Flexibility Index of Black-Box Models with Parameter Uncertainty through Derivative-Free Optimization

Fei Zhao, Ignacio E. Grossmann*

Center for Advanced Process Decision-Making, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213

Salvador García-Muñoz, Stephen D. Stamatis

Synthetic Molecule Design and Development, Lilly Research Laboratories, Indianapolis, IN 46285

Abstract

The existing methods of flexibility index are mainly based on mixed-integer linear or nonlinear programming methods, making it difficult to readily deal with complex mathematical models. In this article, a novel solution strategy is proposed for finding a reliable upper bound of the flexibility index where the process model is implemented in a black box that can be directly executed by a commercial simulator, and also avoiding the need for calculating derivatives. Then, the flexibility index problem is formulated as a sequence of univariate derivative-free optimization (DFO) models. An external DFO solver based on trust-region methods can be called to solve this model. Finally, after calculating the critical point of the model parameters, the vertex enumeration method and two gradient approximation methods are proposed to evaluate the impact of process parameters and to evaluate the flexibility index. A reaction model is studied to show the efficiency of the proposed algorithm.

Keywords: process design; flexibility index; derivative-free optimization; black-box model; Python

*Correspondence concerning this article should be addressed to grossmann@cmu.edu.

1. Introduction

Most chemical plants are subject to uncertainties and variations during operation. Flexibility analysis has been proposed as a quantitative framework for measuring the capability of feasible operation of a given design over a range of values for the uncertain parameters[1]. Flexibility has been studied over three decades and as an important property that must be incorporated into a design, and it has been applied in several industrially inspired case studies[2-5], such as process synthesis of air separation, heat exchanger network design, and supply chain networks. For reviews of previous work on flexibility analysis, see references[6-9].

Ostrovsky *et al.*[10] and Rooney and Biegler[11] extended flexibility analysis by grouping the parameters into process parameters and model parameters[12]. Process parameters refer to degrees of freedom or manipulated variables in an industrial process. Process variables are measured, known and can be set within controller tolerance to a desired value. In typical operation, process parameters have two types of defined ranges: (1) The qualification range given by the physical constraints in the equipment and safety considerations (e.g. maximum volumetric flow an air blower can deliver); (2) The operational range, this is the allowed envelope of operation to ensure product quality. The combination of all envelopes of operation for all process parameters is defined as the flexibility region. The operational range is always contained within the qualification range. On the other hand, a model parameter is an a-priori unknown quantity in a mathematical representation of the process. Although the values of these parameters for an industrial process model are mostly unknown, their values are typically estimated from measurements. The model parameter estimation exercise results in an expected value and a confidence interval for each of the model parameters which are typically assumed to follow a Gaussian distribution. The size of the confidence interval is reflective of the uncertainty in the parameter values.

In this work, we aim to identify the operational range for all process parameters (a.k.a. the flexibility region) within their qualification range, and subject to a model of the process, the expected values for the model parameters and their range of uncertainty, and a set of constraints to the product attributes driven by quality needs. Flexibility index[13,14] is a concept proposed to describe such the operational range for all process parameters, which can represent a maximum scaled departure of process parameters from their nominal value, that is, a largest hyper-rectangle inscribed in the feasible space, inside which, the steady-state operation can be attained by adjusting the control variables. The flexibility index problem is commonly formulated as a multi-level optimization model with existing approaches relying on the solution of mixed integer linear or nonlinear programming solvers. Considering the computational complexity, a direct search algorithm by enumerating all vertices of the hyper-rectangle, that is, the vertex direction search method[13], was proposed. Since the computational effort of the vertex search method is generally proportional to the number of the uncertain parameters, an implicit enumeration scheme with a branch-and-bound procedure was developed to accelerate the search process[14] for cases with convex feasible spaces where the critical points always correspond to the vertices. For nonconvex regions the vertex searches are not guaranteed to provide rigorous solutions. In order to avoid the convexity assumption, Grossmann and Floudas[15] developed an active constraint strategy, where the flexibility index problem can be reformulated as a mixed-integer linear programming (MILP) model or mixed-integer nonlinear programming (MINLP) model by applying the Karush-Kuhn-Tucker (KKT) conditions. However, for a large-scale or complex design problem, it is often hard to solve the corresponding MINLP model and finding the global optimum cannot be guaranteed for the nonconvex cases. Li et al.[4] developed a direction matrix to search the critical points. Through incorporating a simulated annealing algorithm and a decoupling strategy, the flexibility index of a large-scale system can also be obtained.

The surrogate-based and sampling-based methods[16,17] can provide optional ways to ease the complexity of large-scale or complex problems. Banerjee and Ierapetritou[18] proposed a high dimensional model representation (HDMR) approach based on the input-output mapping strategy to determine the flexibility space. Next, Boukouvala and Ierapetritou[19] applied a Kriging-based approach to substitute a surrogate model for the original flexibility function. Laky et al.[20] developed two algorithms to extend the flexibility test and index formulation to identify the probabilistic design space and replace the simulation-based analysis. García Muñoz et al.[21] defined the probabilistic design space through creating a grid of sample points for the process parameters, and Kucherenko et al.[22] proposed the acceptance-rejection method that outperformed the exhaustive sampling achieving a two orders of magnitude speed-up by using metamodeling and adaptive sampling in the design space determination. Zheng et al.[23] created a surrogate model to simplify the system, and then applied a symbolic computation method to approximate the design space. An iterative procedure, including surrogate modeling, model updating, sampling points, boundary checking, is used to describe the final design space.

In summary, however, the existing approaches to evaluate the flexibility index mainly rely on solving MILP or MINLP problems, which makes it difficult to handle large-scale or complex problems. Moreover, the realistic industrial processes, e.g., chemical manufacturing, are often operated in the presence of complex and uncertain dynamics which complicate the application of traditional flexibility analysis. The complexity of the design model and the existence of dynamics can increase the difficulty of solving the flexibility index problems. Based on the steady-state flexibility analysis framework, Dimitriadis and Pistikopoulos[24] proposed a unified approach for the quantification of feasibility and flexibility of systems that operate dynamically under time-varying uncertainty. If the uncertainty profiles are given, the problem can be reduced such that traditional flexibility analysis methods can be used; if not given, a discretization scheme for the differential equations is used to transform it into a mixed-integer nonlinear programming problem. Adi and Chang[25] dealt with flexibility analysis for a system described by a set of differential algebraic equations (DAEs). By adopting a differential quadrature technique to approximate the DAEs with equality constraints, any solution strategy for the conventional steady-state flexibility analysis is applicable. For integrated process and control systems design, Mohideen et al.[26] proposed a framework for addressing the optimal design problem of dynamic systems under uncertainty in which the flexibility aspects were formally incorporated in a multiperiod design subproblem coupled with feasibility analysis of time-varying systems. The mixed-integer dynamic optimization algorithm was implemented in GAMS. Thus, besides the issue of model size, the uncertain dynamics also create another obstacle to solve the flexibility index problems.

An additional challenge for large scale and/or noisy systems is that the derivatives required to solve the optimization problem are neither symbolically nor numerically available. If the derivative information is unavailable, the active set method based on the KKT conditions cannot be used to find the flexibility index. The complexity of mechanistic models in industrial practice is such that it becomes difficult to calculate or derive algebraic derivatives of those process models. The restrictions of commercial simulation software are another barrier to having the algebraic model and its derivatives explicitly available to the practitioner. Reimplementing process models in multiple platforms to apply different analyses is an effort and time intensive activity often prohibitive by the tight timelines in a live project. Hence, it is desirable to use process models as black boxes, because they can be directly simulated on their original commercial software, explicit derivatives however are still not available. Derivative-Free Optimization (DFO) methods were designed to solve such the black-box models with no need to compute the derivatives numerically or algorithmically. Specific to the pharmaceutical industry, Boukouvala and Ierapetritou[27] have reported the use of surrogate models as a way to creating approximate models that can be handled by MINLP algorithms. The direct use of black box models coupled with DFO algorithms has be reported by Boukouvala et al.[28] and Boukouvala et al.[29], the latter for pressure swing adsorption. The use of black box models has also been reported in other areas. For instance, Peaceman[30] applied black-box simulation method to deal with the petroleum reservoir model, while Shiehnejadhesar et al.[31] and Nagy et al.[32] used CFD to analyze furnace models.

While the application of DFO for black box models has been considered in a number of areas as described above, to our knowledge they have not been applied to the area of flexibility analysis. DFO method is an area with a long history and current rapid growth[33-36]. Rios and Sahinidis[37] present a review of derivative-free algorithms, followed by a systematic comparison of 22 related DFO solvers. Note that DFO methods are sometimes employed for convenience rather than by necessity, because the decision to use a DFO method typically can limit the performance (as measured by accuracy, computational expense, or problem size), one might expect from gradient-based optimization methods[38]. In addition, as the dimension of the model increases, the reliability of the DFO methods decreases. Theoretically, the DFO methods can achieve the best performance for the univariate or bivariate models. If the optimization object is a low-dimensional model, the DFO methods can be applicable.

Motivated by the convenience of DFO methods, we propose in this work to apply these techniques to process models that are treated as a black box; thus, any existing models using commercial simulators can be used, and also avoiding the need for calculating derivatives. The input and output information of the black-box model can be used for an external DFO solver developed by trust-region methods. First, before solving the flexibility index, the critical point of the model parameters is determined by calculating the worst constraint violations. Three different methods, i.e., vertex enumeration method and two gradient approximation methods, are proposed to evaluate the upper bound of flexibility index. In order to implement the proposed algorithm, the performance of the DFO methods and DFO solvers is analyzed in advance, and the entire procedure has been programmed and can be executed automatically.

The remainder of this paper is organized as follows. Section 2 proposes to calculate the critical point of model parameters by finding the worst constraint violations. Section 3 describes the DFO model of flexibility index and provides geometric interpretations; Section 4 introduces the DFO methods and analyzes the properties of the DFO solvers. Section 5 proposes three different methods to find the flexibility index. A reaction model is given in Section 6. Section 7 concludes the paper.

2. Critical point of model parameters

For the flexibility analysis we will not consider recourse decisions[1] as this is the current practice in the pharma industry. For a given plant design d, the flexibility constraint with no recourse can be described as a logic expression as follows[12]:

$$\forall \theta \in T_p \{\forall \eta \in T_m, \forall j \in J [g_j(\theta, \eta) \leq 0]\} \tag{1}$$

where θ and η represent process and model parameters, respectively. Process parameters include for instance feed flow rates, pressures, temperatures, and concentrations. Model parameters include for instance activation barriers, preexponential factors, heats of reaction, solubility, Henry's law constants, and analytical response factors. Eq. (1) states that for any possible realization of the process parameters in T_p and any realization of the model parameters in T_m, all of the individual constraints g_j, $j \in J$, should be satisfied. To evaluate the flexibility index, Eq. (1) can be equivalently reformulated by the use of global max operator[1], leading to Eq. (2).

$$\chi = \max_{\theta \in T_p} \max_{\eta \in T_m} \max_{j \in J} g_j(\theta, \eta) \leq 0 \tag{2}$$

χ is defined as flexibility function. Through the following property of max operators,

$$\max_x \max_y f(x, y) \Leftrightarrow \max_y \max_x f(x, y) \tag{3}$$

which means that the order of the maximization problems of Eq. (3) is interchangeable. Therefore, the problem in Eq. (2) can be equivalently reformulated as follows.

$$\max_{\theta\in T_p}\max_{\eta\in T_m}\max_{j\in J} g_j(\theta,\eta) \Leftrightarrow \max_{\theta\in T_p}\max_{j\in J}\max_{\eta\in T_m} g_j(\theta,\eta) \tag{4}$$

The flexibility index F can be defined as the largest value of δ for the uncertainty set of process parameters, as shown in Eq. (5),

$$T_p(\theta) = \{\theta : \theta^N - \delta \cdot \Delta\theta^- \leq \theta \leq \theta^N + \delta \cdot \Delta\theta^+\} \tag{5}$$

where θ^N are the nominal values of the process parameters, and the range $[\Delta\theta^-, \Delta\theta^+]$ represents the allowable range of operation for each. At $\delta = F$, $T_p(\theta)$ can describe the largest hyperrectangle which is inscribed within the feasible range of the process parameters. $T_m(\eta)$ represents the variability of the model parameters, which is commonly described by the hyperrectangle but not restricted to any specific type of set. For simplicity, in this work, we assume that the model parameters are independent and not correlated. Hence, we assume that they are simply described by upper and lower bounds.

$$T_m(\eta) = \{\eta : \eta^L \leq \eta \leq \eta^U\} \tag{6}$$

Therefore, the flexibility index problem can be described by the following optimization model.

$$\begin{aligned} F = &\max_{\delta\in\mathbb{R}^+} \delta \\ s.t.\ &\chi = \max_{\theta\in T_p}\max_{j\in J}\max_{\eta\in T_m} g_j(\theta,\eta) \leq 0 \\ &T_m(\eta) = \{\eta : \eta^L \leq \eta \leq \eta^U\} \\ &T_p(\theta) = \{\theta : \theta^N - \delta \cdot \Delta\theta^- \leq \theta \leq \theta^N + \delta \cdot \Delta\theta^+\} \end{aligned} \tag{7}$$

where the maximization problem in χ determines the worst constraint violation, and the critical model parameters can generate the worst constraint violation. Thus, if specifying the process parameters as θ^N, the worst value of the constraints $g_j(\theta^N, \eta)$ can be determined within the lower and upper bound of the model parameters.

For example, if the constraints are monotonic with respect to the model parameters, the worst constraint violation will be located at the lower bound or the upper bound. As shown in Figure

1, in the range of η, $[\eta^L, \eta^U]$, the maximum values of the constraints g_2 and g_3 are located at η^U, and the maximum value of g_1 occurs at η^L. By comparison, the critical point is $\eta^c = \eta^U$.

If the constraints are not monotonic with respect to the model parameters, the maximum values of the constraints may occur at the peaks. As shown in Figure 2, because

$$g_2(\eta^2) < g_1(\eta^1) < g_3(\eta^3)$$

the worst constraint violation of $\{g_1, g_2, g_3\}$ is the value of g_3 at η^3, thus, the critical point of η can be defined as $\eta^c = \eta^3$.

If the dimensionality of the model parameters is p, the critical point will be a vector of the critical value of each model parameter, that is,

$$\eta^c = [\eta_1^c, \dots, \eta_i^c], \eta_i^c \in [\eta_i^L, \eta_i^U], i = 1, \dots, p \tag{8}$$

After substituting η^c into the flexibility function χ, the original three-level optimization can be simplified as one-level optimization problem.

$$\max_{\theta \in T_p} \max_{j \in J} \max_{\eta \in T_m} g_j(\theta, \eta) \Longrightarrow \max_{\theta \in T_p} g_j(\theta, \eta^c) \tag{9}$$

Note that Eq. (9) can significantly simplify the computational complexity of the flexibility index, especially for flexibility models with many model parameters. The detailed algorithm to calculate the worst constraint violation and the critical point of model parameters is summarized as follows.

1) Fix all of the process parameters as the nominal values, θ^N, and express the constraints as $g_j(\theta^N, \eta)$;
2) For each model parameter η_i, the remaining model parameters are specified as the mean values of the corresponding ranges; then, by solving the univariate optimization model $u_j^i = \max_{\eta_i \in [\eta_i^L, \eta_i^U]} g_j(\theta^N, \eta_i)$, the maximum value of each constraint is obtained, e.g., if there are two constraints, two values can be obtained, $\{u_1^i, u_2^i\}$;

3) The maximum value in $\{u_j^i\}$, i.e., $\max_{j \in J}\{u_j^i\}$, is the worst value of the constraint j with respect to η_i, and the corresponding optimal solution is defined as the critical value, η_i^c;
4) After all the model parameters are processed, the critical point of the model parameters can be obtained, $\eta^c = [\eta_1^c, \dots, \eta_i^c]$.

3. DFO model of flexibility index

3.1. DFO model of flexibility index based on vertex directions

Once η^c is substituted into Eq. (7), χ can be further simplified to a one-level optimization problem and the bound constraints of η can be removed.

$$\begin{aligned} & F = \max_{\delta \in \mathbb{R}^+} \delta \\ & s.t.\ \chi = \max_{\theta \in T_p} g_j(\theta, \eta^c) \leq 0 \\ & \qquad T_p(\theta) = \{\theta : \theta^N - \delta \cdot \Delta\theta^- \leq \theta \leq \theta^N + \delta \cdot \Delta\theta^+\} \end{aligned} \tag{10}$$

Equation (10) is a simplified optimization model that only relates to δ and θ. $T_p(\theta)$ represents a hyperrectangle centering on the location of θ^N. As δ increases, the hyperrectangle will be gradually larger until it is inscribed within the feasible range of θ. If the dimensionality of the process parameters is q, the hyperrectangle has a total of 2^q vertices. Assuming that the critical point of the process parameters corresponds to some vertex of the hyperrectangle, the vertex direction search method[15] can further simplify Eq. (10) as:

$$\begin{aligned} & F^k = \max \delta \\ & s.t.\ \theta = \theta^N + \delta \cdot \Delta^k \\ & \qquad g_j(\theta, \eta^c) \leq 0 \\ & \qquad 0 \leq \delta \leq 1 \end{aligned} \tag{11}$$

where k is the index of vertices, $k \in K$; for convenience, for the flexibility index problems, the deviations can be rewritten such that δ is restricted within 0 and 1; Δ^k represents the k^{th} vertex direction with the given deviations from the nominal point. F^k is the maximum deviation along the k^{th} vertex direction which touches the boundary of the feasible region. The flexibility index is then defined as

$$F = \min_{k \in K}\{F^k\} \tag{12}$$

Since $\theta = \theta^N + \delta \cdot \Delta^k$, and η^c has been obtained by calculating the worst constraint violations, g_j is a univariate function of δ, i.e., $g_j(\delta)$. In order to convert Eq. (11) to a form that is easily handled by general DFO solvers, a penalty coefficient M can be introduced into the objective function. Thus, the final flexibility index model is

$$\begin{aligned} &F^k = \min -\delta + M \cdot \textstyle\sum_j^J \left(\max\left(0, g_j(\delta)\right)\right) \\ &s.t.\ \ 0 \le \delta \le 1 \end{aligned} \tag{13}$$

The objective function in Eq. (13) is a type of exact penalty function, which is capable of finding the exact optimal solution. Zangwill[39], Fletcher[40] and Fiacco[41] proved that if M is sufficiently large, this penalty function will be exact, but not smooth, which means that it generally has a discontinuous first derivative at the local minimum. However, the solution of the penalty problem can yield the exact solution to the original problem for a sufficiently large finite value of M. Since we apply the DFO method to solve this model, the derivatives do not need to be calculated, and therefore the discontinuity is not an issue. The penalty coefficient M simply serves as a way to scale the constraint violation and its value need only be tuned for numerical stability. The penalty function algorithms generally need to increase the value of M sequentially, because we do not know exactly how big M should be. In this work, for simplification, M is set as 1000 for all cases. To verify the suitability of this parameter we test that $M = 1000$ yields the same results as for $M = 100$ and $M = 5000$.

If the constraints $g_j(\delta)$ are restricted within the lower bounds and the upper bounds, i.e., $A_j^L \le g_j(\delta) \le A_j^U$, the flexibility index model is

$$\begin{aligned} &F^k = \min -\delta + M * \textstyle\sum_j^J \left(\max\left(0, A_j^L - g_j(\delta)\right) + \max\left(0, g_j(\delta) - A_j^U\right)\right) \\ &s.t.\ \ 0 \le \delta \le 1 \end{aligned} \tag{14}$$

Eqs. (13) and (14) indicate that the proposed flexibility index model is a univariate model of δ, including a black-box objective function and a box constraint, and the DFO methods can be applied to solve this flexibility index model.

3.2. Geometric interpretation for the critical point of model parameters

Finding the critical point of the model parameters, i.e., η^c, is an important step to calculate the flexibility index, because all the model parameters will be fixed as η^c in the subsequent steps. The following linear example provides a geometric interpretation of the critical point. Figure 3(A) shows the profiles of g_1 and g_2, the nominal value of θ is 1, and the range of the model parameter η is [1.5, 3].

$$\begin{cases} g_1: \theta - \eta \leq 0 \\ g_2: -\theta - \frac{\eta}{3} + \frac{4}{3} \leq 0 \end{cases} \tag{15}$$

After fixing $\theta = 1$, Eq. (15) reduces to Eq. (16). The blue lines in Figure 3(B) represent two linear functions of η in Eq. (16). Due to the monotonicity, the maximum value of the two functions is located at the lower bound of η, which means that the worst constraint violation of g_1 and g_2 occurs at η^L. Thus, the critical point of η is $\eta^c = \eta^L$.

$$\begin{cases} g_1: 1 - \eta \leq 0 \\ g_2: -\frac{\eta}{3} + \frac{1}{3} \leq 0 \end{cases} \tag{16}$$

At $\eta^c = 1.5$, the maximum deviation of θ can be found by comparing the up and down directions from θ^N.

$$\begin{cases} g_1: |\theta^N - (\eta^c)| = \frac{1}{2} \\ g_2: \left|\theta^N - \left(-\frac{\eta^c}{3} + \frac{4}{3}\right)\right| = \frac{1}{6} \end{cases}$$

As shown in Figure 3(C), because 1/6 is less than 1/2, the critical value of θ is 5/6, and the flexibility index with respect to θ is $F = 1/6$. Thus, by centring in the nominal point, the feasible range of θ is [5/6, 7/6]. The yellow region shown in Figure 3(D) is the entire feasible region. This example shows the important role of η^c and the relationship between η^c and F.

3.3. Geometric interpretation for the vertex search method of process parameters

For the case with q process parameters, the hyperrectangle has 2^q vertices. The vertex search method requires to calculate the maximum deviations of θ for 2^q vertex directions. Figure 4(A) illustrates an example with 4 vertex directions. The maximum deviation along each direction is different, and the shortest one indicates a largest inscribed rectangle, i.e., the green region shown in Figure 4(B), which represents the feasible region of the process parameters.

4. Analysis of DFO methods and DFO solvers

We should note that the above proposed algorithm of flexibility index is rigorous for the case that the feasible region is convex. However, since this is a sufficient condition, a rigorous solution may still be obtained for nonconvex problems. Before introducing the solution strategy of the proposed flexibility index model, we discuss the DFO methods and analyze the properties of the DFO solvers in detail.

Based on the search strategies, DFO methods can be grouped into two types: *direct search* methods, which determine the search directions directly from the function evaluation data, and *model-based* methods, which typically use a trust-region framework for selecting new iterations. In addition, DFO methods can be divided into *local search* methods, which start from an initial guess and move within a local trust region, and *global search* methods, which search the entire bounded variable space. However, essentially, neither the local search methods nor the global search methods can guarantee finding the global optima.

Rios and Sahinidis[37] benchmarked the performance of 22 DFO software packages with 502 test problems. They found that a series of TOMLAB MATLAB solvers is more powerful to solve the DFO problems, and BOBYQA (Bound Optimization BY Quadratic Approximation) has superior performance in refining near-optimal solutions. They also pointed out that there is no universal solver that is superior to solve all the problems, and the dimensionality and non-smoothness can increase the complexity of the search process and decrease performance for all the DFO solvers. Gao et al.[42] compared three different DFO methods, namely BOBYQA, MCS and SNOBFIT, and the case studies showed that BOBYQA requires the fewest number of function evaluations. With the development of DFO algorithms, software implementations have been developed; however, most of the software packages are based on MATLAB or C/C++, and only a few solvers have the Python implementations.

Moreover, for the optimization models with a black-box objective function and box constraints, most of the DFO methods are applicable. In particular, the model-based methods, for example, the trust-region based DFO methods, can capture curvature of the objective function well and have better performance[43]. In general, for the DFO solvers, as the size of the problem increases, the chances of obtaining better solutions decreases. Eqs. (13) and (14) show that the flexibility index model is just a univariate DFO model; thus, all the DFO solvers can in principle be used. In this work, a DFO solver, Py-BOBYQA, which is a Python implementation of the BOBYQA Fortran solver by Powell[44], is introduced to solve flexibility index problems. Py-BOBYQA is designed for the optimization models like Eq. (17).

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) \\ &s.t.\ a \le x \le b \end{aligned} \tag{17}$$

Py-BOBYQA is based on the trust-region method, which can find local solutions of nonlinear, nonconvex, least-squares minimization problems (with box constraints), without requiring derivatives of the objective. Py-BOBYQA approximates the function $f(x)$ using a quadratic function, which matches the function value of $f(x)$ at certain interpolation points chosen by the algorithm. The quadratic function is then used in a trust region procedure for updating the decision variables. One interpolation point is changed in each iteration. The trust region radius is cautiously reduced, making sure that the interpolation points span a reasonable range. More detailed description of the algorithm can be found in [45]. It is worth noting that Py-BOBYQA has an optional heuristic method for global search mode. This heuristic method is a multiple restart mechanism, which repeatedly re-initializes Py-BOBYQA from the best point found so far, and a larger trust region radius is used each time. Py-BOBYQA uses the final iterate of a run as the starting point for the restarted run, and the new restart sets up to help to escape from local minima[45]. Thus, as it is a heuristic, there is no guarantee that it will find a global minimum. However, it is likely to escape local minima if there are better values nearby.

In order to test the performance of Py-BOBYQA and compare its local and global search modes, four numerical examples are designed, including one univariate optimization problem and three bivariate optimization problems. All of the models, initializations and results are summarized in the Supporting Information file. The function evaluations and CPU time are also considered. From the comparison results, we can conclude that

1) The optimization results are largely to do with the initial values, regardless of the local or global search mode. A good guess can help to converge to the global optima.
2) The global search mode of Py-BOBYQA has the ability to escape local minima, but the global optimal solutions also cannot guarantee to be found.
3) In general, the global search mode has better performance than the local search mode, but the global search takes many more function evaluations. If the black-box model is too complex, the computational cost of the solution process will be very large.
4) When solving a large-scale black-box optimization problem, before determining to use the local search mode or global search mode, it is necessary to make a tradeoff between the accuracy of the solution and the time cost.
5) If the dimensionality of the model is low, the local search mode may be a better option.

Since the DFO model of flexibility index is a univariate model, theoretically, the local search also can have good performance. Before determining to apply the local or global search mode, it is necessary to test and compare the solutions through some demonstration cases.

5. Solution strategy of the flexibility index model

In this section, three different methods, the vertex enumeration method and two gradient approaximation methods, are proposed to evaluate the flexibility index. For the case with q process parameters, the vertex enumeration method requires to performing optimization over 2^q vertices to determine the smallest scaled deviation. The first gradient approximation method can significantly reduce the number of vertices to be tested and optimized by approximating the gradients of the constraints. The second gradient approximation method assumes that the constraints are monotonic with respect to the process parametrs, which avoids the computational burden of approximating the gradients through optimiztion calculations. However, since the vertex enumeration method is a sufficient condition, it is more rigorous than these two gradient approximation methods.

5.1. Vertex enumeration method

Figure 5 shows the solution strategy of the DFO model of flexibility index based on the vertex enumeration method, which is implemented in Python. If the number of the process parameters is q, a total of 2^q vertex directions will be tested and optmized successively to determine the smallest scaled deviation. Since the critical point of the model paraqmeters, η^c, is obtained, after substituing η^c into the DFO model, the entire solution procedure only includes one loop that enumerates all of the vertex directions.

The detailed algorithm is as follows.

1) An initial value of δ and the vertex direction Δ^k are subsititued into $\theta = \theta^N + \delta \cdot \Delta^k$. The obtained θ and η^c are combined as an input value of the black-box model.
2) Simulate the black-box model. If the simulation fails at the current initialization, the iterative information including the largest feasible solution of δ will be recorded, and go to step 5; otherwise go to step 3;

3) The values of process parameters at the final time are output to evaluate the objective function of the DFO model. If not converged yet, go to step 4; otherwise, Record the candidate flexibility index F^k and go to step 5.
4) The DFO solver maximizes a new value of the scaled deviation δ within the range of δ, go back to step 1;
5) $k = k + 1$; if k is not beyond 2^p, go back to step 1; otherwise, go to step 6;
6) For the obtained candidate values, the smallest one is the final flexibililty index.

5.2. Gradient approximation method

In order to avoid enumerating all the vertices, while generating quickly a good upper bound of the flexibility, two methods are proposed to approximate the gradients of the constraints with respect to the process parameters. Based on this strategy, if there are c constraints, only c vertices need to be calculated, which is commonly much less than 2^q.

For general constraints, the gradient of the constraint g_i with respect to θ_j can be approximated by Eq. (18), where $\max g_i(\theta_j)$ and $\min g_i(\theta_j)$ implies finding the maximum and minimum values of g_i within the bound of θ_j. Figure 6 illustrates the geometric interpretation of Eq. (18).

$$\frac{\partial g_i}{\partial \theta_j} \approx \frac{\max\limits_{\theta_j \in [\theta_j^L, \theta_j^U]} g_i(\theta_j) - \min\limits_{\theta_j \in [\theta_j^L, \theta_j^U]} g_i(\theta_j)}{\theta_j^{max} - \theta_j^{min}} \tag{18}$$

Assuming that g_i is monotonic with respect to θ_j, the gradient can be approximated by only evaluating the function values of the constraint at the lower and upper bound.

$$\frac{\partial g_i}{\partial \theta_j} \approx \frac{g_i(\theta_j^L) - g_i(\theta_j^U)}{\theta_j^L - \theta_j^U} \tag{19}$$

For each constraint g_i, q gradients need to be approximated, and the corresponding q signs can form a vertex direction, as shown in Eq. (20). Since the problem has c constrains, a total of c vertex directions needs to be calculated.

$$\text{vertex direction} = [\text{sign}\left(\frac{\partial g_i}{\partial \theta_1}\right), \ldots, \text{sign}\left(\frac{\partial g_i}{\partial \theta_q}\right)] \tag{20}$$

Instead of evaluating all the vertex directions, the proposed gradient approximation methods can approximate the flexibility index at a specific vertex direction. This vertex direction is given by the sign of the derivatives of the constraints with respect to the process parameters. A negative value of the derivative suggests the evaluation of the feasibility in the lower bound direction, whereas a positive value suggests the upper bound direction. This method will be very clear for a monotonic constraint. As shown in Table 1, for a monotonic constraint with two process parameters, there are four vertex directions. We can determine them by computing the signs of the derivatives. Each combination of signs corresponds to a vertex direction. For nonmonotonic cases, we can apply Eq. (18) or the simplified Eq. (19) to approximate the derivative with respect to each process parameter. The final derivative signs in Eq. (20) correspond to a vertex direction that indicates a potential critical vertex direction.

If the number of the constraints is c, the above three methods to evaluate the upper bound of the flexibility index can be compared as follows:

1) The vertex enumeration method requires to solve $\underline{2^q}$ DFO problems;
2) The gradient approximation method requires to solve $\underline{2cq + c}$ DFO problems;
3) The gradient approximation method with assumption of monotonic constraints requires to solve $\underline{c}$ DFO problems;

The computational costs of three methods are different. The gradient approximation method has higher efficiency because of solving less DFO problems; however, the vertex enumeration method can obtain more reliable upper bound of the flexibility index.

6. Example: A reaction model

In this section, a reaction model with 8 process parameters and 12 model parameters is studied. For this example, gPROMS and Python are combined to handle the black-box model and the flexibility index problem. The entire model is treated as a black box, which is simulated by gPROMS ModelBuilder 6.0.4. The flexibility index model is developed in Python 3.7.1, and

Py-BOBYQA 1.2 is chosen as the DFO solver. The detailed connecting way between gPROMS and Python is described in the Supporting Information file. The entire model is implemented on a 64-bit Windows 10 desktop system with a 3.60 GHz Intel i7-7700 processor and 16 GB of RAM. The specifications of the process and model parameters are listed in Table 2 and Table 3, and θ_9 is one of the products. This reaction model has two quality constraints; that is, the final concentration of θ_3 and θ_4 should be less then 250 ppm. As the values of the constraints may be very small, in order to ensure the accuracy of the calculation, *coef* is introduced to change the order of magnitude.

$$\begin{gathered} 0 \leq g_1 \leq coef * 250 * 10^{-6} \\ 0 \leq g_2 \leq coef * 250 * 10^{-6} \end{gathered} \tag{21}$$

$$\begin{aligned} g_1 &= coef * \frac{\text{final\_}\theta_3}{\text{final\_}\theta_3 + \text{final\_}\theta_4 + \text{final\_}\theta_9} \\ g_2 &= coef * \frac{\text{final\_}\theta_4}{\text{final\_}\theta_3 + \text{final\_}\theta_4 + \text{final\_}\theta_9} \end{aligned}$$

In order to solve the flexibility index, the worst violation of g_1 and g_2 for each model parameter should be calculated first. The formulations are shown in Eq. (22).

$$\begin{aligned} u_1^i &= \max_{\eta_i \in [\eta_i^L, \eta_i^U]} g_1(\theta^N, \eta_i) \\ u_2^i &= \max_{\eta_i \in [\eta_i^L, \eta_i^U]} g_2(\theta^N, \eta_i) \end{aligned} \tag{22}$$

where 8 process parameters are specified as their nominal values, θ^N; for the calculation of each model parameter η_i, the other 11 model parameters are specified as the corresponding mean values shown in Table 3; thus, the maximum values of g_1 and g_2 can be calculated within the feasible range of η_i. For the accuracy of optimization calculation, *coef* is set to 10^4 in g_1 and g_2. Thus, the range of the quality constraints changes to [0, 2.5]. In addition, in order to capture the maximum values as far as possible, the global search mode of Py-BOBYQA is chosen. The worst values of g_1 and g_2 are summarized in Table 4. Two blue columns show the maximum values of two constraints with respect to each model parameter. Obviously, the worst constraint violations always occur in g_2, because $u_2^i > u_1^i, i = 1, \ldots, 12$. Therefore, the critical point of the model parameters, η^c, just corresponds to the solutions of maximizing g_2, i.e.,

$$\eta^c = \begin{bmatrix} 1062.57337, 1034.35395, 1015.29997, 0.13496, \\ 0.0004459, 11359.69258, 264.79870, 2.28422, \\ 1.19793\mathrm{e}-09, 1.16878\mathrm{e}-09, 2.93385\mathrm{e}-10, 6.66969\mathrm{e}-10 \end{bmatrix} \tag{23}$$

In addition, in order to compare the reliability of the global search mode, the local search mode is also tested, and the results are summarized in Table 5. We can find that all of the maximum values of g_1 and g_2 in Table 4 are larger than the corresponding values in Table 5, which indicates that the global search is more suitable than the local search when calculating the worst constraint violations. Thus, Eq. (23) can be defined as the critical point of the model parameters. After substituting η^c into the flexibility index model, we can focus on dealing with the issue of process parameters.

1) ***Vertex enumeration method***

Since there are 8 process parameters, the vertex enumeration method requires to calculate $2^8 = 256$ vertex directions. According to Eq. (14), the penalty coefficient M is set to 1000, and the range of the constraints is [0, 2.5]. The DFO model of flexibility index for this case is

$$\begin{aligned} &F^k = \min -\delta + 1000 * \sum_j^J \begin{pmatrix} \max(0, -g_1) + \max(0, g_1 - 2.5) + \\ \max(0, -g_2) + \max(0, g_2 - 2.5) \end{pmatrix} \\ &s.t.\ \ 0 \le \delta \le 1 \end{aligned} \tag{24}$$

Each vertex direction corresponds a kind of deviation Δ^k from θ^N, and the process parameters are calculated by Eq. (25). Then, θ and η^c can be used to simulate the black-box model. The final values of the process parameters are output to evaluate the objective function of Eq. (24).

$$\theta = \theta^N + \delta \cdot \Delta^k \tag{25}$$

Before solving the flexibility index, it is necessary to pretest the model to compare some information. For example, the local search and global search modes need to be compared in advance to determine which one is applied. Table 6 lists the comparison results at five different vertices. It shows that the obtained flexibility indices are the same for the local and global search modes; however, the computational efforts are very different as the local search mode requires much less time. Since the DFO model of flexibility index has only one variable, both search methods have the similar performance; thus, the local search method is chosen to deal with the vertex enumeration process. In addition, for simplification, we assign the initial value of δ as 0.1 for all of the vertices. The identical results of the local and global search modes in Table 6 indicate that this initial value is acceptable.

By enumerating all of the vertex directions, 256 candidate flexibility indices can be obtained, and the partial results are listed in Table 7. The final flexibility index is the smallest value, i.e., 0.54565, which is located at the #130 direction. Note that the DFO solver cannot converge at five vertices, i.e., #52, #55, #61, #118 and #124, and the reason is gPROMS simulation cannot be executed at some initialization. The recorded largest feasible δ is 0.7 in the iterative procedure. Since 0.7 is greater than 0.54565, these vertices that cannot be converged have no effect on the final result.

For this case, at most of the vertices, the result is equal to 1. If extracting all of the converged vertex directions that the result is not equal to 1, we can find that the critical direction of θ_3 and θ_7 remain the same, i.e., "+" and "-", respectively, which indicates that these two components have a large impact on the flexibility index. The final results are summarized in Table 8. All the quality constraints are satisfied. The feasible range of the process parameters are listed in Table 9.

2) ***Gradient approximation method***

The first gradient approximation method has to solve the maximization and minimization problems of g_1 and g_2 through Eq. (18). Table 10 lists the optimization results. For constraint g_1 and g_2, the signs of the gradients can form a vertex direction. For example, the approximated gradients of g_1 and g_2 are #234 and #142 vertices, respectively; thus, we only need to calculate these two vertices to evaluate the flexibility index. The results show that $F = 0.546$.

Similarly, assuming that g_1 and g_2 are monotonic with respect to each process parameter, the second gradient approximation method only requires the function evaluations of g_1 and g_2 at the lower and upper bounds. Table 10 shows the signs of the approximated gradients and two corresponding vertices, i.e., #252 and #206, and the final flexibility index F is also 0.546. Moreover, Table 11 summarizes the times required for all the parts in the proposed method. Choosing the global search to calculate the critical point of the model parameters and choosing the local search mode to execute the vertex calculations, the total wall time of the vertex enumeration method is 13,671 seconds, while applying the gradient approximation methods, the computational expense can be reduced to 10,607 seconds. In summary, two gradient approximation methods have higher efficiency because of solving fewer DFO problems, and the vertex enumeration method can obtain more reliable upper bound of the flexibility index. Therefore, if the user wants to obtain a better solution, the vertex enumeration method can be chosen; if the user is more concerned with computational efficiency, either of the two gradient approximation methods can be used.

7. Conclusions

In this study, a novel solution strategy is proposed to find a reliable upper bound of the flexibility index for black box models. Univariate DFO calculations for flexibility index are determined of the scaled deviations for different vertices. The input and output information of the black box model is integrated with an external DFO solver based on trust-region methods. After finding the critical point of the model parameters by calculating the worst constraint violations, a reliable upper bound of the flexibility index can be obtained either by enumerating all the vertex directions related to the process parameters, or by approximating the gradients of the constraints. Although the proposed method cannot guarantee to find the optimum solution, i.e., the exact result of flexibility index, the result provides a reliable upper bound of flexibility index as has been shown by the numerical examples.

In summary, the proposed method provides a new approach to deal with flexibility index problems that involve general process models. The contributions of this work can be summarized as follows.

1) The process model is implemented in a black box that can be directly executed by its original commercial simulator, and we can avoid calculating derivatives because of the DFO model of flexibility index. Thus, the proposed method can deal with general types of process models, regardless of convex or nonconvex, steady-state or dynamic models.
2) In the pharmaceutical industry, complex differential equations are very common, and they can result in huge nonlinear optimization problems if discretized by finite differences or by orthogonal collocation. DFO based on black box models is a much simpler way to handle complex process models.
3) Viewing process models as black boxes and creating the interface between the black-box models and Python provides a new way to deal with the flexibility analysis problem of complex process models.

Nomenclature

d	Design variables
θ	Process parameters
θ^N	Nominal value of process parameters
θ^L, θ^U	Lower and upper bound of process parameters
$\Delta\theta^-, \Delta\theta^+$	Deviations of process parameters in the negative and positive directions
η	Model parameters
η^c	Critical point of model parameters
η^L, η^U	Lower and upper bound of model parameters
χ	Flexibility function
T_p	Feasible range of process parameters
T_m	Feasible range of model parameters
g	Quality constraints

A^L, A^U	Lower and upper bound of quality constraints
M	Penalty coefficient
F	Flexibility index
F^k	Candidate flexibility index in the k[th] vertex direction

Acknowledgements

The authors gratefully acknowledge the financial support from Eli Lilly and Company and the Center for Advanced Process Decision-making (CAPD) from Carnegie Mellon University.

Literature Cited

1. Halemane KP, Grossmann IE. Optimal process design under uncertainty. *AIChE J*. 1983;29(3):425-433.
2. Sirdeshpande AR, Ierapetritou MG, Andrecovich MJ, Naumovitz JP. Process synthesis optimization and flexibility evaluation of air separation cycles. *AIChE J*. 2005;51(4):1190-1200.
3. Zheng K, Lou HH, Wang J, Cheng F. A method for flexible heat exchanger network design under severe operation uncertainty. *Chem Eng Technol*. 2013;36(5):757-765.
4. Li J, Du J, Zhao Z, Yao P. Efficient method for flexibility analysis of large-scale nonconvex heat exchanger networks. *Ind Eng Chem Res*. 2015;54(43):10757-10767.
5. Wang H, Mastragostino R, Swartz CLE. Flexibility analysis of process supply chain networks. *Comput Chem Eng*. 2016;84:409-421.
6. Grossmann IE, Calfa BA, Garcia-Herreros P. Evolution of concepts and models for quantifying resiliency and flexibility of chemical processes. *Comput Chem Eng*. 2014;70(6): 22-34.
7. Zhang Q, Lima R, Grossmann IE. On the Relation Between Flexibility Analysis and Robust Optimization for Linear Systems. *AIChE J*. 2016;62(9):3109-3123.
8. Zhao F, Chen X. Analytical and triangular solutions to operational flexibility analysis using quantifier elimination. *AIChE J*. 2018;64(11):3894-3911.
9. Zhao F, Zheng C, Zhang S, Zhu L, Chen X. Quantification of process flexibility via space projection. *AIChE J*. 2019; 65(10), e16706.

10. Ostrovsky GM, Datskov IV, Achenie LEK, Volin YuM. Process uncertainty: case of insufficient process data at the operation Stage. *AIChE J*. 2003;49(5):1216-1232.

11. Rooney WC, Biegler LT. Optimal process design with model parameter uncertainty and process variability. *AIChE J*. 2003;49(2):438-449.

12. Ochoa MP, Grossmann IE. Novel MINLP formulations for flexibility analysis for measured and unmeasured uncertain parameters. *Comput Chem Eng*. 2020;135:106727.

13. Swaney RE, Grossmann IE. An index for operational flexibility in chemical process design. Part I: Formulation and theory. *AIChE J*. 1985;31(4):621-630.

14. Swaney RE, Grossmann IE. An index for operational flexibility in chemical process design. Part II: Computational algorithms. *AIChE J*. 1985;31(4):631-641

15. Grossmann IE, Floudas CA. Active constraint strategy for flexibility analysis in chemical processes. *Comput Chem Eng.* 1987;11(6): 675-693.

16. Rogers A, Ierapetritou MG. Feasibility and flexibility analysis of black-box processes Part 1: Surrogate-based feasibility analysis. *Chem Eng Sci*. 2015;137:986-1004.

17. Rogers A, Ierapetritou MG. Feasibility and flexibility analysis of black-box processes part 2: Surrogate-based flexibility analysis. *Chem Eng Sci*. 2015;137:1005-1013.

18. Banerjee I, Ierapetritou MG. Design optimization under parameter uncertainty for general black-box models. *Ind Eng Chem Res*. 2002;41(26):6687-6697.

19. Boukouvala F, Ierapetritou MG. Feasibility analysis of black-box processes using an adaptive sampling Kriging-based method. *Comput Chem Eng*. 2012;36(36):358-368.

20. Laky D, Xu S, Rodriguez J, Vaidyaraman S, García-Muñoz S, Laird C. An optimization-based framework to define the probabilistic design space of pharmaceutical processes with model uncertainty. *Process*. 2019;7:96-116.

21. García-Muñoz S, Luciani C, Vaidyaraman S, Seibert K. Definition of design space using mechanistic models and geometric projections of probability maps. *Org Process Res Dev*. 2015;54:5128-5138.

22. Kucherenko S, Giamalakis D, Shah N, García-Muñoz S. Computationally efficient identification of probabilistic design spaces through application of metamodeling and adaptive sampling. *Comput Chem Eng*. 2020;132:106608.

23. Zheng C, Zhao F, Zhu L, Chen X. Operational flexibility analysis of high-dimensional systems via cylindrical algebraic decomposition. *Ind Eng Chem Res.* 2020;59(10): 4670-4687.

24. Dimitriadis VD, Pistikopoulos EN. Flexibility analysis of dynamic systems. *Ind Eng Chem Res*. 1995;34(12):4451-4462.

25. Adi VSK, Chang CT. A mathematical programming formulation for temporal flexibility analysis. *Comput Chem Eng*. 2013;57:151-158.

26. Mohideen MJ, Perkins JD, Pistikopoulos EN. Optimal design of dynamic systems under uncertainty. *AIChE J*. 1996;42(8):2251-2272.

27. Boukouvala F, Ierapetritou MG. Surrogate-based optimization of expensive flowsheet modeling for continuous pharmaceutical manufacturing. *J Pharm Innov*. 2013;8(2):131-145.

28. Boukouvala F, Misener R, Floudas CA. Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO. *Eur J Oper Res*. 2016;252(3):701-727.

29. Boukouvala F, Hasan MMF, Floudas CA. Global optimization of general constrained grey-box models: new method and its application to constrained PDEs for pressure swing adsorption. *J Global Optim*. 2017;67(1-2):3-42.

30. Peaceman DW. *Fundamentals of numerical reservoir simulation*. New York: Elsevier Scientific Publishing Co. 1977.

31. Shiehnejadhesar A, Schulze K, Scharler R, Obernberger I. A new innovative CFD-based optimization method for biomass combustion plants. *Biomass Bioenergy*. 2013;53:48-53.

32. Nagy J, Horvath A, Jordan C, Harasek M. CFD simulation of a high temperature furnace. 8th International conference on CFD in Oil & Gas, Metallurgical and Process Industries. Trondheim, Norway, 2011.

33. Zhao Z, Meza JC, Van Hove M. Using pattern search methods for surface structure determination of nanomaterials. *J Phys Condens Matter*. 2006;18:8693-8706.

34. Marsden AL, Feinstein JA, Taylor CA. A computational framework for derivative-free optimization of cardiovascular geometries. *Comput Methods Appl Mech Eng*. 2008;197: 1890-1905.

35. Moré J, Wild S. Benchmarking derivative-free optimization algorithms. *SIAM J Optim*. 2009;20:172-191.

36. Hayes RE, Bertrand FH, Audet C, Kolaczkowski ST. Catalytic combustion kinetics: using a direct search algorithm to evaluate kinetic parameters from light-off curves. *Can J Chem Eng*. 2003;81:1192-1199.

37. Rios LM, Sahinidis NV. Derivative-free optimization: a review of algorithms and comparison of software implementations. *J Glob Optim*. 2013;56(3):1247-1293.

38. Larson J, Menickelly M, Wild SM. Derivative-free optimization methods. *Acta Numerica*, 2019;28:287-404.

39. Zangwill WI. Non-linear programming via penalty functions. *Manag Sci*. 1967;13(5):344-358.

40. Fletcher R. An exact penalty function for nonlinear programming with inequalities. *Math Program.* 1973;5(1):129-150.

41. Fiacco AV. Penalty methods for mathematical programming in E^n with general constraint sets. *J Optim Theory Appl*. 1970;6(3):252–268.

42. Gao H, Waechter A, Konstantinov IA, Arturo SG, Broadbelt LJ. Application and comparison of derivative-free optimization algorithms to control and optimize free radical polymerization simulated using the kinetic Monte Carlo method. *Comput Chem Eng*. 2018;108:268-275.

43. Garmanjani R, Júdice D, Vicente LN. Trust-region methods without using derivatives: worst case complexity and the nonsmooth case. *SIAM J Optim*. 2016;26(4):1987–2011.

44. Powell MJD. The BOBYQA Algorithm for Bound Constrained Optimization Without Derivatives. Technical report, Department of Applied Mathematics and Theoretical Physics, University of Cambridge. 2009.

45. Cartis C, Roberts L, Sheridan-Methven O, Escaping local minima with derivative-free methods: a numerical investigation. Technical report, University of Oxford, 2018.

List of Table Captions:

Table 1. Derivative signs and vertex directions for a monotonic constraint.

Table 2. Specifications of process parameters.

Table 3. Specifications of model parameters.

Table 4. Worst values of g_1 and g_2 with respect to each model parameter (global search).

Table 5. Worst values of g_1 and g_2 with respect to each model parameter (local search).

Table 6. Comparison of the local and global search modes at five vertices.

Table 7. Partial results of vertex enumeration method.

Table 8. Final results of the reaction example.

Table 9. Feasible range of process parameters (F = 0.54565)

Table 10. Results of two gradient approximation methods.

Table 11. Summary of computational expense.

Table 1. Derivative signs and vertex directions for a monotonic constraint.

No.	Derivative sign		Vertex direction
1	$\frac{\partial g_1}{\partial \theta_1} < 0,$	$\frac{\partial g_1}{\partial \theta_2} > 0$	$\theta_1^{LB}, \theta_2^{UB}$
2	$\frac{\partial g_1}{\partial \theta_1} < 0,$	$\frac{\partial g_1}{\partial \theta_2} < 0$	$\theta_1^{LB}, \theta_2^{LB}$
3	$\frac{\partial g_1}{\partial \theta_1} > 0,$	$\frac{\partial g_1}{\partial \theta_2} > 0$	$\theta_1^{UB}, \theta_2^{UB}$
4	$\frac{\partial g_1}{\partial \theta_1} > 0,$	$\frac{\partial g_1}{\partial \theta_2} < 0$	$\theta_1^{UB}, \theta_2^{LB}$

Table 2. Specifications of process parameters.

No.	Process parameter	Nominal value [kg]	Range
1	θ_1	1.0	[0.94, 1.06]
2	θ_2	5.2	[4.888, 5.512]
3	θ_3	253	[237.82, 268.18]
4	θ_4	2.5	[0, 10]
5	θ_5	2254	[1983.52, 2524.48]
6	θ_6	185.0	[173.9, 196.1]
7	θ_7	172.0	[161.68, 182.32]
8	θ_8	16.3	[14.344, 18.256]

Table 3. Specifications of model parameters.

No.	Model parameter	Mean value	Range
1	η_1	1000	[910, 1090]
2	η_2	1000	[910, 1090]
3	η_3	1000	[910, 1090]
4	η_4	0.14	[0.1274, 0.1526]
5	η_5	4.9e-4	[4.459e-4, 5.341e-4]
6	η_6	12000	[10920, 13080]
7	η_7	250	[227.5, 272.5]
8	η_8	2.3	[2.093, 2.507]
9	η_9	1.199e-9	[1.09109e-9, 1.30691e-9]
10	η_{10}	1.102e-9	[1.00282e-9, 1.20118e-9]
11	η_{11}	3.224e-10	[2.93384e-10, 3.51416e10]
12	η_{12}	6.839e-10	[6.22349e-10, 7.45451e-10]

Table 4. Worst values of g_1 and g_2 with respect to each model parameter (global search).

No.	Maximum of g_1 u_1^i	Solution of η_i	Maximum of g_2 u_2^i	Solution of η_i	Function evaluations of	
					g_1	g_2
1	2.48858 e-22	1004.32217	1.55693 e-17	1062.57337	568	382
2	2.37935 e-22	1010.62212	1.45094 e-17	1034.35395	542	569
3	2.62789 e-22	1082.42786	1.53990 e-17	1015.29997	656	579
4	3.26773 e-22	0.15167	1.62858 e-17	0.13496	571	541
5	2.62380 e-22	0.0005341	1.47474 e-16	0.0004459	502	446
6	2.59350 e-22	10968.41075	1.48153 e-17	11359.69258	628	595
7	2.60830 e-22	266.58804	1.53343 e-17	264.79870	668	554
8	2.73007 e-22	2.11658	1.65911 e-17	2.28422	659	641
9	2.41287 e-22	1.30691 e-09	1.46375 e-17	1.19793 e-09	596	614
10	2.64698 e-22	1.05507 e-09	1.86188 e-17	1.16878 e-09	633	579
11	2.17501 e-22	3.44730 e-10	1.83921 e-17	2.93385 e-10	572	506
12	1.85513 e-22	6.83923 e-10	1.62106 e-17	6.66969 e-10	101	535

Table 5. Worst values of g_1 and g_2 with respect to each model parameter (local search).

No.	Maximum of g_1 u_1^i	Solution of η_i	Maximum of g_2 u_2^i	Solution of η_i	Function evaluations of	
					g_1	g_2
1	2.48858 e-22	1004.32217	6.81711 e-18	980.41574	19	22
2	1.89046 e-22	1018.0	6.24371 e-18	1021.01874	21	20
3	1.47025 e-22	1021.80780	1.12013 e-17	1015.3	22	18
4	1.42346 e-22	0.13978	1.40561 e-17	0.13496	20	22
5	1.66158 e-22	0.0004961	1.47352 e-16	0.0004459	25	21
6	1.58130 e-22	11784.054	1.00410 e-17	12056.44450	22	24
7	1.72443 e-22	245.87533	1.40846 e-17	245.5	22	19
8	2.26167 e-22	2.25939	7.49379 e-18	2.38280	25	21
9	1.66437 e-22	1.19684 e-09	7.38624 e-18	1.19853 e-09	20	21
10	1.70733 e-22	1.07345 e-09	6.17231 e-18	1.11904 e-09	21	22
11	1.43325 e-22	3.28246 e-10	1.25757 e-17	3.16597 e-10	21	21
12	1.85513 e-22	6.83923 e-10	8.69789 e-18	6.97010 e-10	22	21

Table 6. Comparison of the local and global search modes at five vertices.

Vertex #	Flexibility index		Function evaluations		CPU time (sec)		Wall time (sec)	
	local	global	local	global	local	global	local	global
2	0.546	0.546	28	254	0.063	0.547	24.906	223.741
18	0.768	0.768	27	193	0.047	0.641	24.189	169.390
100	1.0	1.0	10	39	0.047	0.125	8.510	31.762
150	0.768	0.768	25	647	0.031	1.688	22.031	574.107
200	1.0	1.0	10	39	0.031	0.063	8.510	31.698

Table 7. Partial results of vertex enumeration method.

Index of the vertex directions	Results
0	1.0
1	1.0
2	0.546
3	0.546
…	…
17	1.0
18	0.768
19	0.768
…	…
124	0.7
125	1.0
126	1.0
127	1.0
128	1.0
129	1.0
130	*0.545649974002829*
131	0.546
132	1.0
133	1.0
134	0.545649974002854
135	0.546
136	1.0
137	1.0
138	0.546
139	0.546
140	1.0
…	…

Table 8. Final results of the reaction example.

	Result
Flexibility index	0.54565
Two quality constraints	[1.83459e-17, 1.60134]
Critical vertex direction	[-1, 1, 1, 1, 1, 1, -1, 1] (130th vertex direction)
Critical process parameters	[0.96726, 5.37024, 261.28297, 6.59237, 2401.5874, 191.05671, 166.36889, 17.36729]
Critical model parameters	[1062.57337, 1034.35395, 1015.29997, 0.13496, 0.0004459, 11359.69258, 264.79870, 2.28422, 1.19793e-09, 1.16878e-09, 2.93385e-10, 6.66969e-10]
CPU time (sec)	34.67
Total time (sec)	13671.08 (3.8 hr)

Table 9. Feasible range of process parameters (F = 0.54565)

No.	Process parameter	Feasible range	
1	θ_1	0.96726*	1.03274
2	θ_2	5.02976	5.37024*
3	θ_3	244.71703	261.28297*
4	θ_4	1.13588	6.59237*
5	θ_5	2106.41260	2401.58740*
6	θ_6	178.94329	191.05671*
7	θ_7	166.36889*	177.63111
8	θ_8	15.23271	17.36729*

* Critical bound for the process parameter

Table 10. Results of two gradient approximation methods.

Gradient approximation method	Constraint	Signs of the approximated gradients for process parameters	Corresponding index of vertex direction	Candidate flexibility index
1	g_1	[-1, -1, -1, 1, -1, 1, -1, 1]	*#234*	1
	g_2	[-1, 1, 1, 1, -1, -1, -1, 1]	*#142*	0.546
2	g_1	[-1, -1, -1, -1, -1, -1, 1, 1]	*#252*	1
	g_2	[-1, -1, 1, 1, -1, -1, -1, 1]	*#206*	0.546

1: Gradient approximation with general constraints

2: Gradient approximation with monotonic constraints

Table 11. Summary of computational expense.

Type	CPU time (sec)	Wall time (sec)
1: Calculation of the critical point of the model parameters (local search)	1.55	414.64
2: Calculation of the critical point of the model parameters (global search)	24.28	10560.27
3: Vertex enumeration method (local search)	10.39	3110.81
4: Gradient approximation method (local search)	1.77	581.26
5: Gradient approximation method with monotonic assumptions (local search)	0.095	46.75
Total time of Method 1 (2+3)	***34.67***	***13671.08***
Total time of Method 2 (2+4)	***26.05***	***11141.53***
Total time of Method 3 (2+5)	***24.38***	***10607.02***

List of Figure Captions:

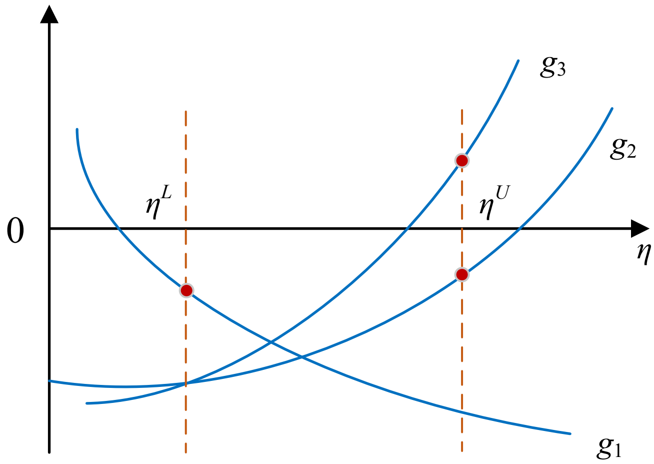

Figure 1. Worst value analysis of the monotonic constraints.

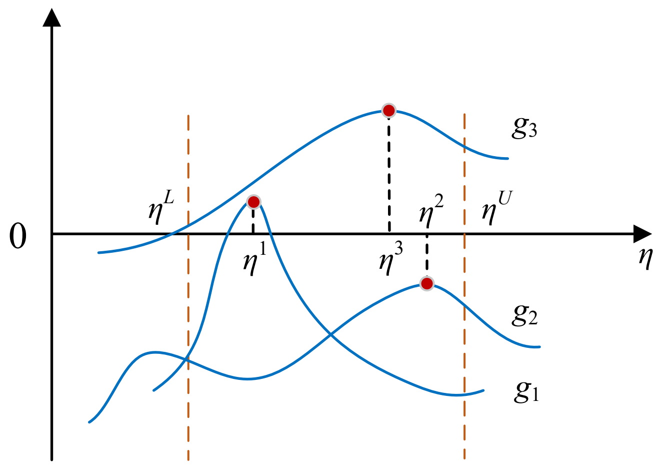

Figure 2. Worst value analysis of the non-monotonic constraints.

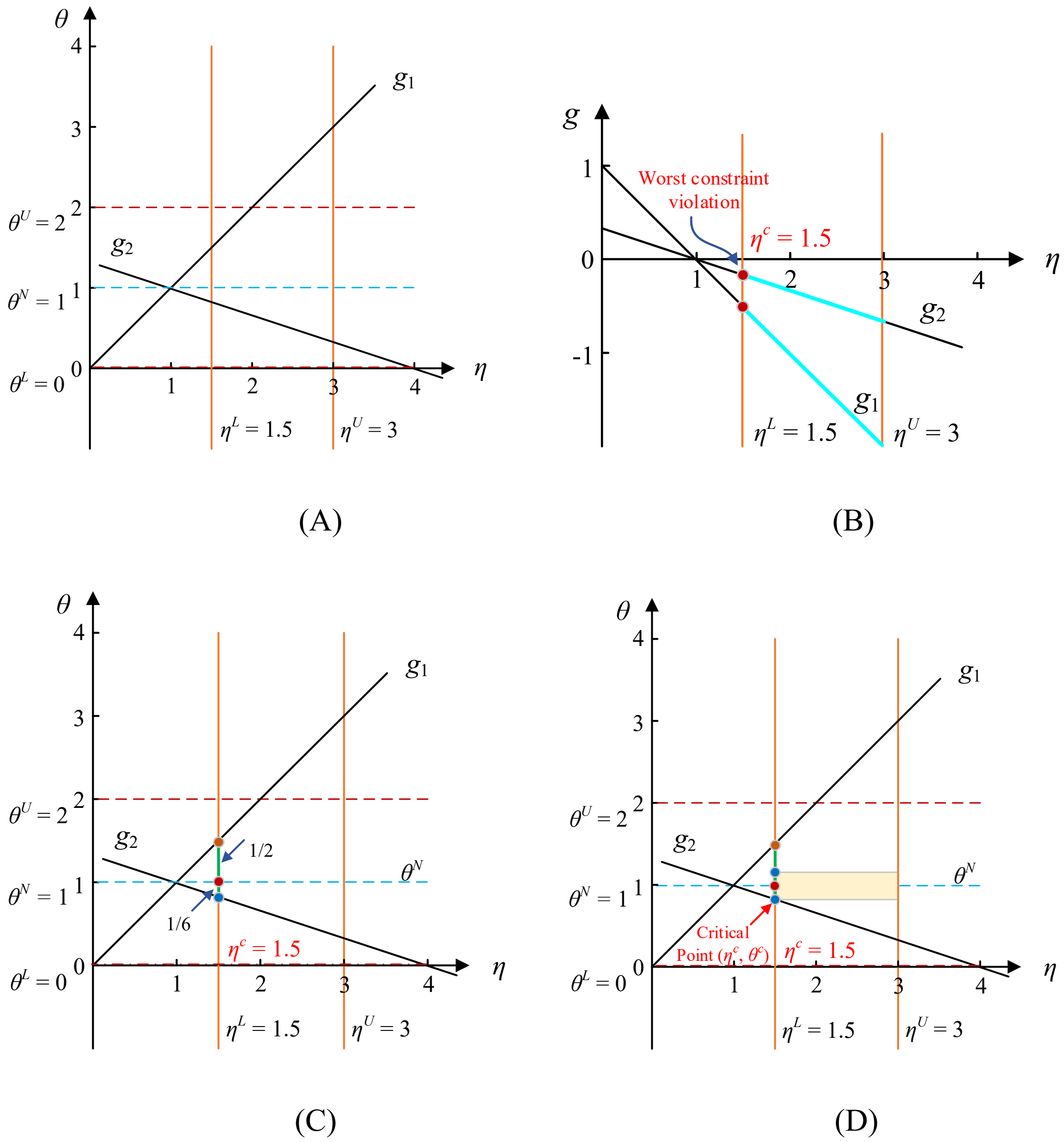

Figure 3. Geometric interpretation of the critical point.

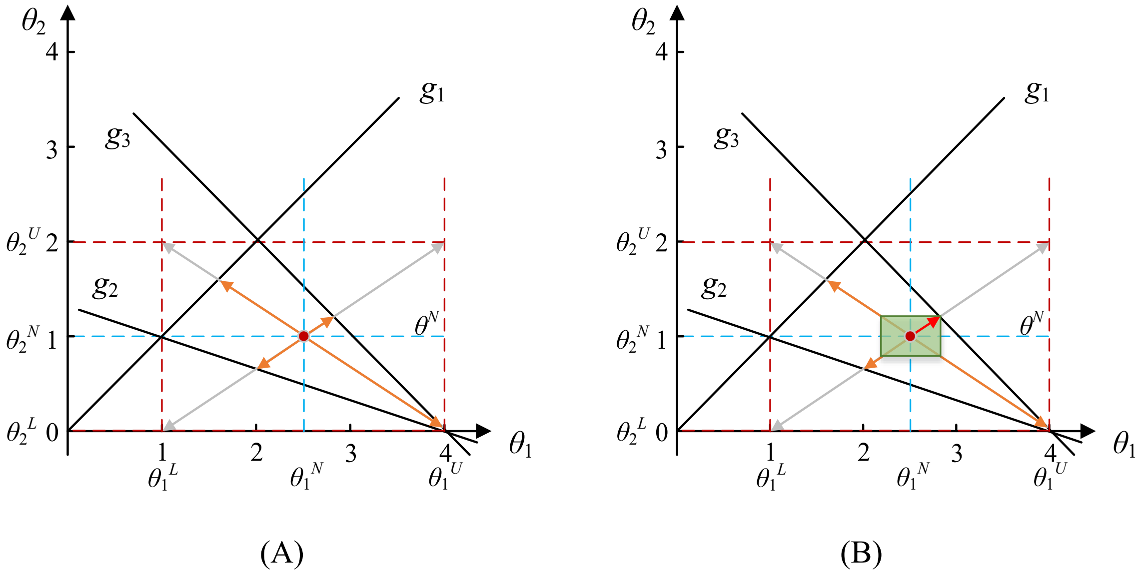

Figure 4. Geometric interpretation of the vertex search method.

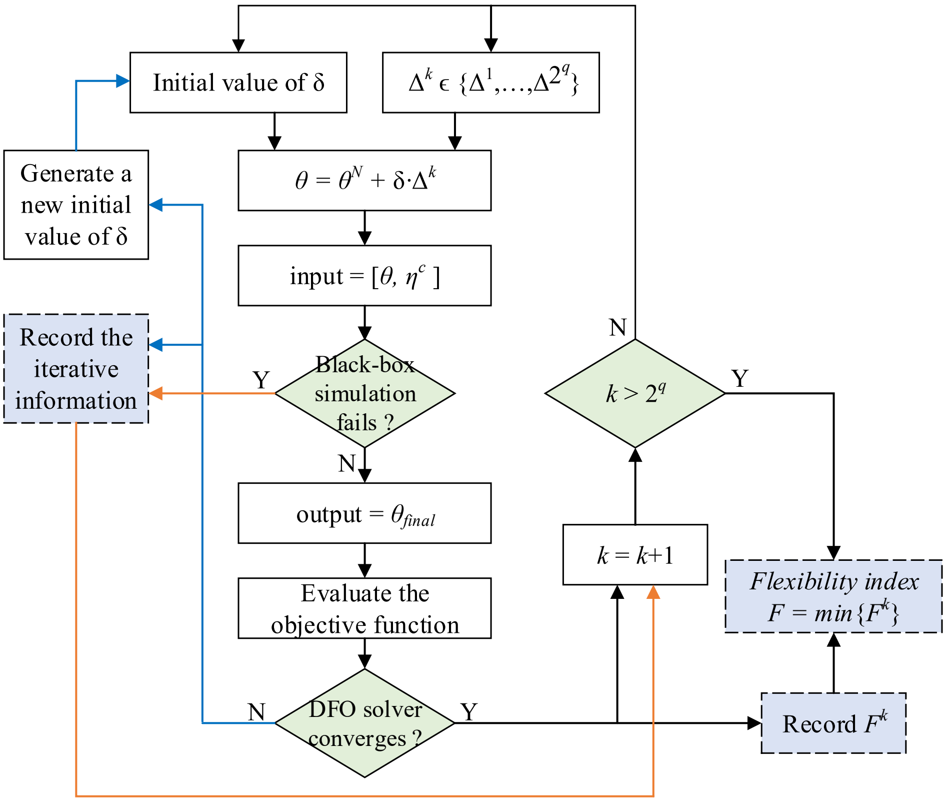

Figure 5. Solution framework of the flexibility index model based on vertex enumeration.

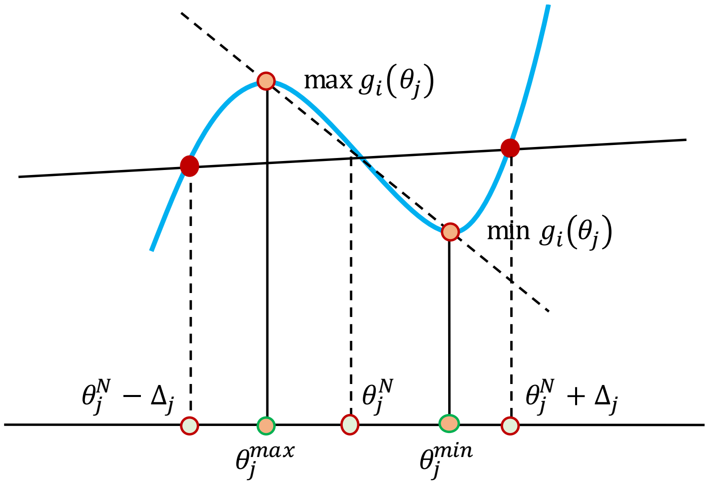

Figure 6. Gradient approximation.