Robust simulation and design using semi-infinite programs with implicit functions

Matthew D. Stuber and Paul I. Barton*

Department of Chemical Engineering,
Massachusetts Institute of Technology,
Cambridge, MA, USA
Email: stuber@mit.edu
Email: pib@mit.edu
*Corresponding author

Abstract: A method is presented for guaranteeing robust steady-state operation of chemical processes using a model-based approach, taking into account uncertainty in the model parameters and disturbances in the process inputs. Intractable constrained max–min optimisation formulations have been proposed for this problem in the past. A new approach is presented in which the equality constraints (process model equations) are solved numerically for the process variables as implicit functions of the uncertain parameters and controls. The problem is then formulated as a semi-infinite program (SIP) constrained only by the performance specifications as semi-infinite inequality constraints. A rigorous, finite ε-optimal convergent algorithm for solving such SIPs is proposed, making no assumptions on convexity, which makes use of the novel developments of parametric interval-Newton methods for bounding implicit functions, and novel developments in McCormick relaxations of algorithms.

Keywords: interval analysis; SIP; semi-infinite optimisation; global optimisation; McCormick relaxation; robust simulation; design under uncertainty.

Reference to this paper should be made as follows: Stuber, M.D. and Barton, P.I. (2011) 'Robust simulation and design using semi-infinite programs with implicit functions', *Int. J. Reliability and Safety*, Vol. 5, Nos. 3/4, pp.378–397.

Biographical notes: Matthew D. Stuber is currently working on his PhD in Chemical Engineering at the Massachusetts Institute of Technology (MIT). He received his BChE from the University of Minnesota, Institute of Technology in Minneapolis, Minnesota. His research interests are in theory and algorithms for nonconvex optimisation, interval analysis and nonlinear equation solving.

Paul I. Barton is the Lammot du Pont Professor of Chemical Engineering at MIT, where he has been since 1992. He received his MEng and PhD in Chemical Engineering from Imperial College, London, in 1988 and 1992. His research interests involve modelling, simulation and optimisation of dynamic systems, and global optimisation theory and algorithms. His group currently focuses on applications in energy systems engineering, continuous pharmaceutical manufacturing and nano-scale systems engineering. In 2004 he was awarded the Outstanding Young Researcher Award by AIChE's CAST Division and in 2008 the Indo-American Frontiers of Engineering Award from the National Academy of Engineering.

This paper is a revised and expanded version of a paper entitled 'Robust simulation and design using parametric interval methods' presented at the '4th International Workshop on Reliable Engineering Computing (REC2010)', Singapore, 3–5 March 2010.

1 Introduction

The task of a process design engineer is to design a process system that will meet all predetermined performance specifications given limited information and resources. Imprecise data and lack of complete environmental information, among other sources, introduce various uncertainties that must be accounted for in the design. In applications such as subsea oil and gas production, increasingly extreme environments and costs make building physical pilot plant systems implausible. In this case, a model-based approach is more plausible. The first question a design engineer must address becomes: 'Given a process model, and taking into account uncertainty in the model and disturbances to the inputs of the system, do there exist control settings such that, at steady state, the physical system will always meet performance and/or safety specifications?' The primary goal is thus to be able to give robustness guarantees for the performance of physical process systems, at the design stage, using a model-based approach.

1.1 Problem formulation

The robustness problem can be formulated as the following constrained max–min optimisation problem (Halemane and Grossmann, 1983):

$$\begin{aligned} \eta^* = & \max_{(\mathbf{d},\mathbf{p})\in D\times P,\ \eta\in\mathbb{R}} \eta \\ \text{s.t. } \eta \le & \min_{\mathbf{u}\in U,\ \mathbf{x}\in X} g(\mathbf{x},\mathbf{u},\mathbf{d},\mathbf{p}) \\ & \text{s.t. } \mathbf{h}(\mathbf{x},\mathbf{u},\mathbf{d},\mathbf{p}) = \mathbf{0} \end{aligned} \tag{1}$$

where $\mathbf{h}$: $X \times U \times D \times P \rightarrow \mathbb{R}^{n_x}$ and g: $X \times U \times D \times P \rightarrow \mathbb{R}$ are the steady-state model equations and performance specification, respectively, with $X \subset \mathbb{R}^{n_x}$, $U \subset \mathbb{R}^{n_u}$, $D \subset \mathbb{R}^{n_d}$, $P \subset \mathbb{R}^{n_p}$. Here, $\mathbf{x} \in X$ are the process state variables, $\mathbf{d} \in D$ are the disturbance uncertainty parameters, $\mathbf{p} \in P$ are the model uncertainty parameters and $\mathbf{u} \in U$ are the control variables that can be adjusted in response to disturbances. Upon solving program (1), if $\eta^* \le 0$, the design is said to be robust. That is, the controls can be adjusted to guarantee the performance specification is satisfied ($g(\mathbf{x}, \mathbf{u}, \mathbf{d}, \mathbf{p}) \le 0$), for all possible uncertainty realisations.

Algorithms for solving general nonconvex max–min optimisation formulations have used a cutting-plane method (Falk and Hoffman, 1977) or interval methods (Zuhe et al., 1990). Although effective for unconstrained max–min problems, these approaches cannot handle max–min problems with equality constraints. Algorithms for solving general nonconvex constrained max–min programs, motivated by complex engineering design, such as program (1), have not been developed. Likewise, the exponential worst-case runtime of all known deterministic global optimisation algorithms motivates the need for a new, reduced-space formulation.

Interval methods have been applied to simulation problems previously. In Schnepper and Stadtherr (1996), the authors use interval methods to find and bound all steady-state solutions to process model equations. Their definition of robust simulation is the guarantee that upon termination of their algorithm, all real steady-state solutions will be bounded with mathematical certainty. They assume that model parameters and inputs are known with absolute certainty, and thus their model equations do not account for

parametric uncertainty. In Byrne and Bogle (2000) and Balendra and Bogle (2009), the authors apply interval methods to optimise process systems models. Again, it is assumed that inputs and model parameters are known with absolute certainty.

In Ben-Tal and Nemirovski (2002), the authors discuss the idea of robust optimisation in the sense of uncertain parameters, and the solution of robust optimisation problems. However, the work in Ben-Tal and Nemirovski (2002) is limited to linear programs, quadratic programs and semi-definite programs. Since even the simplest of engineering design models can be nonconvex, their approach cannot be applied to solve general robust simulation problems, as is the focus in this paper.

1.2 SIP reformulation

Assuming differentiability of **h** and the closed convex hull of the Jacobian matrix with respect to **x** on an open set $Q \subset \mathbb{R}^{n_x}$ does not enclose any singular matrices, then:

$$\text{if } \exists \mathrm{x} \in X : \mathrm{h}(\mathbf{x},\mathbf{u},\mathbf{d},\mathbf{p}) = \mathbf{0} \;\Rightarrow \mathbf{x} = \mathbf{x}(\mathbf{u},\mathbf{d},\mathbf{p}), \tag{2}$$

by asserting the Implicit Function Theorem, with $X \subset Q$. By representing the internal state variables **x** as implicit functions of the controls and uncertainty parameters, there is a significant reduction in size of the optimisation problem. Likewise, by reformulating the inner program as:

$$g(\mathbf{x}(\mathbf{u},\mathbf{d},\mathbf{p}),\mathbf{u},\mathbf{d},\mathbf{p}) \geq \eta, \quad \forall \mathbf{u} \in U,$$

the problem becomes a single-level optimisation problem with a finite number of decision variables subject to an infinite number of constraints, also known as a semi-infinite program (SIP). If these two techniques are applied, the constrained max–min program (1) can be reformulated as:

$$\begin{aligned} &\eta^* = \max_{(\mathbf{d},\mathbf{p}) \in D \times P,\, \eta \in \mathbb{R}} \eta \\ &\text{s.t. } \eta \leq g(\mathbf{x}(\mathbf{u},\mathbf{d},\mathbf{p})\mathbf{u},\mathbf{d},\mathbf{p}), \quad \forall \mathbf{u} \in U. \end{aligned} \tag{3}$$

The key difference from regular SIPs, in general, is that the semi-infinite constraint function *g* is implicitly defined and therefore not known explicitly. It should be noted that the introduced variable η need not be bounded since it is not required to branch on it in order to solve program (3) globally using the proposed algorithm. The primary focus of this paper will be on solving SIPs with embedded implicit functions of the form (3), which arise in robust simulation applications. However, the tools and results summarised in this paper apply, in general, to any SIP where the semi-infinite constraint is implicitly defined.

2 Background

This section provides the reader with an overview of the background mathematical concepts used in the proposed algorithm for solving the program (3). Let $\Xi \subset \mathbb{R}^{n_x}$, $\Omega \subset \mathbb{R}^{n_u}$, $\Delta \subset \mathbb{R}^{n_d}$, $\Pi \subset \mathbb{R}^{n_p}$ be open sets with $X \subset \Xi$, $U \subset \Omega$, $D \subset \Delta$, $P \subset \Pi$ as intervals. A required assumption for some fixed-point iterations is that for all

considered models, $\mathbf{h} : \Xi \times \Omega \times \Delta \times \Pi \to \mathbb{R}^{n_x}$ is continuously differentiable. For those that do not rely on derivative information, the model equations must be continuous on their domain.

2.1 Interval analysis

The algorithm for solving SIPs constrained by implicit functions as in program (3) relies on interval analysis to generate valid enclosures of the range of implicit functions over the uncertainty and control domains. This section will present the definitions and nomenclature of interval analysis in a manner attempting to preserve generality. For a more comprehensive discussion of interval methods, the reader is directed to Moore (1979) and Neumaier (1990).

Definition 1: *An interval* $Z \subset \mathbb{R}$ *is defined as the compact set*:

$$Z = \{z \in \mathbb{R} : z^L \le z \le z^U\},$$

with $z^L \in \mathbb{R}$ *and* $z^U \in \mathbb{R}$ *as the lower and upper bounds of the interval Z.*

Definition 2: *The set of all interval subsets of* $\mathbb{R}$ *is denoted* $\mathbb{IR}$.

Definition 3: *An interval* $Z \in \mathbb{IR}^m$ *is defined as an m-dimensional vector whose elements are intervals denoted by a subscript* Z_i *for i = 1, … , m. An interval* $Z \in \mathbb{IR}^{m \times n}$ *is an m × n dimensional matrix whose elements are intervals denoted by a subscript* Z_{ij} *for i = 1, … , m and j = 1, … , n.*

Definition 4: *Let* $A \subset \mathbb{R}^m$. *The set* $\{Z \in \mathbb{IR}^m : Z \subset A\}$ *is denoted as* $\mathbb{I}A$.

Proposition 1: *An interval* $Z \in \mathbb{IR}^m$ *can be composed from n subinterval partitions,* $Z^i \subset Z$, *such that* $Z = \bigcup_{i=1}^{n} Z^i$.

Definition 5 (Midpoint): *The 'midpoint' or 'median' of an interval* $Z \in \mathbb{IR}$ *will be defined as*:

$$m(Z) \equiv \frac{z^L + z^U}{2}. \tag{4}$$

For $Z \in \mathbb{IR}^m$, *the midpoint will be a real vector m(Z) whose i-th element is* $m(Z_i)$. *Similarly, for* $Z \in \mathbb{IR}^{m \times n}$, *the midpoint will be a real matrix m(Z) whose (i, j)-th element is* $m(Z_{ij})$.

Definition 6 (Image): *The 'image' of the set Z under the mapping* $\mathbf{f} : A \subset \mathbb{R}^m \to \mathbb{R}^n$, *with* $Z \in \mathbb{I}A$ *is denoted as* $\hat{\mathbf{f}}(Z)$.

Definition 7 (Interval Hull): *Let* $A \subset \mathbb{R}^m$ *be bounded.* $\square A \in \mathbb{IR}^m$ *is called the 'interval hull' of A if* $A \subset \square A$ *and for any* $Z \in \mathbb{IR}^m$ *such that* $A \subset Z, \square A \subset Z$.

Definition 8: *The 'interval hull' of the image of* $Z \in \mathbb{I}A$ *under* $\mathbf{f} : A \subset \mathbb{R}^m \to \mathbb{R}^n$ *is denoted as* $\square\hat{\mathbf{f}}(Z) = \left[\hat{\mathbf{f}}^L(Z), \hat{\mathbf{f}}^U(Z)\right]$.

Definition 9: *Let* $A \subset \mathbb{R}^m$. *An interval-valued function* $F : \mathbb{I}A \to \mathbb{IR}^n$, *evaluated at any* $Z \in \mathbb{I}A$, *is denoted as F(Z).*

Definition 10 (Interval Extension): *Let* $A \subset \mathbb{R}^m$. *An interval-valued function* $F : \mathbb{I}A \to \mathbb{IR}^n$ *is called an 'interval extension' of the real-valued function* $\mathbf{f} : A \to \mathbb{R}^n$ *on* $Z \in \mathbb{I}A$, *if*

$$\mathbf{f}(\mathbf{z}) = \mathbf{y} = [\mathbf{y}, \mathbf{y}] = F([\mathbf{z}, \mathbf{z}]), \quad \forall \mathbf{z} \in Z.$$

Definition 11: *An interval extension, F, of the function* $\mathbf{f} : A \subset \mathbb{R}^m \to \mathbb{R}^n$, *is called 'exact' at* $Z \in \mathbb{I}A$, *if* $F(Z) = \square\hat{\mathbf{f}}(Z)$.

Definition 12 (Inclusion Monotonicity) (Moore, 1979; Neumaier, 1990): *Let* $Z \in \mathbb{IR}^m$. *An interval-valued function* $F : \mathbb{I}Z \to \mathbb{IR}^n$ *is called 'inclusion monotonic' on Z if for every* $A, B \in \mathbb{I}Z$

$$B \subset A \Rightarrow F(B) \subset F(A). \tag{5}$$

Definition 13 (Nested Sequences): *A sequence of intervals* $\{Z^k\}$ *is said to be 'nested' if* $Z^{k+1} \subset Z^k$ *for every k.*

Definition 14 (Inclusion Function): *Let* $Z \subset \mathbb{R}^m$. *An interval-valued function* $F : \mathbb{I}Z \to \mathbb{IR}^n$ *is called an 'inclusion function' of* $\mathbf{f}$ *on Z if*

$$\hat{\mathbf{f}}(A) \subset F(A), \quad \forall A \in \mathbb{I}Z.$$

Theorem 1 (Moore, 1979; Neumaier, 1990): *Let* $Z \in \mathbb{IR}^m$. *If* $F : \mathbb{I}Z \to \mathbb{IR}^n$ *is an inclusion monotonic interval extension of* $\mathbf{f} : Z \to \mathbb{R}^n$ *on* $A \in \mathbb{I}Z$, *then F is an inclusion function of* $\mathbf{f}$ *on A.*

Proof: Proof can be found in Moore (1979, p.21).

This property is also known as the Fundamental Theorem of Interval Analysis.

Definition 15 (Interval Width): *The 'width' of an interval* $Z \in \mathbb{IR}$ *is defined as the distance between its upper and lower bounds*:

$$w(Z) = z^U - z^L.$$

Definition 16 (Interval Vector Width): *The 'width' of an interval vector* $Z \in \mathbb{IR}^m$ *is defined as the real vector* $\mathbf{w}(Z)$ *whose i-th element is* $w(Z_i)$.

Definition 17 (Excess Width) (Moore, 1979): *Let F be an inclusion monotonic interval extension of* $\mathbf{f}$ *at* $Z \in \mathbb{IR}^m$. *Then* $F(Z) = \square\hat{\mathbf{f}}(Z) + E(Z)$ *for some interval-valued function E with* $\mathbf{w}(F(Z)) = \mathbf{w}(\square\hat{\mathbf{f}}(Z)) + \mathbf{w}(E(Z))$. *Then* $\mathbf{w}(E(Z)) = \mathbf{w}(F(Z)) - \mathbf{w}(\square\hat{\mathbf{f}}(Z))$ *is called the 'excess width' of F(Z).*

Definition 18: *An interval extension of the partial derivative of the continuously differentiable function* $f_i : A \subset \mathbb{R}^m \to \mathbb{R}$ *with respect to* z_j *is denoted as*:

$$\frac{\partial F_i}{\partial z_j}(Z)$$

for $Z \in \mathbb{I}A$.

Definition 19: *An inclusion monotonic interval extension of the Jacobian matrix, of a vector-valued function* $\mathbf{f} : A \subset \mathbb{R}^m \to \mathbb{R}^n$, *at* $Z \in \mathbb{I}A$ *is denoted as*:

$$J_z(Z) \equiv \begin{bmatrix} \frac{\partial F_1}{\partial z_1}(Z) & \cdots & \frac{\partial F_1}{\partial z_m}(Z) \\ \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial z_1}(Z) & \cdots & \frac{\partial F_n}{\partial z_m}(Z) \end{bmatrix}. \tag{6}$$

Definition 20: *Let* $Z \in \mathbb{IR}^{n\times n}$ *and let every* $\mathbf{Z} \in \mathbb{R}^{n\times n}$ *such that* $\mathbf{Z} \in Z$ *have rank n. Then*

$$Z^{-1} \equiv \square\{\mathbf{Z}^{-1} : \mathbf{Z} \in Z\}$$

is the inverse of an interval matrix.

2.2 Parametric interval Newton-type methods

Parametric interval Newton-type methods are the primary tools for generating valid enclosures of implicit functions which are required for solving program (3). The theory and a detailed discussion of the useful properties of parametric interval Newton methods are contained in a paper currently in preparation by the authors (Stuber and Barton, 2011), and are only summarised here. One useful similarity between all the parametric interval Newton-type methods is that they define nested sequences of intervals. Define $\mathbf{y} = (\mathbf{d}, \mathbf{p})$ to be the general uncertainty variable and let $Y = D \times P$ represent the general uncertainty set such that $\mathbf{y} \in Y$. For ease of presentation, it will be assumed that an initial interval $X^0 \in \mathbb{I}X$ is known that contains the value of the implicit function $\mathbf{x}(\mathbf{u}, \mathbf{y})$, for every $(\mathbf{u}, \mathbf{y}) \in U \times Y$. Such an assumption is not restrictive for practical problems where variables and functions have physically meaningful bounds. Under this assumption, the basic application of the methods is guaranteed to converge finitely to some interval $X^* \subset X^0$ that is also guaranteed to enclose the implicit solution. However, this assumption is not necessary for convergence to an interval enclosure, and the general case is discussed in a paper currently in preparation by the authors (Stuber and Barton, 2011).

Definition 21 (Parametric Interval Newton Method): *Let* $X^0 \in \mathbb{IR}^{n_x}$ *be the initial guess interval with* $X^0 \subset Q$. *Let* $H(\mathbf{x}, U, Y)$ *be an inclusion monotonic interval extension of* $\mathbf{h} : X^0 \times U \times Y \to \mathbb{R}^{n_x}$ *on* $U \times Y$ *for every* $\mathbf{x} \in X^0$. *The parametric interval Newton method is then defined as:*

$$\begin{aligned} \mathbf{x}^k &:= m(X^k) \\ N(\mathbf{x}^k, X^k, U, Y) &:= \mathbf{x}^k - J_{\mathbf{x}}(X^k, U, Y)^{-1} H(\mathbf{x}^k, U, Y) \\ X^{k+1} &:= X^k \bigcap N(\mathbf{x}^k, X^k, U, Y). \end{aligned} \tag{7}$$

It is assumed for simplicity that for $X^0 \subset Q$, $J_{\mathbf{x}}(X^0, U, Y)$ does not contain any singular matrices, and therefore $J_{\mathbf{x}}(X^k, U, Y)$ also does not contain any singular matrices for every $X^k \subset X^0$. It should be noted that $\mathbf{x}^k$ does not need to be the midpoint of X^k but

can be any point $\mathbf{x} \in X^k$. However, the midpoint is convenient and may offer some useful properties to the interval Newton-type algorithms (Moore, 1979; Neumaier, 1990; Hansen and Walster, 2004). For better convergence and enclosure properties, a sequential componentwise calculation of *N* is recommended. This technique is known as the *Gauss–Seidel* method and it is a result of arranging the parametric interval Newton operator, *N*, into the linear system

$$J_{\mathbf{x}}(X, U, Y)(N(\mathbf{x}, X, U, Y) - \mathbf{x}) = -H(\mathbf{x}, U, Y) \tag{8}$$

and solving for $N(\mathbf{x}, X, U, Y)$ componentwise. It is also important to mention that preconditioning (8) by some real matrix $\mathbf{\Psi}$ is recommended to give better convergence and enclosure properties (Kearfott, 1990, 1996; Neumaier, 1990). It is common that $\mathbf{\Psi}$ is taken to be an approximate inverse of the midpoint of the interval Jacobian matrix *J*. The Gauss–Seidel method is defined in the following.

Definition 22 (Parametric Interval Newton with Gauss–Seidel): *For* $i = 1, 2, ..., n_x$:

$$N_i^k := x_i^k - \left[B_i^k + \sum_{j=1}^{i-1} A_{i,j}^k \left(X_j^{k+1} - x_j^{k+1} \right) + \sum_{j=i+1}^{n_x} A_{i,j}^k \left(X_j^k - x_j^k \right) \right] / A_{i,i}^k \tag{9}$$

$$X_i^{k+1} := N_i^k \cap X_i^k,$$

with $A^k \in \mathbb{IR}^{n_x \times n_x}$ *as* $A^k = \mathbf{\Psi}^k J_{\mathbf{x}}(X^k, U, Y)$ *and* $B^k \in \mathbb{IR}^{n_x}$ *as* $B^k = \mathbf{\Psi}^k H(\mathbf{x}^k, U, Y)$.

Another method that does not require the inversion of an interval matrix, or division by intervals, such as the parametric interval Newton method, is known as the parametric Krawczyk method. It is defined in the following.

Definition 23 (Parametric Krawczyk Method): *Let* $X^0 \in \mathbb{IR}^{n_x}$ *be the initial guess interval such that* $X^0 \subset Q$. *Let* $H(\mathbf{x}, U, Y)$ *be an inclusion monotonic interval extension of* $\mathbf{h}$ *on* $U \times Y$ *for every* $\mathbf{x} \in X^0$. *The parametric Krawczyk method is defined as*:

$$\begin{aligned} \mathbf{x}^k &:= m(X^k) \\ \mathbf{\Psi}^k &:= \left[m(J_{\mathbf{x}}(X^k, U, Y)) \right]^{-1} \\ K(\mathbf{x}^k, X^k, U, Y) &:= \mathbf{x}^k - \mathbf{\Psi}^k H(\mathbf{x}^k, U, Y) + (\mathbf{I} - \mathbf{\Psi}^k J_{\mathbf{x}}(X^k, U, Y))(X^k - \mathbf{x}^k) \\ X^{k+1} &:= K(\mathbf{x}^k, X^k, U, Y) \cap X^K. \end{aligned} \tag{10}$$

Similar to the Gauss–Seidel implementation for the parametric interval Newton method above, a componentwise sequential calculation of the *K* operator, with preconditioning, is preferred to give better convergence and enclosure properties. The implementation is defined in the following.

Definition 24 (Componentwise Parametric Krawczyk Method): *For* $i = 1, 2, ..., n_x$:

$$\begin{aligned} K_i^k &:= x_i^k - B_i^k + \sum_{j=1}^{i-1} A_{i,j} \left(X_j^{k+1} - x_j^k \right) + \sum_{j=i}^{n_x} A_{i,j} \left(X_j^k - x_j^k \right), \\ X_i^{k+1} &:= K_i^k \cap X_i^k \end{aligned} \tag{11}$$

with $A^k \in \mathbb{IR}^{n_x \times n_x}$ *as* $A^k = \mathbf{I} - \boldsymbol{\Psi}^k J_{\mathbf{x}}(X^k, U, Y)$ *and* $B^k \in \mathbb{IR}^{n_x}$ *as* $B^k = \boldsymbol{\Psi}^k H(\mathbf{x}^k, U, Y)$, *where* $\boldsymbol{\Psi}^k$ *is taken as defined above.*

In Neumaier (1990), it is shown that $N \subset K$ when implemented componentwise, for the non-parametric case. The extension to the parametric case can be made and the result holds. However, since N requires interval divisions, the parametric interval Newton method may not be recommended over the parametric Krawczyk method for all systems. The last parametric interval method considered is the nested parametric interval successive substitution method. It is essentially a nested interval form of the generic successive substitution method for real numbers extended to parameter-dependent systems. This is also known as the forward–backward contractor for constraint propagation (Jaulin et al., 2001) and has been studied in the past. In Balendra and Bogle (2009), this approach is applied to process systems for the non-parametric case. Although it was shown in Balendra and Bogle (2009) that the interval Newton method offered better bounds, it is the simplest interval method and worth mentioning.

Definition 25 (Nested Parametric Interval Successive Substitution): *Let* $\mathbf{f}(\cdot, \mathbf{u}, \mathbf{y}) : X \to X$ *be an algebraic rearrangement of* $\mathbf{h}(\cdot, \mathbf{u}, \mathbf{y})$: $X \to \mathbb{R}^{n_x}$ *such that* $\mathbf{h}(\mathbf{x}, \mathbf{u}, \mathbf{y}) = \mathbf{0} \Leftrightarrow \mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{y})$. *Let* F *be an inclusion monotonic interval extension of* $\mathbf{f}$ *on* $X^0 \times U \times Y$ *with* $X^0 \in \mathbb{I}X$. *The Nested Parametric Interval Successive Substitution method is then defined as:*

$$\begin{aligned} S(X^k, U, Y) &:= F(X^k, U, Y) \\ X^{k+1} &:= S(X^k, U, Y) \bigcap X^k. \end{aligned} \tag{12}$$

The nested parametric interval successive substitution method offers a few useful properties. One property is that it does not require derivative information. Although this method is rather restrictive to relatively simple systems, it is a computationally inexpensive alternative. Also, under some assumptions, the converged interval will be the interval hull of the image. Thus, it has the potential to yield the tightest possible enclosure of the range of the implicit function. The following example illustrates such a case and compares the three parametric interval methods.

Example 1: *Let us set* $\mathbf{q} = (\mathbf{u}, \mathbf{y})$. *Let the steady-state model of interest be:*

$$\mathbf{h}(\mathbf{x}, \mathbf{q}) = \begin{pmatrix} x_1^2 + x_2^2 + q_1 x_1 + 4 \\ x_1 + q_2 x_2 \end{pmatrix} = 0$$

with $\mathbf{q} \in \Theta = [5, 7] \times [5, 7]$. *Let* $X^0 = [-1.5, 0] \times [0, 0.5]$. *The successive substitution method can be applied if the problem is reformulated such that* $\mathbf{h}(\mathbf{x}, \mathbf{q}) = \mathbf{0}$ *is rewritten as* $\mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{q})$:

$$\mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{q}) = \begin{pmatrix} -(x_1^2 + x_2^2 + 4)/q_1 \\ -x_1/q_2 \end{pmatrix}.$$

Likewise, the parametric interval Newton and parametric Krawczyk methods can also be applied. On this Θ *interval, the exact and approximate interval hull of the solution is:*

$$\square\hat{\mathbf{x}}(\Theta) = \begin{pmatrix} \left[(5\sqrt{209}-125)/52,\ (7\sqrt{1601}-343)/100\right] \\ \left[(49-\sqrt{1601})/100,\ (25-\sqrt{209})/52\right] \end{pmatrix}. \approx \begin{pmatrix} [-1.0137661...,\ -0.629125...] \\ [0.089875...,\ 0.202753...] \end{pmatrix}$$

The three algorithms were implemented using INTLAB (Rump, 1999). The algorithms terminate finitely when $X^{k+1} = X^k$, *which is an inherent stopping criterion for all interval iterations when outward rounding techniques are employed to approximate real intervals to machine precision. Tables 1–3 display the results of the three interval algorithms. We see* $X_S^*(\Theta) \subset X_N^*(\Theta) \subset X_K^*(\Theta)$, *where* $X_S^*(\Theta)$ *approximates (rigorous) the interval hull to machine precision.*

Table 1 Parametric interval successive substitution

i	x_i^{*L}	x_i^{*U}	$w(E(X_i^*))$
1	–1.013766…	–0.629125…	0
2	0.0898750…	0.202753…	0

Note: After 44 iterations (0.13 s), the parametric interval successive substitution method terminates. The interval encloses the hull exactly.

Table 2 Parametric interval Newton's method

i	x_i^{*L}	x_i^{*U}	$w(E(X_i^*))$
1	–1.04243…	–0.49276…	0.16503…
2	0.047379…	0.208486…	0.04823…

Note: After 24 iterations (0.24 s), the parametric interval Newton method with interval Gauss–Seidel terminates.

Table 3 Parametric Krawczyk method

i	x_i^{*L}	x_i^{*U}	$w(E(X_i^*))$
1	–1.04243…	–0.49276…	0.16503…
2	0.047379…	0.208486…	0.04823…

Note: After 54 iterations (0.57 s), the parametric Krawczyk method implemented componentwise terminates.

2.3 McCormick relaxations

McCormick's relaxations (McCormick, 1976) are used in the proposed robust simulation algorithm to construct (nonsmooth) convex and concave relaxations of nonconcave functions, which are guaranteed to overestimate the nonconcave function. The direct application of McCormick relaxations is in calculating valid upper bounds for global maximisation problems. In Mitsos et al. (2009), it was shown how to calculate McCormick relaxations and subgradients of (nonconvex or nonconcave) functions

evaluated by algorithms. They also automate the process using the C++ library libMC (Chachuat, 2007), which can calculate relaxations and subgradients automatically, similar to automatic differentiation (AD). The proposed application and key result is in solving large global optimisation problems in a reduced space (Mitsos et al., 2009). Their results stop short of considering fixed-point algorithms such as Newton's method, where the number of iterations is not known *a priori.* We have extended these McCormick-based relaxations of algorithms to include fixed-point algorithms. The theory behind the general construction of convex/concave relaxations of algorithms can be found in Mitsos et al. (2009), with the automatic implementation discussed in Chachuat (2007). This section will be used to summarise the basic concepts and define the nomenclature.

Definition 26 (Relaxation of Functions): *Given a convex set* $Z \subset \mathbb{R}^n$ *and a function* $f : Z \to \mathbb{R}$*, a convex function* $f^c : Z \to \mathbb{R}$ *is a convex relaxation (or convex underestimator) of f on Z if* $f^c(\mathbf{z}) \le f(\mathbf{z})$ *for every* $\mathbf{z} \in Z$*. A concave function* $f^C : Z \to \mathbb{R}$ *is a concave relaxation (or concave overestimator) of f on Z if* $f^C(\mathbf{z}) \ge f(\mathbf{z})$ *for every* $\mathbf{z} \in Z$*.*

A key assumption made on **h** and *g* in this paper is that they are factorable functions.

Definition 27 (Factorable Function): *A function is factorable if it is defined by a finite recursive composition of binary sums, binary products and a given library of univariate intrinsic functions.*

Definition 28 (McCormick Relaxations): *The relaxations of a factorable function that are formed via the recursive application of rules for the relaxation of univariate composition, binary multiplication, and binary addition from convex and concave relaxations of the univariate intrinsic functions, without the introduction of auxiliary variables, are termed McCormick relaxations.*

The general rules for constructing McCormick relaxations are outlined in McCormick (1976), Mitsos et al. (2009) and Scott et al. (2011). In order to calculate valid convex/concave relaxations of implicit functions, it is necessary to first know a valid estimate of their range. Therefore, parametric interval Newton-type methods are required for constructing convex/concave relaxations of nonconvex/nonconcave implicit functions. The second step in constructing convex/concave relaxations of implicit functions is to rearrange the parametric system of equations into a fixed-point form:

$$\mathbf{h}(\mathbf{x}, \mathbf{u}, \mathbf{y}) = \mathbf{0} \Leftrightarrow \mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{y}).$$

In general, this can be done for any parametric system of equations $\mathbf{h}(\mathbf{x}, \mathbf{u}, \mathbf{y}) = \mathbf{0}$ (Ortega and Rheinboldt, 1970). For example

$$\Psi\mathbf{h}(\mathbf{x},\mathbf{u},\mathbf{y}) = \mathbf{0} = \mathbf{x} - \mathbf{x}$$
$$\Rightarrow \mathbf{x} = \mathbf{f}(\mathbf{x},\mathbf{u},\mathbf{y}) = \mathbf{x} - \Psi\mathbf{h}(\mathbf{x},\mathbf{u},\mathbf{y}),$$

again with Ψ as a preconditioning matrix. This is used to define the iteration:

$$\mathbf{x}^{k+1} := \mathbf{f}(\mathbf{x}^k, \mathbf{u}, \mathbf{y}).$$

Taking $\Psi^k = \mathbf{J}_{\mathbf{x}}(\mathbf{x}^k, \mathbf{u}, \mathbf{y})^{-1}$ results in Newton's method, or taking $\Psi = \mathbf{I}$ results in the Picard iteration. Now, convex/concave relaxations of $\mathbf{x}^{k+1}$ on $U \times Y$ can be calculated by applying a generalisation of McCormick's method (Scott et al., 2011) to calculate convex/concave relaxations of $\mathbf{f}(\mathbf{x}^k, \mathbf{u}, \mathbf{y})$ on $U \times Y$, given convex/concave relaxations of $\mathbf{x}^k$ on $U \times Y$. Assuming the iteration $\mathbf{x}^{k+1} := \mathbf{f}(\mathbf{x}^k, \mathbf{u}, \mathbf{y})$ is a contraction mapping on X

for each $(\mathbf{u}, \mathbf{y}) \in U \times Y$, the sequence of iterates will converge to the implicit function $\{\mathbf{x}^k(\mathbf{u}, \mathbf{y})\} \rightarrow \mathbf{x}(\mathbf{u}, \mathbf{y})$, for each $(\mathbf{u}, \mathbf{y}) \in U \times Y$. Therefore, calculating convex/concave relaxations of $\mathbf{x}^{k+1}$ until $\left\|\mathbf{x}^{k+1}(\mathbf{u},\mathbf{y}) - \mathbf{x}^k(\mathbf{u},\mathbf{y})\right\| \leq \varepsilon$, $\forall(\mathbf{u},\mathbf{y})$ results in valid convex and concave relaxations of the implicit function on $U \times Y$, $\mathbf{x}^c$ and $\mathbf{x}^C$, respectively, if the relaxations were initialised as rigorous lower and upper bounds, respectively, on its range, calculated via a parametric interval Newton method. The full theoretical details for calculating relaxations of parametric solutions to nonlinear systems have been worked out in a paper currently in preparation by the authors (Stuber et al., 2011).

2.4 Nonsmooth programs

Since McCormick-based relaxations are nonsmooth, their incorporation into an NLP results in a nonsmooth convex NLP. Likewise, inclusion functions, which the proposed algorithm also relies on, are also potentially nonsmooth. Therefore, standard gradient-based methods for NLPs, such as Sequential Quadratic Programming (SQP) (Gill et al., 2005), cannot be used. In Mitsos et al. (2009), the authors used the idea of *affine relaxations* of the nonsmooth convex relaxations to generate a new optimisation problem with linear objective function and constraints whose solution is a lower bound on the original nonconvex problem. In order to calculate affine relaxations, subgradient information is required.

Definition 29 (Subgradient): *Let $Z \subset \mathbb{R}^n$ be a nonempty convex set, $f^c : Z \rightarrow \mathbb{R}$ be convex and $f^C : Z \rightarrow \mathbb{R}$ be concave. A vector $\mathbf{s}^c \in \mathbb{R}^n$ is called a subgradient of f^c at $\overline{\mathbf{z}} \in Z$ if $f^c(\mathbf{z}) \geq f^c(\overline{\mathbf{z}}) + (\mathbf{s}^c)^{\mathrm{T}}(\mathbf{z} - \overline{\mathbf{z}})$ for all $\mathbf{z} \in Z$. Likewise, a vector $\mathbf{s}^C \in \mathbb{R}^n$ is called a subgradient of f^c at $\overline{\mathbf{z}} \in Z$ if $f^C(\mathbf{z}) \leq f^C(\overline{\mathbf{z}}) + (\mathbf{s}^C)^{\mathrm{T}}(\mathbf{z} - \overline{\mathbf{z}})$ for all $\mathbf{z} \in Z$.*

Definition 30 (Affine Relaxation): *Let $Z \subset \mathbb{R}^n$ be a nonempty convex set, $f^c : Z \rightarrow \mathbb{R}$ be a convex relaxation and $f^C : Z \rightarrow \mathbb{R}$ be a concave relaxation of $f : Z \rightarrow \mathbb{R}$ on Z, respectively. Let $\mathbf{s}^c$ and $\mathbf{s}^C$ be corresponding subgradients of f^c* and *f^c at some point $\overline{\mathbf{z}} \in Z$, respectively. Then $f^{cl}(\mathbf{z}) \equiv f^c(\overline{\mathbf{z}}) + (\mathbf{s}^c)^{\mathrm{T}}(\mathbf{z} - \overline{\mathbf{z}})$ and $f^{Cl}(\mathbf{z}) \equiv f^C(\overline{\mathbf{z}}) + (\mathbf{s}^C)^{\mathrm{T}}(\mathbf{z} - \overline{\mathbf{z}})$ are an affine underestimator and affine overestimator of f on Z, respectively*.

One may also wish to solve the nonsmooth convex problem directly, rather than calculate a valid lower bound, using a nonsmooth optimisation method such as a *bundle method.* However, this approach could be more expensive than the subgradient approach, and the details for constrained problems need to be worked out. Solving nonsmooth optimisation problems using bundle methods has been discussed in the literature (Mäkelä, 2001). These methods require that the objective function value and a subgradient can be evaluated at all desired points, a property that holds regardless of the embedded implicit functions from the discussion in Section 2.3. The rules for calculating subgradients are outlined in Mitsos et al. (2009). Likewise, in Mitsos et al. (2009), the authors show how subgradients can be propagated through McCormick relaxations. The calculation of subgradients of implicit functions through McCormick relaxations of implicit functions is analogous and can be automated using libMC. Therefore, subgradient information for the implicit functions, as well as the objective function, is

available. This allows users of libMC to select the affine relaxation approach to calculate a valid upper bound on the solution of the nonsmooth maximisation problem or to employ a novel nonsmooth solver directly.

3 Robust simulation

Solving SIPs globally with explicit constraints using interval methods and McCormick's convex relaxations within the branch-and-bound framework is discussed in Bhattacharjee et al. (2005b). In Bhattacharjee et al. (2005b), the authors generate upper and lower bounding problems for the original SIP that are refined through the branch-and-bound framework. The sequences of upper and lower bounds are guaranteed to converge to the true solution of the SIP under some relatively mild assumptions (Bhattacharjee et al., 2005a, 2005b). In order to solve SIPs constrained by implicit functions, as in equation (3), a method similar to Bhattacharjee et al. (2005b) is applied.

An alternative lower bounding procedure was proposed in Mitsos et al. (2008). The extension to implicit functions is straightforward by applying the novel theoretical results described briefly in Section 2.3. This method may similarly be well suited for SIPs with implicit functions; however, further investigation is necessary to determine benefits (if any) and relative cost-per-iteration.

It should be noted that other methods for approximating the solution of SIPs with explicit functions have been developed. In Hettich and Kortanek (1993) an overview of methods for solving general SIPs is given. In Falk and Hoffman (1977), the unconstrained max–min problem was reformulated into an SIP similar to program (3) with *g* explicit. The cutting-plane algorithm summarised in Falk and Hoffman (1977) requires solving two global optimisation problems per iteration. Therefore, the cost-per-iteration is expensive. Coupled with the fact that program (3) involves implicit functions, the application of this method to program (3) is not well suited. All of the surveyed methods produce sequences of lower bounds on the SIP which approximate the true solution of the SIP. Therefore, using these methods, no rigorous guarantee of robustness can be given since upon finite termination of the algorithm, the approximate solution is not guaranteed to be a feasible point of the SIP.

3.1 Lower bounding problem

Solving the lower bounding problem (LBP) to local optimality is guaranteed to give a feasible point for the original SIP, if one exists. (Note that program (3) is a maximisation problem.) That is, the set of feasible points of the LBP is a subset of the set of feasible points of the original SIP. Graphically, this means the feasible region of the LBP is an inner approximation of the feasible region of the SIP. This result is especially useful in the case when simply guaranteeing feasibility of a proposed design is sufficient. The LBP is a finite nonsmooth reformulation of the original SIP known as an interval-constrained reformulation (ICR). It is only required to solve the ICR to local optimality, if a feasible point exists, for a valid lower bound on the solution. The ICR requires partitioning of the control domain U into n subintervals such that $U = \bigcup_{i=1}^{n} U^i$ and calculating an inclusion function, G, of the implicit semi-infinite constraint over each subset U^i. The inclusion functions are calculated as inclusion monotonic interval extensions of the constraint

function g over U^i utilising a parametric interval Newton-type method to generate valid inclusions of the implicit function $\mathbf{x}$ over U^i. For example, if the standard parametric interval Newton method is employed, its form would be:

$$\begin{aligned} &\mathbf{x}^k \in X^k \\ &N(\mathbf{x}^k, \mathrm{X}^k, U, \mathbf{y}) := \mathbf{x}^k - J_{\mathbf{x}}(X^k, U, \mathbf{y})^{-1} H(\mathbf{x}^k, U, \mathbf{y}) \\ &\qquad X^{k+1} := X^k \bigcap N(\mathbf{x}^k, X, U, \mathbf{y}). \end{aligned} \tag{13}$$

The algorithm (13) will converge to the value of an interval-valued function $X(U, \mathbf{y})$. The calculation of G is then straightforward using the rules of interval arithmetic. It is explained in Bhattacharjee et al. (2005a) that given the existence of a Slater point arbitrarily close to a maximiser, as the width of the subintervals approaches degeneracy, the solution value of the ICR approaches the true global solution value of the original SIP from below, for this formulation. The ICR is defined in the following.

Definition 31 (Interval-Constrained Reformulation): *Let* $G^L(X(U^i, \mathbf{y}), U^i, \mathbf{y})$ *be the lower bound of an inclusion monotonic interval extension of* $g(\mathbf{x}(\mathbf{u}, \mathbf{y}), \mathbf{u}, \mathbf{y})$ *on the i-th subinterval* U^i. *The lower bounding ICR is as follows*:

$$\begin{aligned} &\eta^{LBD} = \max_{\mathbf{y}\in Y,\, \eta\in\mathbb{R}} \eta \\ &\text{s.t. } \eta \le G^L(X(U^i, \mathbf{y}), U^i, \mathbf{y}), \quad \forall i = 1, \ldots, n. \end{aligned} \tag{14}$$

Solving program (14) to local optimality yields a feasible point, if one exists, and is a lower bound on the solution value of the SIP, $\eta^{LBD} \le \eta^*$. It should be noted, however, that the implicit function G^L may be nonsmooth, thus program (14) may be a nonsmooth NLP.

3.2 Upper bounding problem

The upper bounding problem (UBP) is based on the discretisation technique where the semi-infinite constraint is replaced by a finite collection of constraints evaluated at each point in a finite subset of the control set U. The discretised program is as follows:

$$\begin{aligned} &\max_{\mathbf{y}\in Y,\, \eta\in\mathbb{R}} \eta \\ &\text{s.t. } \eta \le g(\mathbf{x}(\mathbf{u}_i, \mathbf{y}), \mathbf{u}_i, \mathbf{y}), \quad \forall i = 1, \ldots, n \end{aligned} \tag{15}$$

where $\mathbf{u}_i \in U$ denotes the i-th realisation of the control variable $\mathbf{u}$. As explained in Bhattacharjee et al. (2005b), as the number of discretisation points, n, increases, the solution of this finite relaxation approaches the solution to the SIP from above, for this formulation. However, since the feasible set of program (15) is likely nonconvex, in order to guarantee its solution yields a valid upper bound on the original SIP, it must be solved to global optimality. Assuming program (15) is composed of factorable functions, the results of McCormick convex/concave relaxations of fixed-point iterations on Y can be utilised to generate a valid (nonsmooth) overestimating problem, with a convex feasible set, as follows:

$$\begin{aligned} &\eta^{\mathrm{UBD}} = \max_{\mathbf{y}\in Y,\, \eta\in\mathbb{R}} \eta \\ &\text{s.t. } \eta \le g^C(\mathbf{x}^c(\mathbf{u}_i, \mathbf{y}), \mathbf{x}^C(\mathbf{u}_i, \mathbf{y}), \mathbf{u}_i, \mathbf{y}), \quad \forall i = 1, \ldots, n, \end{aligned} \tag{16}$$

where the superscripts *c* and *C* denote the convex and concave relaxations, respectively. The feasible set of program (16) is a convex outer approximation of the feasible set of the SIP. Solving program (16) yields a valid upper bound on the solution to the SIP, $\eta^{\mathrm{UBD}} \geq \eta^{*}$.

3.3 Branch & bound

The Branch & Bound (B&B) algorithm for solving nonconvex optimisation problems globally is discussed in Horst and Tuy (1997). The extension to SIPs is described in Bhattacharjee et al. (2005b). The idea here is the same for SIPs with an implicit semi-infinite constraint. The B&B framework relies on the solution of LBP and UBP over a finite number of subsets, or nodes, of the decision domain, in this case *Y*. If it is known that a node cannot contain a global maximiser $\mathbf{y}^{*}$, the node is said to be *fathomed.* The general rule for fathoming nodes is to compare the values of the η^{UBD} of each node and the overall lower bound (LBD). For instance, if $LBD > \eta_j^{UBD}$, a global maximiser cannot be in node *j*, and node *j* is said to be fathomed. Once a node is fathomed, it is no longer considered in the search space. This procedure continues by refining each node that has not been fathomed. The B&B algorithm, as applied here, is finitely convergent to ε-optimality $(\eta^{UBD} - \eta^{LBD} \leq \varepsilon_{tol}, \varepsilon_{tol} > 0)$.

3.4 Solution algorithm

1 Initialise algorithm.
   (a) Initialise stack of nodes $\Sigma = \{Y\}$.
   (b) Set convergence tolerance ε_{tot}, $LBD = -\infty$, $UBD = +\infty$.
   (c) Define partitioning and discretisation sequences for *U* for the UBP and LBP.

2 Check if $\Sigma = \varnothing$.
   (a) If true, terminate.
   (b) Else select and delete a node Y^k from Σ according to a node selection rule/heuristic.

3 LBP.
   (a) Generate valid inclusion function $G(X(U^i, \mathbf{y}), U^i, \mathbf{y})$ over each subinterval U^i using a parametric interval Newton-type method.
   (b) Solve ICR (14) on Y^k for a feasible point (if one exists). If a feasible point is found get η_k^{LBD}.
   (c) If no feasible point is found, set $\eta_k^{LBD} := -\infty$.
   (d) If $\eta_k^{LBD} > 0$, terminate algorithm (design infeasible).
   (e) Else if $\eta_k^{LBD} > LBD$, set $LBD := \eta_k^{LBD}$.

4 UBP.
   (a) Calculate valid bounds for implicit semi-infinite constraint on Y^k at each discrete $\mathbf{u}_i \in U$ using a parametric interval Newton-type method.
   (b) Generate concave relaxation on Y^k and solve UBP (16) on Y^k, get η_k^{UBD}.

(c) Set $UBD := \max_{Y^k \in \Sigma} \eta_k^{UBD}$.

(d) If $UBD \leq 0$, terminate algorithm (design feasible).

5 Check $\eta_k^{UBD} < LBD$ or $\eta_k^{UBD} = -\infty$.

(a) If true, go to 2 (Y^k has been fathomed).

6 Check $UBD - LBD \leq \varepsilon_{tol}$.

(a) If true, set $\eta^* = LBD$, terminate algorithm.

7 Branching

(a) Branch on Y^k according to a branching rule.

(b) Push new nodes onto stack Σ.

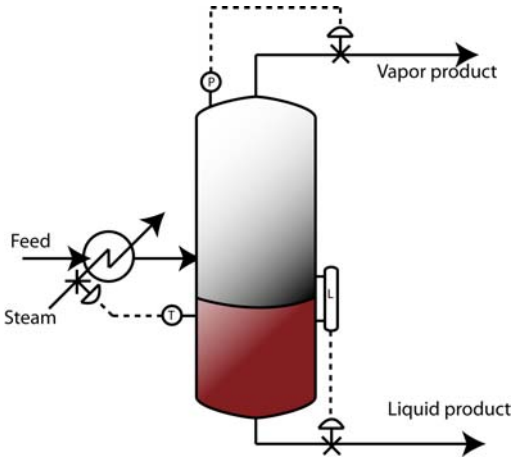
(c) Go to 2.

Step 6 may appear to be a redundant termination criterion. The algorithm may encounter a scenario when $UBD > 0$ and $LBD \leq 0$. In which case, step 6 may be required to guarantee finite convergence. If the algorithm terminates at step 6, no rigorous guarantee of robust feasibility can be made. This is a topic of further research.

3.5 Numerical example

Example 2 (Flash Separation of Benzene and Toluene): *As an illustrative example, take the benzene/toluene flash separation originally introduced in King (1980) (shown in Figure 1). We wish to guarantee the process meets robustly a preset performance specification on the separation. Uncertainty in the separator temperature* τ*, which could be caused by fluctuations in the incoming steam temperature or faulty sensor readings, will be taken into account. The pressure in the separator, p, will be the only thing we can control, say via the valve on the vapour line. The uncertainty temperature interval will be* T = [80, 110]°C *and the control interval will be* P = [90, 100] *torr. The performance constraint is such that the cut-fraction* α *must be less than or equal to* 0.7 ($g(\alpha, p, \tau) = \alpha - 0.7$).

Figure 1 The continuous equilibrium flash separator as taken from King (1980) (see online version for colours)



Of course, it should be noted that any $\alpha \notin [0, 1]$ *is non-physical. This problem is stated as the following robust simulation problem:*

$$\begin{aligned} &\max_{\tau\in T,\ \eta\in\mathbb{R}} \eta \\ \text{s.t.}\quad &\eta \le \min_{\alpha\in A,\ p\in P} g(\alpha, p, \tau) \\ &\quad \text{s.t.}\quad h(\alpha, p, \tau) = 0 \\ &\qquad P = [90,100] \\ &\qquad T = [80,110] \\ &\qquad A = [0,1]. \end{aligned} \tag{17}$$

Solving the steady-state process model, h, for the cut-fraction as an implicit function of pressure and temperature, $\alpha(p, \tau)$*, the equivalent SIP can be formulated as follows:*

$$\begin{aligned} &\max_{\tau\in T,\ \eta\in\mathbb{R}} \eta \\ \text{s.t.}\quad &\eta \le g(\alpha(p,\tau), p, \tau) = \alpha(p,\tau) - 0.7, \quad \forall p \in P \\ &P = [90,100] \\ &T = [80,110]. \end{aligned} \tag{18}$$

The steady-state process model equation is written as:

$$h(\alpha, p, \tau) = \sum_{i=1}^{2} \frac{z_i (K_i(\tau, p) - 1)}{(K_i(\tau, p) - 1)\alpha + 1} = 0, \tag{19}$$

where z_i *is the mole-fraction of component i in the feed, taken to be 0.5, and* $K_i : T \times P \to \mathbb{R}$ *is the vapour–liquid equilibrium distribution coefficient of component i, defined as:*

$$K_i(\tau, p) = \frac{p_i^{sat}(\tau)}{p} \tag{20}$$

with

$$\log_{10} p_i^{sat}(\tau) = A_i - \frac{B_i}{C_i + \tau}. \tag{21}$$

The Antoine constants A_i*,* B_i *and* C_i *are shown in Table 4. The pressure-domain is partitioned into two subintervals at the root node of the B&B tree,* $P_1 = [90, 95]$ *and* $P_2 = [95, 100]$*, for use in the LBP. Likewise, the P-domain is discretised into three points corresponding to the endpoints of the subintervals* P_1 *and* P_2*,* $p_1 = 90$*,* $p_2 = 95$ *and* $p_3 = 100$*. The parametric interval Newton method* (7) *was applied to calculate valid ranges for the implicit function at each discrete pressure value and to calculate the inclusion functions valid over each pressure subinterval. Corresponding concave relaxations of the implicit semi-infinite constraint on T were calculated automatically with* libMC *using a convergence tolerance on Newton's method of* 10^{-8}*. Figure 2 is a plot of the discrete implicit constraints, g, and their corresponding concave relaxation. It should be noted that due to the relative weakness of the relaxations for this system on the entire T interval, the concave relaxations were taken as the global upper bound,* g^U*, as dictated by the rules for constructing generalised McCormick relaxations. For illustrative purposes, Figure 3 shows concave relaxations of g corresponding to*

branches in T at p = 90. For τ ≥ 98, the calculated relaxations are greater than the upper bound g^U; therefore, by the rules of generalised McCormick relaxations, the concave relaxations would be taken as the rigorous upper bound g^U.

Figure 2 Implicit constraints and their corresponding concave relaxations as the rigorous upper bound g^U (see online version for colours)

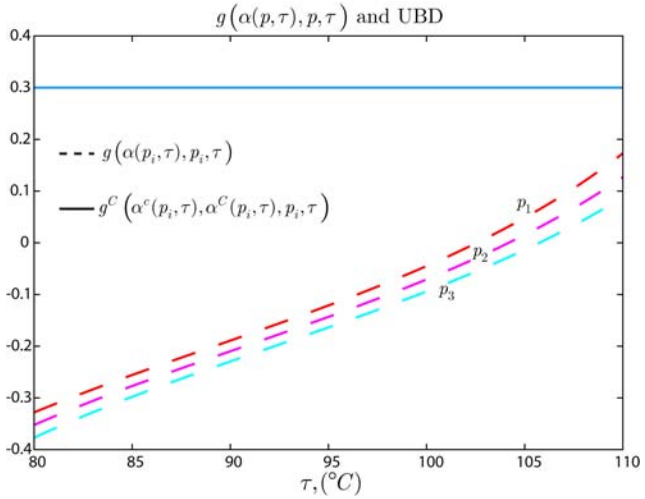


Figure 3 Implicit constraints and their corresponding concave relaxations for a partitioned uncertainty domain (see online version for colours)

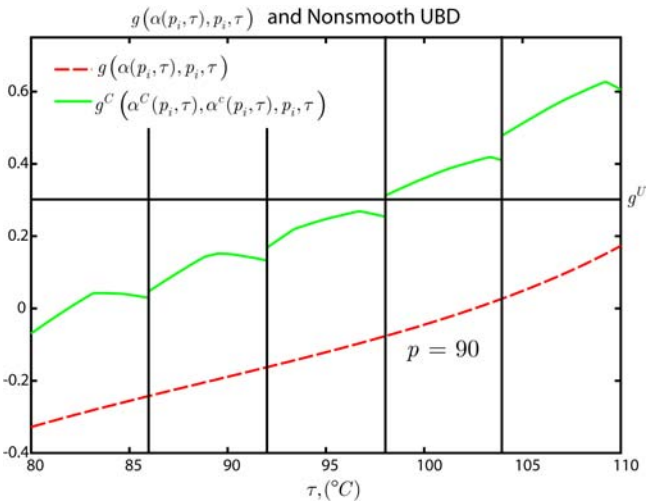


Table 4 Antoine constants for benzene and toluene (Elliot and Lira, 1999)

i	A_i	B_i	C_i
1 Tol.	6.95087	1342.31	219.187
2 Benz.	6.87987	1936.01	258.451

Figures 4 and 5 show the plots of the discrete implicit constraints g and the corresponding inclusion lower bound G^L *at* $P_1 = [90, 95]$ *and* $P_2 = [95, 100]$, *respectively. For this simple example, the SIP can be solved at the root node without branching. From the figures, it is clear that as* $\tau \to 110$ *we have the inclusion lower bounds* G_i^L *moving positive. Thus,* $\eta^* > 0$, *implying that for all realisations of uncertainty in the temperature within the interval T, there does not exist a pressure setting such that* $\alpha \leq 0.7$ *for all temperatures. However, if we change the performance constraint to* $\alpha \leq 0.9$, *we find* $\eta^* < 0$, *implying robust feasibility of the process.*

Figure 4 Implicit constraints and their corresponding inclusion lower bound (see online version for colours)

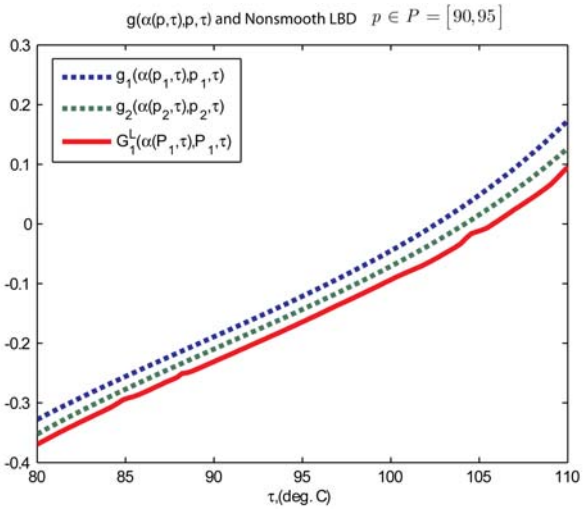
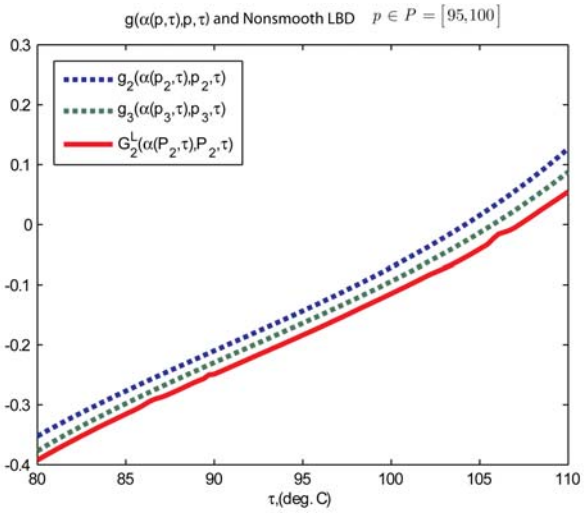


Figure 5 Implicit constraints and their corresponding inclusion lower bound (see online version for colours)



4 Conclusion

A model-based approach to designing process systems is required to guarantee the system will meet all performance specifications robustly, given various uncertainties in the process. The high level of complexity of such simulations warrants the need for a reduced-space approach. Solving steady-state model equations for process state variables as functions of the controls and uncertainty parameters offers a potentially large reduction in size, making robust simulation and design under uncertainty more tractable. However, this technique results in a semi-infinite optimisation problem in which the semi-infinite constraint is implicitly defined. The authors have shown how, with the application of the novel ideas of parametric interval Newton-type methods and McCormick-based relaxations of algorithms, the implicit SIP can be solved to ε-global optimality. The proposed rigorous method makes use of a LBP whose feasible region is a nonsmooth, nonconvex inner approximation to the SIP feasible region, and an UBP whose feasible region is a nonsmooth convex outer approximation of the SIP feasible region. The approximations are continually refined while branching in the decision domain. The process in question is said to be robust if the SIP solution value $\eta^* \leq 0$.

References

Balendra, S. and Bogle, I.D.L. (2009) 'Modular global optimization in chemical engineering', *Journal of Global Optimization*, Vol. 45, pp.169–185.

Ben-Tal, A. and Nemirovski, A. (2002) 'Robust optimization – methodology and applications', *Mathematical Programming*, Series B, Vol. 92, pp.453–480.

Bhattacharjee, B., Green Jr., W.H. and Barton, P.I. (2005a) 'Interval methods for semi-infinite programs', *Computational Optimization and Applications*, Vol. 30, pp.63–93.

Bhattacharjee, B., Lemonidis, P., Green Jr., W.H. and Barton, P.I. (2005b) 'Global solution of semi-infinite programs', *Mathematical Programming*, Series B, Vol. 103, pp.283–307.

Byrne, R.P. and Bogle, I.D.L. (2000) 'Global optimization of modular process flowsheets', *Industrial & Engineering Chemistry Research*, Vol. 39, pp.4296–4301.

Chachuat, B. (2007) *libMC: A Numeric Library for McCormick Relaxation of Factorable Functions*. Available online at: http://yoric.mit.edu/libMC/

Elliot, J.R. and Lira, C.T. (1999) *Introductory Chemical Engineering Thermodynamics*, Prentice-Hall, Englewood Cliffs, NJ.

Falk, J.E. and Hoffman, K. (1977) 'A nonconvex max–min problem', *Naval Research Logistics Quarterly*, Vol. 24, No. 3, pp.441–450.

Gill, P.E., Murray, W. and Saunders, M.A. (2005) 'SNOPT: an SQP algorithm for large-scale constrained optimization', *SIAM Review*, Vol. 47, No. 1, pp.99–131.

Halemane, K.P. and Grossmann, I.E. (1983) 'Optimal process design under uncertainty', *AIChE Journal*, Vol. 29, No. 3, pp.425–433.

Hansen, E. and Walster, G.W. (2004) *Global Optimization using Interval Analysis*, 2nd ed., Marcel Dekker, New York.

Hettich, R. and Kortanek, K.O. (1993) 'Semi-infinite programming: theory, methods, and applications', *SIAM Review*, Vol. 35, No. 3, pp.380–429.

Horst, R. and Tuy, H. (1997) *Global Optimization: Deterministic Approaches*, Springer-Verlag, New York.

Jaulin, L., Kieffer, M., Didrit, O. and Walter, E. (2001) *Applied Interval Analysis*, Springer-Verlag, London.

Kearfott, R.B. (1990) 'Preconditioners for the interval Gauss–Seidel method', *SIAM Journal on Numerical Analysis*, Vol. 27, No. 3, pp.804–822.

Kearfott, R.B. (1996) *Rigorous Global Search: Continuous Problems*, Kluwer Academic Publishers, Boston, MA.

King, J.C. (1980) *Separation Processes*, 2nd ed., McGraw-Hill, New York.

Mäkelä, M.M. (2001) 'Survey of bundle methods for nonsmooth optimization', *Optimization Methods and Software*, Vol. 17, No. 1, pp.1–29.

McCormick, G.P. (1976) 'Computability of global solutions to factorable nonconvex programs: part I – convex underestimating problems', *Mathematical Programming*, Vol. 10, pp.147–175.

Mitsos, A., Chachuat, B. and Barton, P.I. (2009) 'McCormick-based relaxations of algorithms', *SIAM Journal on Optimization*, Vol. 20, No. 2, pp.573–601.

Mitsos, A., Lemonidis, P., Lee, C.K. and Barton, P.I. (2008) 'Relaxation-based bounds for semi-infinite programs', *SIAM Journal on Optimization*, Vol. 19, No. 1, pp.77–113.

Moore, R.E. (1979) *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, PA.

Neumaier, A. (1990) *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge.

Ortega, J.M. and Rheinboldt, W.C. (1970) *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press Inc., Boston, MA.

Rump, S.M. (1999) 'INTLAB-INTerval LABoratory', in Csendes, T. (Ed.): *Developments in Reliable Computing*, Kluwer Academic Publishers, Dordrecht, pp.77–104.

Schnepper, C.A. and Stadtherr, M.A. (1996) 'Robust process simulation using interval methods', *Computers & Chemical Engineering*, Vol. 20, No. 2, pp.187–199.

Scott, J.K., Stuber, M.D. and Barton, P.I. (2011) 'Generalized McCormick relaxations', *Journal of Global Optimization*, DOI 10.1007/s10898-011-9664-7.

Stuber, M.D. and Barton, P.I. (2011) 'On parametric interval Newton methods', in preparation.

Stuber, M.D., Scott, J.K. and Barton, P.I. (2011) 'Global optimization of implicit functions', in preparation.

Zuhe, S., Neumaier, A. and Eiermann, M.C. (1990) 'Solving minimax problems by interval methods', *BIT*, Vol. 30, pp.742–751.