

3.2 Selection of the next evaluation point

Arguably, the most important aspect of the algorithm is how to choose the next evaluation point. The idea is to define a target objective function value f_k^* lower than the best known objective value f^*, and find the point x_{k+1} in the domain where the surrogate model is more likely to take the value f_k^*. More precisely, this point will be the one for which the interpolant $s_{k+1}(x)$ that interpolates the previous points $(x_i, f(x_i)), \forall i \in \{1, \dots, k\}$ and (x_{k+1}, f_k^*), is the least “bumpy”.

We give an example in Figure 4. Given some points at which the function value is known (blue circles), i.e., the previous iterates, and a target objective function value f_k^* (the dashed line), we can obtain different surrogate models depending on the choice of the point x_{k+1} where the function should assume value f_k^*. In the example, the green surrogate model is less bumpy than the red one. The next evaluation point will be the one yielding the least bumpy surrogate model reaching the value f_k^*. Intuitively, this favors the “simplest” model that interpolates the current points and achieves the target f_k^*.

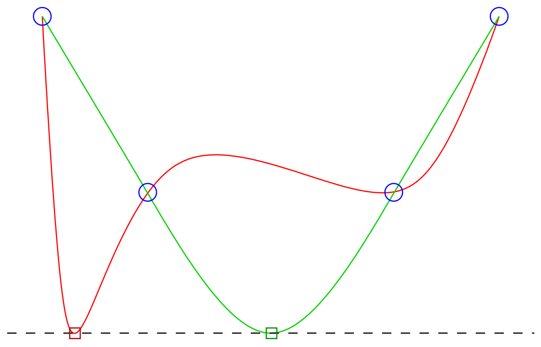

Fig. 4 Example showing two interpolants obtained by choosing a different value of x_{k+1} for a given target value f_k^* (dashed line). The red surrogate model is more bumpy than the green one.

As stated earlier, we can define a measure of bumpiness using the properties of RBFs. Indeed, we can write a nonlinear, nonconvex optimization problem to compute the point that yields the least bumpy interpolant at function value f_k^* [6, 9]. Even though solving this auxiliary problem is not easy in general, we have access to first and second order information, hence it is not as difficult as the original black-box problem. Moreover, it is usually sufficient to find a good local minimum.

We still need to specify how to choose the target objective function value f_k^*. In order to guarantee the convergence of the method, [6] proposes a cyclic strategy to choose f_k^*:

- **inf step**: the target objective function value is set to $-\infty$. The new point x_{k+1} minimizing the bumpiness will be in a large gap between existing interpolation points. This is an *exploration* phase: the algorithm tries to improve the surrogate model in unknown parts of the domain.
- **local step**: the target objective function value is set to $\min_x s_k(x)$, hence the new evaluation point x_{k+1} will be the minimum of the interpolant. This is an *exploitation* phase: we try to find the best objective function value based on the current surrogate model.
- **global step**: the target objective function value is between the two extremes $-\infty$ and $\min_x s_k(x)$. In this case, there is a balance between improving the model quality and finding the minimum.

3.3 Extensions

We describe here some extensions to the basic RBF algorithm. Our preliminary computational evaluations shows that these can significantly improve performance of the method, but for space reasons we only give a brief overview.

3.3.1 A measure of model quality

One of the drawbacks of the RBF method is that there is no mechanism to assess the accuracy of the surrogate model. The algorithm uses a cyclic search strategy that oscillates between global search and local search (possibly with intermediate steps), regardless of model quality. However, one can expect that if the current model is a poor approximation of the true objective function, the local search phase is unlikely to find a point with a good objective function value.

We propose to assess model quality using a cross validation scheme. Cross validation is a commonly used model validation technique in statistics and machine learning, and is discussed virtually by any textbook in the field. Given a data set, cross validation consists in using part of the data set to fit a model, and testing its quality on the remaining data. The process is then iterated by choosing different parts of the data set for model fitting and for testing.

We say that a measure of model quality $\mathscr{M}$ is *convergent* if $\lim_{k\to\infty} \mathscr{M}(s_k) = \lim_{k\to\infty} \int |s_k(x) - f(x)|\,\mathrm{d}x$. This implies that $\mathscr{M}$ converges to a reliable measure of difference between the surrogate model and the true objective function f. We are implicitly assuming that the integral exists and is well-defined. It is not hard to prove that $\lim_{k\to\infty} \int |s_k(x) - f(x)|\,\mathrm{d}x = 0$, due to properties of the algorithm. It is easy to design convergent measures, but from a practical point of view we are interested in how closely the sequence $\{\mathscr{M}(s_k)\}_k$ follows $\{\int |s_k(x) - f(x)|\,\mathrm{d}x\}_k$.

The method we advocate is as follows. Remember that function values are known at k points $x_1,\dots,x_k$. Then, for $j \in \{1,\dots,k\}$ we can fit a surrogate model $\tilde{s}_k$ to the points $\{x_i, \forall i \in \{1,\dots,k\} \,|\, i \neq j\}$, and evaluate the peformance at $(x_j, f(x_j))$. In particular, we compute the value $|\tilde{s}_k(x_j) - f(x_j)|$, which is zero if the model is a perfect fit, and > 0 otherwise. We can then average these statistics for $j \in \{1,\dots,k\}$ to compute an overall model quality score: this will be our measure $\mathscr{M}$. Our approach can be categorized as leave-one-out cross-validation. While leave-one-out cross-validation is often computationally expensive, we can show that $\mathscr{M}$ can be computed efficiently by solving a sequence of k linear programs, where each linear program in the sequence can be warmstarted.

We use this measure of quality to dynamically select among different surrogate models of the objective functions, obtained using different RBFs. In particular, periodically during the optimization process we select which model seems to yield the best predictive power globally (for the global search phase of the algorithm) and locally around the optimum (for the local search phase of the algorithm).

Furthermore, at the end of the algorithm we can output the surrogate model and a corresponding "confidence level" to quantify how reliable we expect the surrogate model to be. This has important practical implications: the surrogate model can be used to perform sensitivity analysis around the optimal solution, i.e., explore how the objective function changes if the design parameters are perturbed, without having to resort to new (expensive) oracle evaluations; see the discussion in Section 2. The confidence level informs the user about the reliability of such sensitivity analysis.

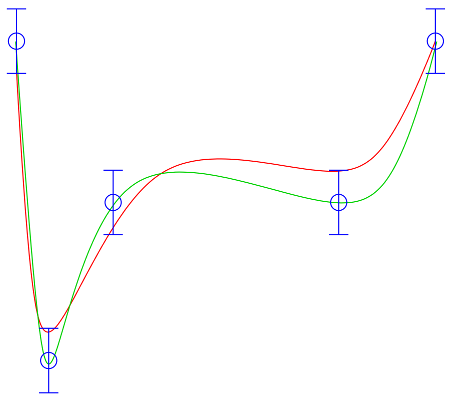

Fig. 5 Function evaluations affected by errors: the values returned by LQ simulations are the blue dots. The green function interpolates exactly at those points, the red function is a less bumpy interpolant, and is still within the allowed error tolerances.

3.3.2 Noisy functions evaluation

In the application context studied in this paper, there is a trade-off between computing time and the accuracy of the black-box function f. This is a common situation in practice: the simulation software used to compute f can often be parameterized to achieve different levels of accuracy. Our target application has this characteristic.

To speed-up the optimization process, we would like to exploit low accuracy and therefore faster simulations. In the following, we will denote by LQ ("low quality") the low accuracy, high speed setting, and by HQ ("high quality") the high accuracy, low speed setting. Our assumption is that function evaluations computed in LQ are affected by an unknown error, whereas function evaluations computed in HQ represent the true objective function $f(x)$. The specifics of what LQ and HQ correspond to (and as a consequence the estimation of the error) are application-dependent. Further, we assume that we have an estimate of the maximum relative (or absolute) error ε_r. Such an estimate is typically based on domain knowledge. Notice that the surrogate model interpolating LQ points should be allowed to deviate from the values returned by the oracle by an amount within ε_r. Consider the example of Figure 5: if the points were HQ, i.e., considered "exact", the interpolant would be the green function. If we assume that function evaluations are affected by noise, and the true function values may be located anywhere in an interval around their true value, then we prefer the red surrogate model, because it is the least bumpy among the models that interpolate within the given noise intervals. We can show that the RBF surrogate model under noisy function evaluations can be determined by solving a convex quadratic problem. We advocate the following approach: at a first stage we solve (2) in LQ until a termination condition is met, based on number of function evaluation, an estimation of model quality (for example employing the cross validation explained in the previous section), or stalling. At a second stage (2) is reoptimized performing additional function evaluations in HQ, still allowing the LQ function evaluations to vary by at most ε_r. During the local search phase, if the minimum of the interpolant occurs at a LQ point, we re-evaluate the same point in HQ.

Function	Dimension	Our RBF	TOMLAB RBF	DE
Branin	2	36	26	1190
Goldstein-Price	2	30	27	1018
Hartman 3	3	42	22	476
Hartman 6	6	114	87	7220
Shekel 5	4	116	96	6400

Table 1 Number of function evaluations needed to find a value within 1% from the global optimum for different black-box methods on instances from the Dixon-Szegö test set.

3.4 Open-source implementation

We implemented the RBF method for MATLAB™/Octave and Python. The former requires AMPL® license [5]. Our implementation is open-source and is available from the authors[3]. To solve the nonlinear (mixed-integer in the presence of integer variables) optimization problems generated during the various steps of the algorithm a nonlinear solver is also needed. In our tests we use BONMIN [2] with the nonlinear solver IPOPT [22]. Currently, our algorithm can handle problems in any dimension, and it can handle both continuous and integer variables. Clearly an increase of the problem dimension significantly increases the difficulty of finding a global optimum. Besides our own version, we are aware of only one available implementation of the RBF method: the commercial TOMLAB® toolkit for MATLAB™.

To give an idea of the performance of our implementation, we show in Table 1 the number of iterations needed to find a solution within 1% from the global optimum for some well-known instances of the literature (i.e., the Dixon-Szegö [4]) having multiple global and local optima and dimension up to 6. We compare the number of evaluations of the oracle of our method, the best results obtained by TOMLAB® as reported by [8], and an evolutionary algorithm (DE [21]). We remark that results for these test were obtained with the default parameters of our implementation, i.e., an "out-of-the-box" run without any sort of tuning. The number of function evaluations is comparable to the best results of TOMLAB® as given by [8]. It is more than one order of magnitude better than an evolutionary algorithm, which is common practice for optimization in architectural design.

To summarize, our open-source implementation is competitive with state-of-the-art commercial software, even without any parameter tuning. At the present stage, the integration with architectural design software is still in progress. However, we believe that this paper shows that the application of state-of-the-art optimization methods in the architectural design context could prove an invaluable decision-making tool.

[3] At the moment of writing this paper, the license terms are still to be defined, but we expect a BSD-derivate that will be free for academic use.

References

1. Björkman, M., Holmström, K.: Global optimization of costly nonconvex functions using radial basis functions. Optimization and Engineering **1**(4), 373–397 (2000)
2. Bonami, P., Lee, J.: `BONMIN` user's manual. Tech. rep., IBM Corporation (2007)
3. Delanda, M.: Deleuze and the use of the genetic algorithm in architecture. In: Contemporary techniques in architecture, pp. 9–11. London New York: Wiley-Academy (2002)
4. Dixon, L., Szego, G.: The global optimization problem: an introduction. In: L. Dixon, G. Szego (eds.) Towards Global Optimization, pp. 1–15. North Holland, Amsterdam (1975)
5. Fourer, R., Gay, D.: The AMPL Book. Duxbury Press, Pacific Grove (2002)
6. Gutmann, H.M.: A radial basis function method for global optimization. Journal of Global Optimization **19**, 201–227 (2001)
7. Hemker, T.: Derivative free surrogate optimization for mixed-integer nonlinear black-box problems in engineering. Master's thesis, Technischen Universität Darmstadt (2008)
8. Holmström, K.: An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization. Journal of Global Optimization **41**(3), 447–464 (2008)
9. Holmström, K., Quttineh, N.H., Edvall, M.M.: An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer constrained global optimization. Optimization and Engineering **9**(4), 311–339 (2008)
10. Jakubiec, J.A., Reinhart, C.F.: DIVA 2.0: Integrating daylight and thermal simulations using Rhinoceros 3d, DAYSIM and EnergyPlus. In: proceedings of the 12th International IBPSA Conference on Building simulation, pp. 2202–2209 (2011)
11. Jones, D., Schonlau, M., Welch, W.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization **13**(4), 455–492 (1998)
12. Kaveh, A., Laknejadi, K.: A new multi-swarm multi-objective optimization method for structural design. Advances in Engineering Software **58**, 54 – 69 (2013)
13. Kicinger, R., Arciszewski, T., Jong, K.D.: Evolutionary computation and structural design: A survey of the state-of-the-art. Computers and Structures **83**(23-24), 1943–1978 (2005)
14. Krob, D.: Modelling of complex software systems: A reasoned overview. In: Proceedings of the 26th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems, FORTE'06, pp. 1–22. Springer-Verlag, Berlin, Heidelberg (2006)
15. Lin, S.H.E., Gerber, D.J.: Designing-in performance: A framework for evolutionary energy performance feedback in early stage design. Automation in Construction **38**, 59 – 73 (2014)
16. Mardaljevic, J., Andersen, M., Roy, N., Christoffersen, J.: Daylighting, Artificial Lighting and Non-Visual Effects Study for a Residential Building. Tech. rep., Loughborough UK (2012)
17. Miles, J.: Genetic algorithms for design. In: Z. Waszczyszyn (ed.) Advances of Soft Computing in Engineering, pp. 1–56. Springer, Vienna (2010)
18. Oxman, R.: Performance-based design: current practices and research issues. International Journal of Architectural Computing **6**(1), 1–17 (2008)
19. Regis, R., Shoemaker, C.: Improved strategies for radial basis function methods for global optimization. Journal of Global Optimization **37**, 113–135 (2007)
20. Rittel, H.W.J., Webber, M.M.: Dilemmas in a general theory of planning. Policy Sciences **4**(2), 155–169 (1973)
21. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization **11**, 341–359 (1997)
22. Wächter, A., Biegler, L.T.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Mathematical Programming **106**(1), 25–57 (2006)
23. Woodbury, R.F., Burrow, A.L.: Whither design space? AI EDAM **20**(2), 63–82 (2006)