Worst-Case Design of Subsea Production Facilities Using Semi-Infinite Programming

Matthew D. Stuber, Achim Wechsung, Arul Sundaramoorthy, and Paul I. Barton
Process Systems Engineering Laboratory, Dept. of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139

DOI 10.1002/aic.14447
Published online April 3, 2014 in Wiley Online Library (wileyonlinelibrary.com)

The problem of designing novel process systems for deployment in extreme and hostile environments is addressed. Specifically, the process system of interest is a subsea production facility for ultra deepwater oil and gas production. The costs associated with operational failures in deepwater environments are prohibitively high and, therefore, warrant the application of worst-case design strategies. That is, prior to the construction and deployment of a process, a certificate of robust feasibility is obtained for the proposed design. The concept of worst-case design is addressed by formulating the design feasibility problem as a semi-infinite optimization problem with implicit functions embedded. A basic model of a subsea production facility is presented for a case study of rigorous performance and safety verification. Relying on recent advances in global optimization of implicit functions and semi-infinite programming, the design feasibility problem is solved, demonstrating that this approach is effective in addressing the problem of worst-case design of novel process systems. © 2014 American Institute of Chemical Engineers *AIChE J*, 60: 2513–2524, 2014
Keywords: robust design, design under uncertainty, verification, global optimization, semi-infinite programming

Introduction

As oil and gas reserves continue to be depleted from traditional on-land and shallow-water fields, there has been a significant effort made toward production from increasingly more hostile environments such as those in the ultra deepwater—greater than 7500 ft depths—of the Gulf of Mexico. In 2004, a vast deposit of petroleum, known as the "lower tertiary trend," containing 3–15 billion barrels of petroleum, was discovered by Chevron geologists.[1] However, as was demonstrated by BP in 2010 when it suffered a catastrophic failure of its leased ultra deepwater drilling platform—in only about 5000 ft of water—resulting in 11 lives lost and an estimated $30 billion in expenses and five million barrels spilled[2,3] with significant ecological damage, pursuing oil reserves in deepwater environments comes with inherently high risk magnified by a lack of sufficient technology. In this environment, the costs associated with operational failures far outweigh the costs associated with "overdesigning" the process, and so the goal must be to avoid them altogether.

Industry engineers have suggested that the application of traditional floating platforms to ultra deepwater production is too risky. Instead, novel remote compact subsea production facilities are considered a key enabling technology for ultra deepwater oil and gas production. Due to imprecise data and incomplete knowledge of the extreme subsea conditions, among various other factors, it is apparent that uncertainty must be accounted for. Thus, the task of designing such a process system is far from trivial.

Correspondence concerning this article should be addressed to P. I. Barton at pib@mit.edu.

As field conditions are extreme, they are difficult and expensive to recreate in the laboratory, and as building physical pilot plant systems for testing at field conditions is implausible, model-based design must support and complement empirical studies. Furthermore, it is worth mentioning that even if building and deploying pilot systems were a cost-effective approach, they can only be tested under a finite number of conditions, and therefore, no rigorous guarantee of worst-case performance/safety can be verified.

Stuber and Barton[4] previously stated that for these types of systems, the first question a design engineer must address is:

> Given a process model, and taking into account the 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/safety specifications?

This question will be formulated mathematically later and its application to subsea production facilities will be the primary focus of this article. In the following section, the subsea process system model will be presented and the case study will be set up.

Model and Case Study

The subsea separator is considered to be at the heart of subsea production facilities because it is the key process system for performing upstream phase separation as material is being produced from the wellhead. In the steady-state model presented here, it is considered that a three-phase mixture of oil/water/gas is being sufficiently separated to allow for reinjection of the water back into the environment and the production of separate oil and gas

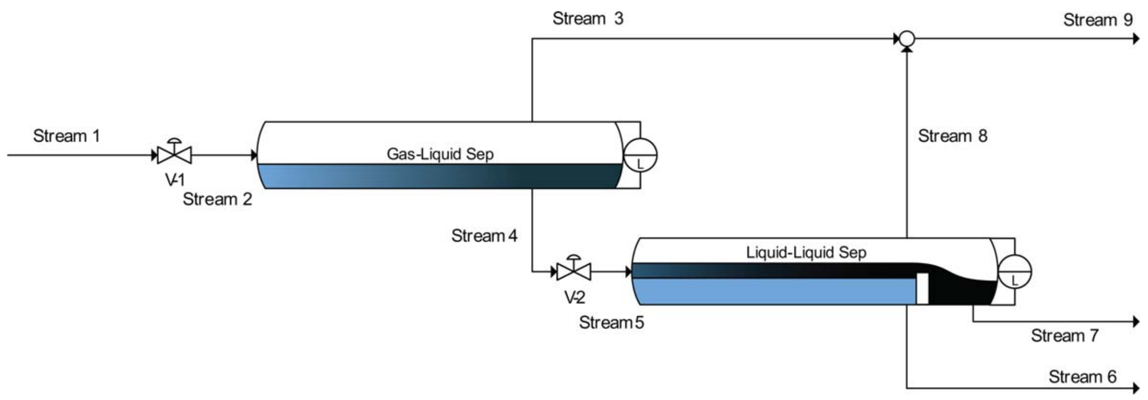

Figure 1. The process flow diagram of the subsea separation process.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

streams. It is assumed that sand has been separated from this stream prior to being fed to this separator. Figure 1 shows the process flow diagram of the subsea separator with some modeling details. The system consists of a gas–liquid separator (GLS), a liquid–liquid separator (LLS), two control valves, and a gas mixer.

Assumptions

Because the model is intended to illustrate the basic approach, there is a list of simplifying assumptions. However, as various levels of complexity may be added to this basic model, certain assumptions may be eliminated in the future. For the purposes of the basic model, the following assumptions are made:

1. Ideal homogeneous mixtures in multiphase streams and the GLS.
2. No liquid entrainment in the gas phase.
3. Perfect oil–water phase separation in the LLS.
4. Unrestricted flow from the LLS implying constant phase volumes.
5. Horizontal separator vessels are horizontal cylinders with flat end-caps.
6. Oil and water phases remain in a homogeneous emulsion with only the gas phase separating in the GLS.

Input parameters

The various physical properties of the system are specified by the user as input parameters. Tables A1–A4 in Appendix contain the parameters that are specified by the user. Table A5 in Appendix contains a description of the relevant symbols used herein.

The following calculation for the specific gravity of oil is used

$$\mathrm{SG_O}=\frac{141.5}{131.5+\mathrm{API}}$$

The specific gravity of the mixture at the wellhead is

$$\mathrm{SG_{mix}}=(\xi_{G1}/\mathrm{SG_G}+\xi_{W1}/\mathrm{SG_W}+\xi_{O1}/\mathrm{SG_O})^{-1}$$

where ξ_{ji} is the mass fraction of material j (O = oil, G = gas, W = water) in stream i.

Of course, the following constraint on the composition of the mixture at the wellhead must hold

$$\xi_{G1}+\xi_{W1}+\xi_{O1}=1$$

Control valve V-1

The variables associated with V-1 are

$$\dot m_1,\dot m_2,P_1,P_2,\xi_{G1},\xi_{O1},\xi_{W1},\xi_{G2},\xi_{O2},\xi_{W2}$$

where $\dot m_i$ is the mass flow rate of Stream i in kg/s, P_i is the pressure of Stream i in Pa. For simplicity, the vector of mass fractions can be expressed as $\boldsymbol{\xi}_i=(\xi_{Gi},\xi_{Wi},\xi_{Oi})$.

The model equations are

$$P_1=P_{\mathrm{well}}$$

which is specified by the wellhead

$$P_2=P_{\mathrm{GLS}}$$

which is the design pressure of the GLS, and

$$\boldsymbol{\xi}_1=\boldsymbol{\xi}_2$$

which is specified as the source of disturbance uncertainty. The mass flow rate through the valve is given by

$$\dot m_1=\dot m_2=u_1C_{v1}\sqrt{\frac{P_1-P_2}{\mathrm{SG_{mix}}}} \quad (1)$$

where u_1 is the control setting.

Gas–liquid separator

The variables associated with the GLS are

$$\dot m_2,\dot m_3,\dot m_4,P_2,P_3,P_4,H_{\mathrm{GLS}},\rho_4,V_{\mathrm{GLS}},\boldsymbol{\xi}_2,\boldsymbol{\xi}_3,\boldsymbol{\xi}_4$$

where H_{GLS} is the liquid level in the GLS in m, ρ_4 is the density of the mixture in Stream 4 in kg/m^3, V_{GLS} is the liquid volume in the GLS in m^3, and $\boldsymbol{\xi}_i$ is the vector of mass fractions corresponding to Stream i. The associated model equations are given below.

The pressure relationships are given by

$$\begin{gathered}P_3=P_2(\text{specified by design})\\ P_4=P_3+\rho_4 g_a H_{\mathrm{GLS}}\end{gathered} \quad (2)$$

The liquid volume in the GLS is given by

$$V_{\mathrm{GLS}}=L_{\mathrm{GLS}}\left((H_{\mathrm{GLS}}-R_{\mathrm{GLS}})\sqrt{2R_{\mathrm{GLS}}H_{\mathrm{GLS}}-H_{\mathrm{GLS}}^2}+R_{\mathrm{GLS}}^2\cos^{-1}\left[1-\frac{H_{\mathrm{GLS}}}{R_{\mathrm{GLS}}}\right]\right) \quad (3)$$

The mass balances are

$$\dot{m}_2 = \text{specified via wellhead}$$

$$\dot{m}_2 = \dot{m}_3 + \dot{m}_4$$

$$1 = \xi_{G4} + \xi_{W4} + \xi_{O4}$$

$$\xi_{G2}\dot{m}_2 = \xi_{G3}\dot{m}_3 + \xi_{G4}\dot{m}_4$$

$$\xi_{W2}\dot{m}_2 = \xi_{W4}\dot{m}_4 (\text{water balance})$$

$$\boldsymbol{\xi}_2 = \text{specified by wellhead composition}$$

$$\boldsymbol{\xi}_3 = (1,0,0)(\text{only gas in Stream 3})$$

where the density of Stream 4 is given by

$$\rho_4 = \rho^\circ_W (\xi_{G4}/\text{SG}_G + \xi_{W4}/\text{SG}_W + \xi_{O4}/\text{SG}_O)^{-1} \tag{4}$$

Last, a model describing the gas–liquid separation is needed. This model can be derived rather easily by noticing that the rate in which gas is separated from the mixture is governed by the velocity of the bubbles traveling upward and out of the mixture which is determined by the bubble size and the physical properties of the mixture. Here, the mean bubble size is assumed to be determined by the internal construction of the separator and the separator inlet conditions, and therefore, it is assumed that a performance constant for the separator can be assigned. Let ξ_G be the mass fraction of gas in the separator at any given point. Then, the rate of gas separation is

$$\frac{d\xi_G}{d\tau} = -k\tau$$

where k is the separator performance constant and τ is the residence time of the mixture in the separator. Solving gives us the GLS model which is a simple exponential decay

$$\xi_{G4} = \xi_{G2}\exp\left[-k_{GLS}\frac{V_{GLS}}{\dot{m}_4/\rho_4}\right] \tag{5}$$

where the separator performance constant is k_{GLS} and the quantity $V_{GLS}/(\dot{m}_4/\rho_4)$ is the residence time of the liquid solution in the GLS.

Control valve V-2

The control valve V-2 is almost identical to V-1. The associated variables are

$$P_4, P_5, \dot{m}_4, \dot{m}_5, \boldsymbol{\xi}_4, \boldsymbol{\xi}_5, \rho_4$$

The mass-balance equations are

$$\boldsymbol{\xi}_4 = \boldsymbol{\xi}_5$$

$$\dot{m}_4 = \dot{m}_5 = u_2 C_{v2}\sqrt{\frac{P_4 - P_5}{\rho_4/\rho^\circ_W}}$$

The outlet pressure—which is specified—is given by

$$P_5 = P_{LLS}$$

Liquid–liquid separator

The LLS is very similar to the GLS. One key difference is the liquid level in the LLS is specified assuming no restrictions on exit stream flow rates. The associated variables are

$$\dot{m}_5, \dot{m}_6, \dot{m}_7, \dot{m}_8, \boldsymbol{\xi}_5, \boldsymbol{\xi}_6, \boldsymbol{\xi}_7, \boldsymbol{\xi}_8, P_8, V_{LLS}, V_{oil}, H_{LLS}, \rho_7$$

where ρ_7 is the density of the solution in Stream 7 in kg/m^3, V_{LLS} is the total liquid volume in the LLS in m^3, H_{LLS} is the total liquid level in the LLS in m, and V_{oil} is the volume of just the oil/gas mixture in the LLS.

The liquid volume in the LLS is given by

$$V_{LLS} = L_{LLS}\left((H_{LLS} - R_{LLS})\sqrt{2R_{LLS}H_{LLS} - H^2_{LLS}} + R^2_{LLS}\cos^{-1}\left[1 - \frac{H_{LLS}}{R_{LLS}}\right]\right) \tag{6}$$

where the liquid height H_{LLS} is specified. The volume of the oil/gas mixture phase in the separator is given by the following relationship assuming an ideal mixture

$$V_{oil} = V_{LLS}\left(\frac{\dot{m}_7}{\rho_7}\frac{\rho_5}{\dot{m}_5}\right) \tag{7}$$

The quantity $\dot{m}_7\rho_5/\rho_7\dot{m}_5$ is the volume fraction of the combined oil and gas exiting in Stream 7 with respect to the total solution incoming in Stream 5. The product with V_{LLS} is, therefore, the volume of the oil/gas solution for which further gas–liquid separation is taking place.

The mass-balance equations are

$$\boldsymbol{\xi}_8 = (1,0,0)(\text{only gas in Stream 8})$$

$$\boldsymbol{\xi}_6 = (0,1,0)(\text{only water in Stream 6})$$

$$1 = \xi_{G7} + \xi_{O7}(\text{only gas and oil in Stream 7})$$

$$\dot{m}_5 = \dot{m}_6 + \dot{m}_7 + \dot{m}_8$$

$$\xi_{G5}\dot{m}_5 = \xi_{G8}\dot{m}_8 + \xi_{G7}\dot{m}_7 \text{ (gas balance)}$$

$$\xi_{W5}\dot{m}_5 = \xi_{W6}\dot{m}_6 \text{ (water balance)}$$

$$\xi_{W7} = 0 \text{ (no water in Stream 7)}$$

Further gas–liquid separation is again being modeled as a simple exponential decay

$$\xi_{G7} = \xi_{G5}\exp\left[-k_{LLS}\frac{V_{oil}}{\dot{m}_7/\rho_7}\right] \tag{8}$$

where, again, k_{LLS} is a separator performance constant, and the quantity $V_{oil}/\dot{m}_7\rho_7$ is a residence time of the gas/oil mixture in the separator.

The density of the oil/gas mixture stream is given by

$$\rho_7 = \rho^\circ_W (\xi_{G7}/\text{SG}_G + \xi_{O7}/\text{SG}_O)^{-1}$$

Since the liquid outlet streams have no restrictions to flow, their flow-pressure relationships can be ignored. The pressure in the gas Stream 8 is specified

$$P_8 = P_{LLS}$$

Gas mixer

The gas mixer is simply a junction of two pure gas streams. The associated variables are

$$\boldsymbol{\xi}_8, \boldsymbol{\xi}_3, \boldsymbol{\xi}_9, \dot{m}_3, \dot{m}_8, \dot{m}_9, P_3, P_8, P_9$$

The associated equations are

$$\boldsymbol{\xi}_3 = \boldsymbol{\xi}_9$$

$$\dot{m}_9 = \dot{m}_3 + \dot{m}_8$$

$$P_9 = \min\{P_3, P_8\}$$

Pointwise numerical simulation

Pointwise numerical simulation was performed using a Windows 7 machine with Intel Core2 Quad Q9450 CPU to study the behavior of the model over a range of uncertainty

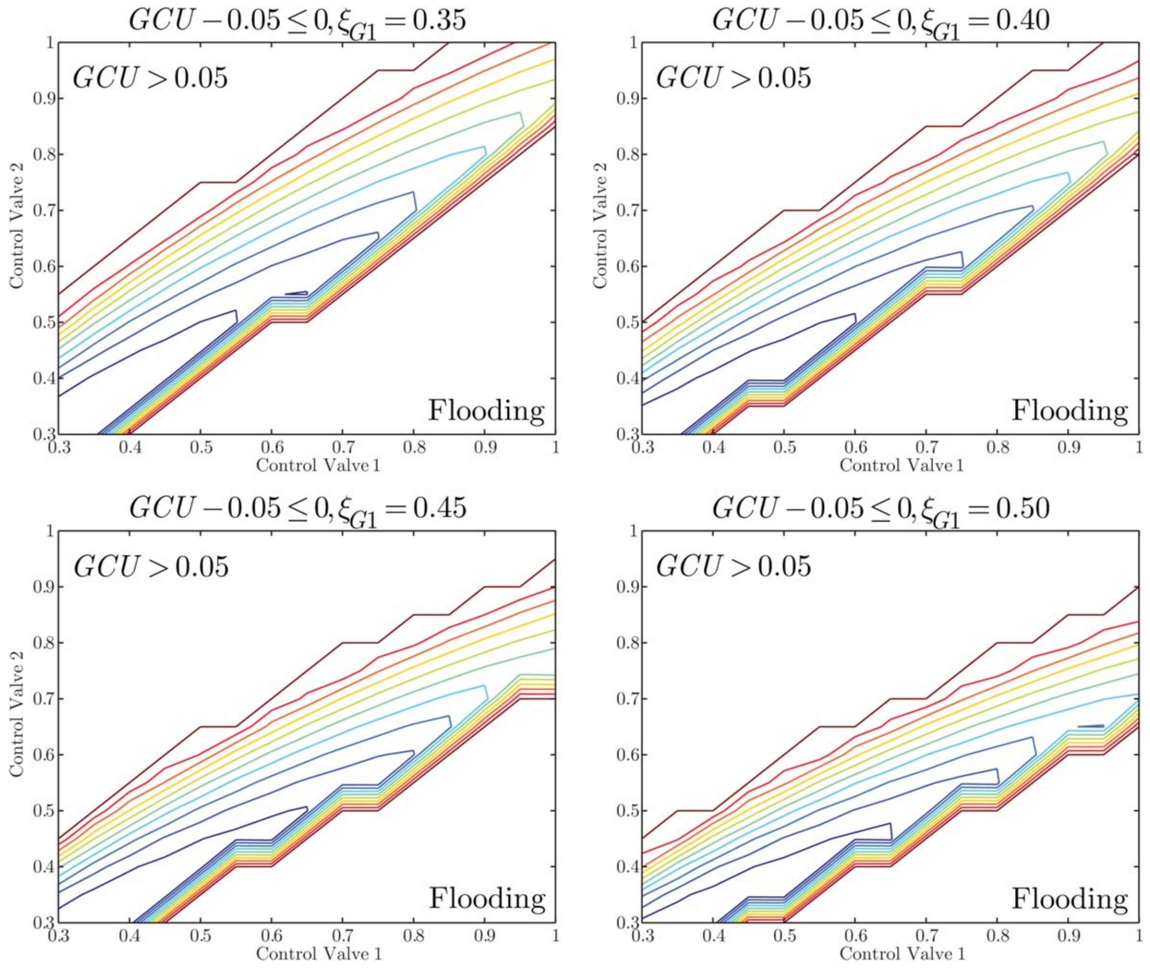

Figure 2. The results of the coarse-grained numerical simulation of the subsea separator model.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

parameter and control values using the JACOBIAN® process simulator.[5] It should be noted that the min operator appearing in the gas mixer model is nonsmooth. This does not introduce any problems for the JACOBIAN solver, which can handle nonsmoothness.

Performing pointwise simulation of the subsea separator system resulted in a very coarse-grain view of the system under varying input conditions and control actions. For this study, the mass fractions of the gas (ξ_{G1}) and water (ξ_{W1}) in the input stream were treated as uncertain while the oil fraction (ξ_{O1}) was held constant. This may be interpreted as a gas bubble being produced from the wellhead. The gas fractions of 0.35, 0.4, 0.45, and 0.5 were studied while the oil fraction was set to 0.4. The dimensions of the GLS were such that $R_{GLS} = 0.6$ m and $L_{GLS} = 5$ m. The control values were varied between 30% open and fully opened positions in 5% increments. A constraint was imposed that the gas carry-under (GCU)—which is the fraction of gas that is entrained in the oil product stream—must be less than, or equal to, 5%. There was also an inherent physical constraint regarding the liquid volume in the GLS such that it could not be greater than the total volume of the vessel.

In Figure 2, the coarse-grain view of the feasible region at each fixed uncertainty realization of the gas fraction of the incoming stream is shown. Each contour plot is the result of running 196 steady-state simulations, taking a total time of approximately 45 s for each gas fraction. As can be seen from Figure 2, for the four realizations of uncertainty simulated, there exists a fairly large set of controls that ensure feasible operation of the process. The contours do not necessarily have any specific relevance for the purposes of this article as the concern is simply on whether the performance constraint is satisfied. However, for completeness, the contours are included and they correspond with the level at which the constraint on the GCU is satisfied. The outer borders correspond with the constraint on the GCU just being satisfied, whereas the inner-most contour corresponds with the least amount of GCU. The region in the upper-left corner of the plots corresponds with the region where the GCU is too high (and the performance constraint is not satisfied). The region in the lower-right corner of the plots corresponds with the regime in which the GLS is flooded. The reader will also notice that the contours appear to be piecewise affine, however, this is simply an artifact of discretization. At the expense of increased computational effort, a finer discretization would produce contours that appear smoother.

By analyzing the results of the pointwise numerical simulation, it becomes apparent that it is insufficient in addressing the original question posed in the Introduction. First, identifying feasible control settings for a particular uncertainty realization may actually depend on the resolution of the discretization mesh applied to the control space for the simulation. Second, and much more importantly, there are infinitely many realizations of uncertainty in the given uncertainty interval. Therefore, pointwise numerical simulation is incapable of identifying the worst-case realization with any sort of guarantee. In this case, only a rigorous optimization-based approach can be taken. In the next section, this worst-case design problem is formulated mathematically as an optimization problem which, on solving, has the ability to determine with mathematical certainty whether or not the process is robustly feasible (i.e., there exists at least one control setting such that the performance constraints may be satisfied in the face of the worst-case realization of uncertainty).

Problem Formulation

The steady-state design question presented in the introduction section, and discussed in the previous section, can be stated equivalently as the logical feasibility statement[6]

$$\forall \mathbf{p} \in P, \exists \mathbf{u} \in U : g(\mathbf{z}, \mathbf{u}, \mathbf{p}) \leq 0, \mathbf{h}(\mathbf{z}, \mathbf{u}, \mathbf{p}) = \mathbf{0} \tag{9}$$

Here, $\mathbf{h} : X \times U \times P \to \mathbb{R}^{n_x}$ represents the steady-state process model equations presented in the previous section. The performance/safety specification or constraint will be represented by $g : X \times U \times P \to \mathbb{R}$ which, in the previous section, was the performance specification on the GCU. The variables $\mathbf{z} \in X \subset \mathbb{R}^{n_x}, \mathbf{u} \in U \subset \mathbb{R}^{n_u}, \mathbf{p} \in P \subset \mathbb{R}^{n_p}$ will be the process state variables, controls, and uncertainty parameters, respectively. The set X will represent rigorous bounds on the process state variables which oftentimes initially come from physical information (e.g., density is nonnegative and has a reasonable upper-bound for solids and liquids, etc.). The set U will represent rigorous bounds on the control actions (e.g., control valves can only position themselves between fully opened and fully closed positions). Last, the set P will represent the uncertainty set (e.g., the mass fraction of gas in the feed which takes values between bounds). Thus, in words Eq. 9 reads:

For all values of the uncertain parameters, there exists a feasible control action such that the process model and the safety specification are satisfied.

Now, suppose unique $\mathbf{z} \in X$ exist that satisfy $\mathbf{h}(\mathbf{z}, \mathbf{u}, \mathbf{p}) = \mathbf{0}$ at each $(\mathbf{u}, \mathbf{p}) \in U \times P$, with U and P as intervals, then they define an implicit function of $\mathbf{u}$ and $\mathbf{p}$ that will be expressed as $\mathbf{x}(\mathbf{u}, \mathbf{p})$. It will be assumed that there exists a unique continuously differentiable implicit function $\mathbf{x} : U \times P \to X$ such that $\mathbf{h}(\mathbf{x}(\mathbf{u}, \mathbf{p}), \mathbf{u}, \mathbf{p}) = \mathbf{0}$ holds for every $(\mathbf{u}, \mathbf{p}) \in U \times P$. Conditions which satisfy this assumption are stated in the literature,[4,7,8] (Stuber et al., Submitted). In the previous section, a pointwise numerical simulation was used to evaluate $\mathbf{x}$ at given values of $\mathbf{u}$ and $\mathbf{p}$.

In the articles by Halemane and Grossman[6] and by Swaney and Grossman,[9] the authors take the same steps presented here to eliminate the model equations by introducing an implicit function of the controls and uncertainty parameters. However, in the mentioned articles, it is unclear if it is required that the model equations lead to implicit functions that have known closed forms. It seems that this must be the case as certain convexity properties of g are required.[6,9] However, in the general case, $\mathbf{x}$ does not have a known closed form and can only be approximated using iterative techniques. Furthermore, convexity of g cannot be assumed for the general case as the required properties of $\mathbf{x}$ cannot necessarily be guaranteed. In any case, the authors formulate the feasibility statement (9) as a max-min problem, which they mention is "very difficult to solve"[6,9]; in the general case it is $\mathcal{NP}$-hard. Ostrovsky et al.[10] presented a method for bounding the solution of the feasibility problem of Halemane and Grossman[6] and Swaney and Grossman[9] that amounts to solving a sequence of nonlinear programming subproblems. However, the same convexity requirements are maintained.[10]

Stuber and Barton[4] presented the nonconvex max-min problem discussed previously.[6,9] They reformulate the problem as a semi-infinite program (SIP) and solve it using an algorithm developed from the work of Bhattacharjee et al.[11] For more complicated models, this algorithm proves to be ineffective. One major drawback of that algorithm is that the number of constraints used to formulate the lower-bounding problem (LBP) grows exponentially with the depth in the branch-and-bound tree. Furthermore, because the effectiveness of the LBP relies primarily on the relative tightness of the interval bounds on the semi-infinite constraint, the algorithm often creates a branch-and-bound tree with significant depth.[11] For this reason, a heuristic for discarding irrelevant constraints was developed by Bhattacharjee et al.[11] However, the authors mention that the heuristic does not accelerate convergence; it only reduces the computational storage requirement.[11] For these reasons, the algorithm presented by Stuber and Barton[4] will be inefficient and potentially inadequate to address the feasibility of complex process systems examples, such as subsea production facilities. Falk and Hoffman[12] presented a robust design problem formulated as a nonconvex max-min problem, which they reformulate as a nonconvex SIP and solve using the algorithm developed by Blankenship and Falk.[13] A refinement of the same technique[14] will be applied in this work.

The feasibility statement (9) can be written equivalently as the following SIP[4]

$$\begin{aligned} \eta^* = &\max_{\mathbf{p} \in P, \eta \in \mathbb{R}} \eta \\ &\text{s.t. } \eta \leq g(\mathbf{x}(\mathbf{u}, \mathbf{p}), \mathbf{u}, \mathbf{p}), \forall \mathbf{u} \in U \end{aligned} \tag{10}$$

which is referred to as an implicit SIP because the semi-infinite constraint g is an implicit function of the controls and uncertainty parameters. Solving (10) pertaining to the subsea production facility model will verify performance/safety in the face of the worst-case realization of uncertainty. If $\eta^* \leq 0$, the design is feasible and worst-case performance/safety has been verified. Alternatively, if $\eta^* > 0$, the design is said to be infeasible.

Mitsos[14] recently refined the cutting-plane algorithm of Blankenship and Falk.[13] In particular, a novel upper-bounding procedure (for SIPs formulated as minimization problems) was added to guarantee—under some relatively mild assumptions—that the algorithm can produce an SIP-feasible point after finitely many iterations.[14] In effect, this guarantees that the global extremum value can be bounded rigorously and the SIP can be solved to ϵ-optimality in finitely many iterations of the algorithm.[14] For the upper-bounding problem (UBP), a restriction parameter $\epsilon^g > 0$ was introduced, effectively restricting the SIP-feasible set.[14] Another parameter $r > 1$ was introduced that effectively relaxes the restricted feasible set at certain stages in the SIP cutting-plane algorithm, effectively increasing the size of the originally restricted SIP-feasible set. All other details of the UBP are identical to the LBP (e.g., how cutting planes are chosen, etc.).

Stuber[15] adapted the SIP cutting-plane algorithm[14] to solve SIPs with implicit functions embedded. The application to max-min and min-max problems is also considered by explicitly reformulating them as SIPs with implicit functions embedded.[15] Two chemical engineering design examples were also given which demonstrate the effectiveness of this algorithm. In the next section, the application of this algorithm to the worst-case design problem is addressed.

Solution Method

SIP algorithm

In the previous section, it was stated that the SIP algorithm of Stuber,[15] which is an adaptation of the SIP cutting-plane algorithm[14] to implicit SIPs, will be utilized in this article to solve the problem of worst-case design of subsea production facilities. In the previous works,[15] various features and behaviors of the implicit SIP algorithm were explored using relatively simple examples. In this work, the implicit SIP algorithm will be demonstrated on a more complicated process model, which poses many numerical difficulties, motivated by a current challenge to engineers in the oil and gas sector. The numerical experiments will demonstrate that the implicit SIP algorithm is not only applicable to more complicated models but also that it is effective in providing a rigorous answer in relatively little time.

The reader should note that in the SIP literature,[14,15] the algorithm was written with respect to solving minimization problems, whereas the standard worst-case problem formulation is a maximization problem. The maximization SIP (10) can be transformed into a minimization problem by simply distributing minus signs appropriately. Then, one just needs to take care and notice that the UBP of the previous works,[14,15] which may provide SIP-feasible points and in turn a rigorous upper bound for the minimization problem, actually provides a rigorous upper bound on $-\eta$ (i.e., its negative is a rigorous lower bound on η). Similarly, the LBP of the previous works[14,15] will provide a rigorous lower bound on $-\eta$ (i.e., its negative is a rigorous upper bound on η). The three subproblems are stated explicitly as follows.

The LBP is based on a simple relaxation technique whereby the original SIP is reduced to an implicit nonlinear program (NLP) by only considering finitely many constraints corresponding to realizations of $\mathbf{u} \in U^{\text{LBP}}$ with $U^{\text{LBP}} \subset U$ as a finite set. The LBP is formulated as

$$\begin{aligned} f^{\text{LBP}} &= \min_{\mathbf{p}\in P, \eta\in\mathbb{R}} -\eta \\ \text{s.t. } & \eta - g(\mathbf{x}(\mathbf{u},\mathbf{p}),\mathbf{u},\mathbf{p}) \le 0, \forall \mathbf{u} \in U^{\text{LBP}} \end{aligned} \tag{11}$$

To guarantee that f^{LBP} is a valid bound, (11) must be solved to global optimality.

The inner problem is equivalent to the semi-infinite constraint and defines the SIP feasible region. Given a candidate $(\overline{\mathbf{p}},\overline{\eta})$, feasibility can be determined by solving the inner problem

$$\overline{\gamma}(\overline{\mathbf{p}},\overline{\eta}) = \max_{\mathbf{u}\in U} [\overline{\eta} - g(\mathbf{x}(\mathbf{u},\overline{\mathbf{p}}),\mathbf{u},\overline{\mathbf{p}})] \tag{12}$$

If $\overline{\gamma}(\overline{\mathbf{p}},\overline{\eta}) \le 0$, then $(\overline{\mathbf{p}},\overline{\eta})$ is an SIP-feasible point. In general, (12) must be solved to global optimality.

Similar to the LBP, for the UBP the original SIP is reduced to an implicit NLP by only considering finitely many constraints corresponding to realizations of $\mathbf{u} \in U^{\text{UBP}}$ with $U^{\text{UBP}} \subset U$ as a finite set. Furthermore, the restriction parameter $\epsilon^{g,k} > 0$ is introduced that bounds the semi-infinite constraint away from zero. The UBP is formulated as

$$\begin{aligned} f^{\text{UBP}} &= \min_{\mathbf{p}\in P, \eta\in\mathbb{R}} -\eta \\ \text{s.t. } & \eta - g(\mathbf{x}(\mathbf{u},\mathbf{p}),\mathbf{u},\mathbf{p}) \le -\epsilon^{g,k}, \forall \mathbf{u} \in U^{\text{UBP}} \end{aligned} \tag{13}$$

In order for the algorithm to solve the SIP to global optimality, (13) must be solved to global optimality. However, solving (13) locally will yield a valid bound on the solution.

It should also be noted that, since the worst-case design problem is a feasibility problem, it is only necessary to obtain a rigorous guarantee of feasibility or infeasibility. That is, it is unnecessary to solve (10) to global optimality if an SIP-feasible point can be found that provides a rigorous lower bound (LBD) on the objective function η such that LBD>0; implying $\eta^* > 0$. In this case, the algorithm can terminate with a guarantee of infeasibility. Similarly, if a rigorous upper bound (UBD) on the objective function can be obtained such that UBD ≤ 0, this implies $\eta^* \le 0$. In which case, the algorithm can terminate with a guarantee of robust feasibility. With regards to the optimization algorithm subproblems (11) and (13), the relationship is

$$\text{LBD} = -f^{\text{UBP}} \le \eta^* \le -f^{\text{LBP}} = \text{UBD}$$

These two additional termination criteria make the algorithm drastically more efficient since solving the SIP to global optimality is quite expensive. However, to guarantee that the algorithm terminates after finitely many iterations, the standard ϵ-optimality termination criterion remains present. If, however, the algorithm terminates with ϵ-optimality, further investigation is required to determine robust feasibility of the design with mathematical rigor. These termination criteria are identical to those of the algorithm in Stuber and Barton,[4] labeled 3(d) and 4(d), respectively, which was written with reference to solving the maximization problem.

As the SIP algorithm by Stuber[15] is to be applied here, it is required that global optimization subproblems, with embedded implicit functions, can be solved. Currently, the only algorithm available for doing so, in the general case, is that of Stuber et al. (Submitted).[15] The algorithm of Stuber et al. (Submitted)[15] uses the conventional branch-and-bound[16] (B&B) framework for global optimization. To summarize, the B&B algorithm iteratively subdivides the search space and sequentially solves UBP and LBP on each subdivision. The LBP is formulated by convexifying the original problem (say by McCormick relaxations,[17] αBB,[18] etc.) and solving the convex problem with a local optimization algorithm, such as sequential-quadratic programming.[19] The upper bound can either be obtained by simply evaluating the objective function at a candidate point located within the current subdivision or by solving the original nonconvex optimization problem using a local solver. By comparing the bounds calculated on each subdivision, regions of the search space can be discarded as they are guaranteed not to contain a global optimal solution (if any exist). The iterative procedure continues until the upper and lower bounds are sufficiently close at which time the algorithm is said to have converged.

Since the problems being solved here involve implicit functions, new methods for constructing convex and concave relaxations (and subgradients) were developed (Stuber et al., Submitted) (i.e., $\mathbf{x}^c$ convex and $\mathbf{x}^C$ concave such that $\mathbf{x}^c(\mathbf{u},\mathbf{p}) \le \mathbf{x}(\mathbf{u},\mathbf{p}) \le \mathbf{x}^C(\mathbf{u},\mathbf{p}), \forall(\mathbf{u},\mathbf{p}) \in U \times P$). These methods are analogous in many ways to parametric interval-Newton-type methods[8] for constructing interval bounds on implicit functions; however, they make use of generalized McCormick Relaxations[20] to construct convex and concave relaxations of these functions. First, interval bounds are calculated, say by interval-Newton-type methods. Then, the relaxation technique iteratively refines these bounds yielding convex lower-bounding and concave upper-bounding functions. Since convex and concave relaxations of implicit functions can be calculated, they can be composed with g using generalized McCormick relaxations[20] and the traditional lower-bounding subproblem of the B&B framework can be

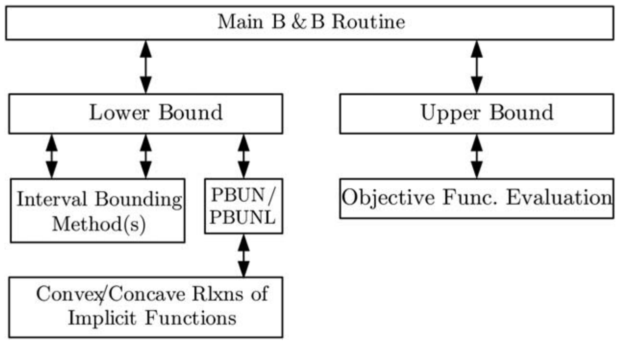
Main B & B Routine
Lower Bound
Upper Bound
Interval Bounding Method(s)
PBUN/ PBUNL
Objective Func. Evaluation
Convex/Concave Rlxns of Implicit Functions

Figure 3. Global optimization of implicit functions.

solved, using a local NLP solver, which supplies the B&B global optimization algorithm with a valid lower bound. The B&B algorithm simply treats (11) and (13) as constrained nonconvex NLPs and (12) as an unconstrained nonconvex NLP. The hierarchy for the information flow of the algorithm for global optimization of implicit functions, used to solve Eqs. 11–13, is shown in Figure 3.

The algorithm for solving implicit SIPs was implemented using C++ in a manner analogous to Stuber[15] with the additional stopping criteria added to the algorithm. The flowchart for the algorithm is shown in Figure 4. All other details of the algorithm are identical to the previous works.[15] In particular, the global optimization subproblems are solved using the global optimization of implicit functions algorithm (Stuber et al., Submitted)[15] previously discussed. PBUNS and PBUNL[21] are again used to solve nonsmooth convex problems for the global optimization of implicit functions algorithm. Similar to the previous works,[15] to circumvent the limitations imposed by PBUNL (i.e., only accepting affine inequality constraints), affine relaxations of convex nonlinear inequality constraints with respect to multiple reference points are used.

As a first attempt, the parametric interval-Newton method[4,8] was applied to calculate the required interval bounding information. It was determined that this was insufficient in supplying the required bounding information for this problem due to the behavior of the model equations. To remedy this, another interval technique, which falls under the constraint propagation category, was identified as being quite effective when used in conjunction with the parametric interval-Newton method.

Constraint propagation techniques

In solving SIPs with implicit functions embedded, such as (10), it is required to obtain meaningful bounding information for the function g, which is often a limiting step in the overall performance of the algorithm. This is because all that is known about the state variables initially are their natural bounds, which in turn leads to a prohibitively large initial bound on g from which no meaningful information can be deduced. Because interval-Newton-type methods[8] often prove to be ineffective in refining sufficiently large initial intervals, which was observed when applied to the separator model, this poses a serious problem for the algorithm. Although interval-Newton methods are quite effective on smaller intervals, a method that can obtain meaningful bounds on the function g starting with large initial intervals on the state variables efficiently is necessary for the overall success and performance of the algorithm.

Interval analysis has been widely applied to many simulation and optimization applications in chemical engineering, for example, the work by Lin and Stadtherr[22] and the work by Balendra and Bogle.[23] Lin and Stadtherr[22] presented strategies for bounding the solution of interval linear systems which were solved in the context of the interval-Newton method. The authors reviewed several preconditioning techniques for the above mentioned method and proposed a new bounding approach based on the use of linear programming (LP) strategies. They demonstrated the performance of the proposed technique on global optimization problems such as parameter estimation and molecular modeling. Balendra and Bogle[23] addressed interval-based global optimization of modular process design. In their work, the authors explored the use of five different interval contraction methods to improve the performance of a previous interval optimization algorithm.[24] The contraction methods used were: consistency techniques, constraint propagation, LP contractors, interval Gaussian elimination, and the interval-Newton contractor. Using a set of mathematical problems and chemical engineering flow sheet design problems such as the Haverly pooling problem, reactor flow sheet problem, and a reactor network problem, they compared the impact of various contraction methods on the overall performance of the interval optimization algorithm. Their computational experiments showed that the LP contractors performed the best while the constraint propagation and interval Gaussian elimination methods were ineffective.

In the context of interval contraction, there exist several methods developed by researchers outside of the process systems community. For a detailed review of such methods, see the book by Jaulin et al.[25] Schichl and Neumaier[26] presented the fundamentals of interval analysis on directed acyclic graphs

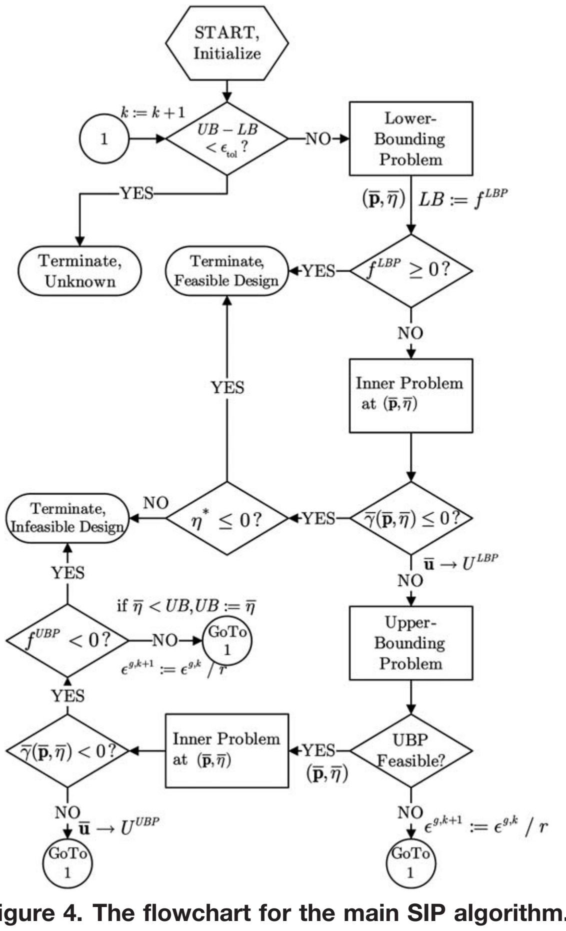
START, Initialize
$k := k+1$
1
$UB - LB < \epsilon_{tol}$?
NO
YES
Lower-Bounding Problem
$(\bar{\mathbf{p}}, \bar{\eta})$ $LB := f^{LBP}$
Terminate, Unknown
Terminate, Feasible Design
YES
$f^{LBP} \geq 0$?
NO
YES
Inner Problem at $(\bar{\mathbf{p}}, \bar{\eta})$
Terminate, Infeasible Design
NO
$\eta^* \leq 0$?
YES
$\bar{\gamma}(\bar{\mathbf{p}}, \bar{\eta}) \leq 0$?
$\bar{\mathbf{u}} \to U^{LBP}$
NO
YES
if $\bar{\eta} < UB, UB := \bar{\eta}$
$f^{UBP} < 0$?
NO
GoTo 1
$\epsilon^{g,k+1} := \epsilon^{g,k} / r$
Upper-Bounding Problem
YES
$\bar{\gamma}(\bar{\mathbf{p}}, \bar{\eta}) < 0$?
Inner Problem at $(\bar{\mathbf{p}}, \bar{\eta})$
YES
$(\bar{\mathbf{p}}, \bar{\eta})$
UBP Feasible?
NO
$\bar{\mathbf{u}} \to U^{UBP}$
GoTo 1
NO
$\epsilon^{g,k+1} := \epsilon^{g,k} / r$
GoTo 1

Figure 4. The flowchart for the main SIP algorithm.

(DAGs) for global optimization and constraint propagation. The proposed framework overcomes the limitation of propagating each constraint individually by taking into account the effects of any common subexpressions that are shared by many constraints. Later, the above framework was extended to perform adaptively forward evaluations and backward projections on only some select nodes of a DAG.[27] The computational study showed that the adaptive framework performs at least one to two orders of magnitude faster than the other state-of-the-art interval constraint propagation techniques.

More recently, the adaptive DAG framework of Vu et al.[27] was used in a branch-and-prune algorithm to find multiple steady states in homogeneous azeotropic and ideal two-product distillation problems.[28] Their computational experiments showed some promising results in the application of constraint propagation techniques of Vu et al.[27]

In this work, a forward-backward constraint propagation technique, similar to the DAG framework of Schichl and Neumaier,[26] will be discussed and exploited. The technique is used to obtain meaningful bounds on the implicit functions of (10). Thus, the goal is to expedite the above bounding procedure over a given large initial box using the constraint propagation technique, and subsequently obtain rigorous, tight, and convergent bounds on implicit functions using the interval-Newton method. Combining the strengths of constraint propagation and interval-Newton methods seem to be a promising approach to obtaining useful bounding information required for solving (10), and this will be the focus of the proposed solution framework.

Forward and backward propagation of intervals

Different interval arithmetic implementations have been developed in the past, for example, for C++.[29,30] These provide a new data type and use operator and function overloading to calculate interval extensions of arithmetic expressions. They can be easily used for the forward interval evaluation of an explicit factorable function. However, to enable the backward propagation of intervals, it is necessary to keep information about intermediate factors in memory [a similar requirement is found in the reverse mode of automatic differentiation (AD)[31]]. There, a record of each operation is kept on a so-called tape during a forward function evaluation. During the reverse pass, the tape is read to reconstruct the operation and calculate the derivative. The stored information includes the type of operation and the address of the operands in the tape.

Here, for the implementation of a backward interval propagation, it is proposed to proceed in a slightly different fashion. First, the factorable function is parsed using operator and function overloading to construct its computational graph. All other operations will be performed on this graph object. In contrast to typical AD implementations, the computational graph, which can be thought of as a kind of tape, is persistent in memory and can be reused after it is constructed once. Basically, the graph is stored in an array where each element, or factor, contains information about the type of operation and the address of the operands. In addition, for each factor, an interval is also stored. These intervals can be accessed for the independent and dependent variables, that is, variables and function values, respectively. Also, it is possible to provide *a priori* bounds on some specific intermediate factors, if desired.

Forward Interval Evaluation. Prior to the forward interval evaluation, an interval is specified for each independent variable. Also, *a priori* intervals can be specified for intermediate factors. Then, during the forward evaluation, the graph can be traversed element-by-element and an inclusion interval can be constructed for each factor according to its operation type as each factor depends only on factors that have been evaluated already and for which this inclusion information is already available. If a node is an intermediate factor for which *a priori* bounds have been provided, then these bounds are intersected with the newly calculated interval so as to provide potentially tighter bounds. If the intersection is empty, then this bound cannot be satisfied for all possible realizations of the independent variables. Once all factors have been calculated, the inclusion intervals of the dependent variables, that is, the function values, are exported from the graph object.

Backward Interval Propagation. After a forward interval evaluation has provided valid bounds on each factor, the intervals for the dependent variables can be updated by intersecting these with additional information such as constraints that must be satisfied. Then, the computational graph will be traversed in reverse order. For example, suppose that the current factor is $v_k = v_i + v_j$. Then, it also must be true that $v_i = v_k - v_j$ and $v_j = v_k - v_i$. Analogous rules can be constructed for other operations too, where more discussion can be found in the work by Schichl and Neumaier.[26] This provides additional bounds on the operands v_i and v_j, which can be intersected with their current bounds resulting in potentially tighter bounds. Again, element after element is revisited until the first factor, that is not an independent variable, is reached. If an intersection resulted in an empty interval, then one can conclude that no possible realizations of the independent variables on the original box can satisfy the constraints. Otherwise, potentially tighter intervals have been computed for the independent variables. Note that the backward propagation can be done in several different orders, each of which may generate a different result.

It is possible to perform multiple forward evaluations and backward propagation passes consecutively as the computed intervals do not necessarily converge to fixed point intervals in a single iteration. The following example illustrates the forward evaluation and backward propagation steps.

Example 1. *Consider*

$$f(z,p) = z^2 + zp + 4, X^0 = [-0.8, -0.3], P^0 = [6, 9]$$

A factorable representation is given by

$$v_1 = z^2$$

$$v_2 = zp$$

$$v_3 = v_1 + v_2$$

$$v_4 = v_3 + 4$$

Forward interval evaluation results in

$$V_1 = [-0.8, -0.3]^2 = [0.09, 0.64]$$

$$V_2 = [-0.8, -0.3] \cdot [6, 9] = [-7.2, -1.8]$$

$$V_3 = [0.09, 0.64] + [-7.2, -1.8] = [-7.11, -1.16]$$

$$V_4 = [-7.11, -1.16] + [4, 4] = [-3.11, 2.84]$$

Prior to the backward pass, we set V_4 = [0, 0], which corresponds to f(z,p) = 0. First, we update V_3 according to $V_3 := V_3 \cap (V_4 - 4) = [-4, -4]$. *Next, we reverse the* *assignment* $V_3=V_1+V_2$ *to update* V_1 *and* V_2: $V_1 := V_1 \cap (V_3 - V_2) = [0.09, 0.64], V_2 := \quad V_2 \cap (V_3 - V_1) = [-4.64, -4.09]$. *Then,* $V_2 = X \cdot P$ *is reversed:* $X := X^0 \cap (V_2/P^0\ ) = [-0.7734, -0.4544], P := P^0 \cap (V_2/X) = [6, 9]$. *Last,* $V_1 = X^2$ *is reversed:* $X := \mathrm{hull}\{X \cap -\sqrt{V_1}, X \cap \sqrt{V_1}\} = [-0.7734, -0.4544]$, *which concludes the backward interval propagation. As a result,* $X = [-0.7734, -0.4544], P = [6, 9]$ *are a refinement of the original interval* $X^0 \times P^0$ *with the guarantee that any* $(x, p) \in X^0 \times P^0$ *for which* $f(z,p) = 0$ *is also contained in* $X \times P$.

In some cases, it is possible that a univariate function operating on an intermediate factor is only defined for a subset of $\mathbb{R}$, for example, $\arccos x$ is only defined for $x \in [-1, 1]$. In the model presented in this article, a domain violation of the arccos function corresponds to the physical phenomenon of flooding of the separator unit. In this case, the model is invalid and evaluating it returns no meaningful solution. To prevent numerical artifacts from impacting the forward interval evaluation, the following convention will be used. Consider the univariate function $\phi : D \subset \mathbb{R} \rightarrow \mathbb{R}$. If $x \notin D$, then $\phi(x) \equiv \emptyset$. Let Φ be an interval extension of ϕ and suppose X is an interval that is not fully contained in D. In this case, it is safe to evaluate $\Phi(X \cap D)$ to obtain conservative bounds on the image of X under ϕ. It may be possible that $X \cap D = \emptyset$ which means that all points in X cause domain violations and, hence, the separator floods. Otherwise, at least one operating condition exists that does not cause flooding and the model can be evaluated safely.

Robust Simulation Results

The robust simulation case studies were performed on the same hardware mentioned previously operating Ubuntu Linux 10.04. Similar to the pointwise simulation case study, the mass fraction of gas in the feed stream from the wellhead was considered to be uncertain and the oil composition was held constant. Relating the notation from the problem formulation to the model, the state variables, the controls, and uncertainty parameter are stated formally as

$$\mathbf{z} = (\xi_{G4}, \xi_{W4}, \xi_{O4}, \dot{m}_3, \dot{m}_4, H_{GLS}, \xi_{G7}, \xi_{O7}, \dot{m}_6, \dot{m}_7, \dot{m}_8)$$
$$\mathbf{u} = (u_1, u_2)$$
$$\mathbf{p} = (\xi_{G1})$$

The total size of the system ends up as $n_x = 11$, $n_u = 2$, and $n_p = 1$. Although there are many more variables, they are associated with trivial equations and do not show up in the implicit function calculations (i.e., they do not end up as functions of $\mathbf{u}$ and $\mathbf{p}$). The performance specification is such that, in order to avoid damaging the pump, the GCU must be less than or equal to some specified amount, $G^{\max}$, which will be varied for the purpose of demonstrating the behavior of the algorithm. Stated formally, $g(\mathbf{z}, \mathbf{u}, \mathbf{p}) = \xi_{G7} - G^{\max} \leq 0$, where $\mathbf{z}$ is the vector of all the internal state variables, such as flow rates and compositions of each stream. As the simulation algorithm solves the model equations for the state variables as implicit functions of the uncertainty parameters and controls, represented as $\mathbf{x} : U \times P \rightarrow X$, the performance specification can be written as the following nonlinear implicit function

$$g(\mathbf{x}(\mathbf{u}, \mathbf{p}), \mathbf{u}, \mathbf{p}) = x_{G7}(\mathbf{u}, \mathbf{p}) - G^{\max} \leq 0$$

where x_{G7} is the relevant component of $\mathbf{x}$ representing the mass fraction of gas in the oil product stream. The robust simulation SIP (10) can be written for this problem as

$$\begin{aligned} \eta^* &= \max_{\mathbf{p} \in P, \eta \in \mathbb{R}} \eta \\ \text{s.t.}\ \eta &\leq x_{G7}(\mathbf{u}, \mathbf{p}) - 0.05, \forall \mathbf{u} \in U \\ P &= [0.35, 0.50] \\ U &= [0, 1] \times [0, 1] \end{aligned} \tag{14}$$

It is clear from Figure 2 that certain control valve settings lead to the flooding of the GLS. This phenomenon has not only physical implications but also, more importantly, numerical ones, which were discussed in the previous section. Since the GLS is modeled as a horizontal cylinder, the pertaining model equation contains the term $\cos^{-1}[1 - H/R]$ which is only defined on the compact set $\{H \in \mathbb{R} : 0 \leq H \leq 2R\}$. Thus, if the control valves are allowed to take values from fully opened to fully closed, it is easy to produce scenarios with $H > 2R$, and numerically, there is a domain violation of the $\cos^{-1}$ term and the model has no solution. When this situation was encountered in the study from the previous section, with results depicted in Figure 2, the process simulator simply fails, as expected. Such domain violations are the topic of current research.

In this work, the valid interval from which H_{GLS} may take values is $[0, 2R_{GLS}]$. However, the model solution is an implicit function of the controls $\mathbf{u}$ and the uncertain variable $\mathbf{p}$, and therefore, the liquid level in the GLS is dependent on both $\mathbf{u}$ and $\mathbf{p}$. As the level in the GLS is limited to the interval $[0, 2R_{GLS}]$, it is clear that the implicit function $\mathbf{x}$ does not exist for every $(\mathbf{u}, \mathbf{p}) \in U \times P$. Because of this, the global optimization algorithm for solving the implicit subproblems may encounter three situations. The first situation is one where a partition $U^l \times P^l \subset U \times P$ is popped off the stack in which $\mathbf{x}$ exists for every $(\mathbf{u}, \mathbf{p}) \in U^l \times P^l$, after applying forward-backward constraint propagation. This situation is of course no different than if $\mathbf{x}$ exists on all of $U \times P$. The second situation that may be encountered is one where $\mathbf{x}$ does not exist for any $(\mathbf{u}, \mathbf{p}) \in U^l \times P^l$. In this case, the subproblem is simply labeled as infeasible and the partition $U^l \times P^l$ is discarded. The third situation which may be encountered is one where $\mathbf{x}$ only exists for $(\mathbf{u}, \mathbf{p}) \in Q \subset U^l \times P^l$ after applying forward-backward constraint propagation. The question arises of how the global optimization algorithm may address this situation. For the purposes of this article, when this situation is encountered, relaxations of $\mathbf{x}$ can still be constructed using the theory of (Stuber et al., Submitted)[15] but $\mathbf{x}$ does not exist on all of $U^l \times P^l$, the functions that are constructed are underestimators and overestimators, respectively, of $\mathbf{x}$ on $Q \subset R \subset U^l \times P^l$ with $R \in \mathbb{IR}^{n_u} \times \mathbb{IR}^{n_p}$ and they are convex and concave, respectively, on R. In this case, the functions constructed do not exist for $(\mathbf{u}, \mathbf{p}) \in \{(\mathbf{a}, \mathbf{b}) \in U^l \times P^l : (\mathbf{a}, \mathbf{b}) \notin R\}$. The following example illustrates the idea.

Example 2. *Consider* $x(p) = \cos^{-1}[1 - p/2]$ *on* $P \in [-2, 6]$. *A factorable representation is given by*

$$v_1 = p/2$$
$$v_2 = 1 - v_1$$
$$v_3 = \cos^{-1} v_2$$

Calculating the forward interval evaluation of x on P results in

$$V_1 = [-2, 6]/2 = [-1, 3]$$
$$V_2 = 1 - [-1, 3] = [-2, 2]$$
$$\begin{aligned} V_3 &= \cos^{-1}([-2, 2] \cap [-1, 1]) \\ &= \cos^{-1}([-1, 1]) = [0, \pi] \end{aligned}$$

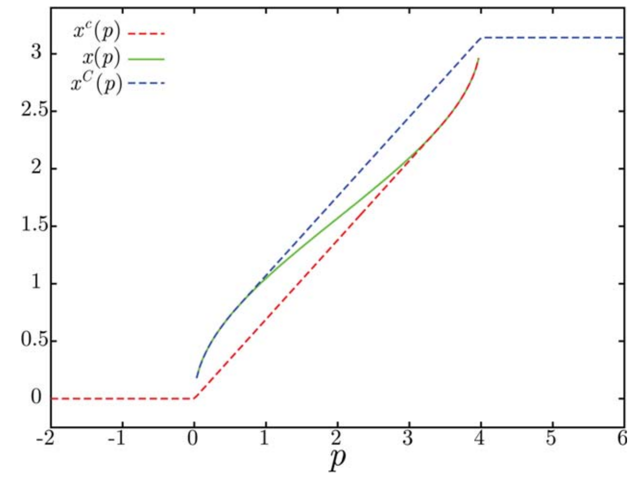

Figure 5. Convex and concave relaxations of $x(p)=\cos^{-1}[1-p/2]$ on $P=[-2,6]$ which are guaranteed to exist only for $p\in[0,4]$.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Using the theory of Stuber et al. (Submitted),[15] relaxations of x on P can be calculated. Again, these relaxations are only guaranteed to exist for p ∈ [0,4], as Figure 5 illustrates. Nevertheless, the relaxations computed are convex and concave on their domains of definition.

If the solver requires the evaluation of a function at a point in which it does not exist, it simply fails. In the event that the solver fails, the algorithm is notified. If this event occurs, the bounding information from the interval evaluation is used and the algorithm continues in the normal way. In essence, the problematic regions of the search space are systematically discarded by the algorithm.

To get a sense of the effectiveness of each of the interval methods on this problem, the forward-backward constraint propagation technique was applied to the model (with compact dimensions) with 100 iterations at the point $\mathbf{p}=(0.36)$ and $\mathbf{u}=(0.6, 0.8)$. The initial interval X^0, which is calculated automatically from *a priori* physical information about the problem, is shown in the second column of Table A6. Forward-backward constraint propagation took 0.006 s to complete and the results are reported in the third column of Table A5. Finally, interval-Newton with interval-Gauss–Seidel was applied iteratively to the refined interval from the forward-backward constraint propagation technique. The interval-Newton method converged after five iterations taking 0.008 s. The result of applying interval-Newton is shown in the last column of Table A5. It can be seen that the forward-backward constraint propagation technique was very effective at refining the initial interval; however, it failed to converge to degeneracy within 100 iterations. In fact, this technique is nearly converged at 100 iterations and no further refinement is observed after 200 iterations. However, interval-Newton is very effective in further refining the interval returned from forward-backward constraint propagation converging to a degenerate interval that represents the unique solution. This result is likely the consequence of the convergence criteria for this interval iteration being satisfied (see Neumaier[8]). It should be noted that interval-Newton, when applied to X^0, fails to make any refinement.

As previously mentioned, for this model, considerable overestimation was encountered using interval analysis to bound the implicit function $\mathbf{x}$, even with the forward-backward constraint propagation implementation. To further improve the effectiveness of the methods and circumvent the issues that arose due to the overestimation, the uncertainty interval P was subdivided and (14) was solved for each subdivision of P. The four uncertainty intervals considered are $P^1=[0.35, 0.3875]$, $P^2=[0.3875, 0.425]$, $P^3=[0.425, 0.4625]$, and $P^4=[0.4625, 0.50]$. To demonstrate the performance and applicability of the algorithm, four different cases were studied varying the size of the GLS and the performance specification.

Case 1

For this study, the control interval $U=[0.35,0.8]^2$ was considered. The GLS dimensions were such that $R_{\text{GLS}}=0.6$ m and $L_{\text{GLS}}=5$ m. The performance specification was such that $G^{\max}=0.05$.

The robust simulation algorithm solved the problem in a total time of 2.83 s, with a rigorous upper bound on η^* of -0.0105, implying $\eta^* \leq 0$. That is, for all realizations of wellhead compositions within the intervals considered, there exists a control setting from the interval considered such that the design meets the product purity specification that no more than 5% of the oil product stream can be gas.

Case 2

For this study, the same control interval and GLS dimensions from Case 1 were used and the uncertainty interval remains the same as the previous study. The performance specification was made much more strict although with $G^{\max}=0.0015$.

The robust simulation algorithm solved the problem in a total time of 56.7 s, with a rigorous upper bound on η^* of -1.17×10^{-3}, implying $\eta^* \leq 0$. The solution time of the algorithm was 20 times longer for this case as compared to Case 1, with the more relaxed performance specification. This is likely due to the fact that the feasible region for this problem is significantly smaller than that of Case 1.

Case 3

For this study, the GLS dimensions were such that $R_{\text{GLS}}=0.4$ m and $L_{\text{GLS}}=4$ m and the control interval remains the same as the previous studies. These new dimensions result in the GLS having roughly 36% of the volume of that in the previous two studies. The performance specification was such that $G^{\max}=0.05$. Intuitively, the smaller GLS dimensions will result in the new design having reduced performance as compared to the larger design. Therefore, it is expected that if this design is feasible, the feasible region of the SIP will be much smaller than that of Case 1.

The robust simulation algorithm solved the problem in a total time of 549.3 s with a rigorous upper bound on η^* of -5.77×10^{-3}. Again, this means that $\eta^* \leq 0$, implying that the design is feasible. The solution time of the algorithm suggests that verifying robust feasibility of the more compact design is much more difficult than for the original larger design. Again, this is likely due to the significantly smaller feasible region of the robust simulation SIP.

Case 4

The purpose of this study is to demonstrate the behavior of the algorithm when the design is not nearly as robust as that in the previous studies. The dimensions of the GLS are the same as in Case 3. For this study, the control valve V-2 is stuck at 50% and the control valve V-1 will have limited movement between 30 and 35%. The performance specification will be such that $G^{\max} = 0.05$.

The robust simulation algorithm solved the problem in a total time of 27.6 s, with a rigorous lower bound on η^* of 1.46×10^{-2}, implying $\eta^* > 0$. As expected, this more compact design with extremely limited control valves does not satisfy robust feasibility. With regards to solution time, the SIP algorithm performed rather favorably for this study. This seems rather counterintuitive as guaranteeing infeasibility of the design requires locating an SIP-feasible point that has a corresponding objective function value that is greater than zero (such a point provides a rigorous lower bound on η^*). However, for this problem, to guarantee infeasibility of the design, the algorithm simply needs to guarantee infeasibility of the design for one of the four uncertainty intervals. Therefore, although generating a rigorous lower bound on η^* that is greater than zero may be computationally expensive (especially for a problem with such a small SIP-feasible region), once it is done, the algorithm can terminate without needing to solve the remaining SIPs for the other uncertainty intervals.

Conclusions

In this work, the problem of rigorous worst-case design of subsea production facilities was addressed using a semi-infinite programming approach. A subsea separator that includes a GLS and a LLS was modeled and studied. A pointwise numerical simulation was performed, and it was identified that this technique would be insufficient at addressing the worst-case design question posed for subsea production facilities.

The worst-case design question was formulated as a logical feasibility constraint which was subsequently reformulated as a SIP with implicit functions embedded. The main algorithmic framework developed by Mitsos[14] and extended to implicit SIPs[15] was used to solve this design problem.

As this problem was cast as a feasibility problem, it was identified that solving the nonconvex implicit SIP to global optimality may not be necessary. In response, two additional criteria were implemented to terminate the implicit SIP algorithm if a rigorous guarantee on feasibility/infeasibility can be deduced prior to finding a global solution.

Due to the complex behavior of process systems models, it was identified early on that the overestimation encountered using interval analysis; in particular using parametric interval-Newton methods, with these models may be detrimental to calculating useful bounding information for implicit functions. To combat this, an automatic forward-backward constraint propagation technique was implemented to help refine interval bounding information required by the algorithm. The combination of the constraint propagation with the parametric interval-Newton method proved to be effective at calculating rigorous and convergent bounds on the implicit functions involved with the subsea separator model.

The worst-case design of a subsea separator was demonstrated by considering four case studies with varying operating scenarios and/or design choices. In each case, uncertainty in the gas composition of the input stream was considered along with two control valves: one on the entrance of the GLS and one on the entrance of the LLS. Even though the forward-backward constraint propagation technique proved to be effective at calculating the required bounding information when used in conjunction with parametric interval-Newton, the uncertainty interval was subdivided to avoid considerable overestimation which proved to be problematic for the algorithm. Overall, the algorithm performed favorably obtaining rigorous guarantees on robust feasibility/infeasibility with relatively little effort for each study. However, it should be noted that, in the general case, the computational complexity scales exponentially with the number of parameters and controls. Similarly, overestimation of the interval methods becomes more problematic not only with the width of the uncertainty intervals but also with the number of uncertainty parameters considered. Although the method presented herein proved to be effective at addressing the worst-case design problem of the subsea separator system, addressing the worst-case design problem of the entire subsea oil and gas production facility may prove to be difficult at this stage. Reducing the overestimation problem so that larger process systems can be addressed is the topic of continuing research.

Acknowledgment

The authors acknowledge the Chevron University Partnership program for funding this research through the MIT Energy Initiative.

Literature Cited

1. Little AG. Pumped up: chevron drills down 30,000 feet to tap oil-rich gulf of mexico. *Wired Magazine*. Vol. 15(9). 2007.
2. Biello D. One year after BP oil spill, at least 1.1 million barrels still missing. *Scientific American*. 2011.
3. Herron J. No closure yet on BP oil spill costs. *Wall St J*. 2010. Available at: http://blogs.wsj.com/source/2010/11/02/no-closure-yet-on-bp-oil-spill-costs/. Accessed on 3 November, 2010.
4. Stuber MD, Barton PI. Robust simulation and design using semi-infinite programs with implicit functions. *Int J Reliab Safety*. 2011; 5(3/4):378–397.
5. RES Group. *JACOBIAN Process Simulator*. Available at: http://www.numericatech.com/products.htm.
6. Halemane KP, Grossmann IE. Optimal process design under uncertainty. *AIChE J*. 1983;29(3):425–433.
7. Rudin W. *Principles of Mathematical Analysis*, 3rd ed. New York: McGraw-Hill, 1976.
8. Neumaier A. *Interval Methods for Systems of Equations*. Cambridge: Cambridge University Press, 1990.
9. 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.
10. Ostrovsky GM, Volin YM, Barit EI, Senyavin MM. Flexibility analysis and optimization of chemical plants with uncertain parameters. *Computers*. 1994;18(8):755–767.
11. Bhattacharjee B, Lemonidis P, Green WH Jr, Barton PI. Global solution of semi-infinite programs. *Math Program*. 2005;103:283–307.
12. Falk JE, Hoffman K. A nonconvex max-min problem. *Nav Res Logist Q*. 1977;24(3):441–450.
13. Blankenship JW, Falk JE. Infinitely constrained optimization problems. *J Optim Theory Appl*. 1976;19(2):261–281.
14. Mitsos A. Global optimization of semi-infinite programs via restriction of the right-hand side. *Optimization*. 2011;60(10–11):1291–1308.
15. Stuber MD. Evaluation of process systems operating envelopes. Ph.D. Thesis. Massachusetts Institute of Technology. 2013. Available at: http://yoric.mit.edu/sites/default/files/StuberThesis.pdf.
16. Falk JE, Soland RM. An algorithm for separable nonconvex programming problems. *Manag Sci*. 1969;15(9):550–569.
17. McCormick GP. Computability of global solutions to factorable nonconvex programs: Part I-convex underestimating problems. *Math Program*. 1976;10:147–175.
18. Adjiman CS, Floudas CA. Rigorous convex underestimators for general twice-differentiable problems. *J Global Optim*. 1996;9:23–40.
19. Gill PE, Murray W, Saunders MA. SNOPT: an SQP algorithm for large-scale constrained optimization. *SIAM Rev*. 2005;47(1):99–131.
20. Scott JK, Stuber MD, Barton PI. Generalized McCormick relaxations. *J Global Optim*. 2011;51(4):569–606.
21. Luksan L, Vlcek J. Algorithms for non-differentiable optimization. *ACM Trans Math Softw*. 2001;27(2):193–213.
22. Lin Y, Stadtherr MA. Advances in interval methods for deterministic global optimization in chemical engineering. *J Global Optim*. 2004; 29(3):281–296.

23. Balendra S, Bogle IDL. Modular global optimisation in chemical engineering. *J Global Optim.* 2009;45(1):169–185.
24. Bogle IDL, Byrne RP. Global optimisation of constrained non-convex programs using reformulation and interval analysis. *Comput Chem Eng.* 1999;23(9):1341–1350.
25. Jaulin L, Kieffer M, Didrit O, Walter E. *Applied Interval Analysis*. London: Springer-Verlag, 2001.
26. Schichl H, Neumaier A. Interval analysis on directed acyclic graphs for global optimization. *J Global Optim.* 2005;33(4):541–562.
27. Vu XH, Schichl H, Sam-Haroud D. Interval propagation and search on directed acyclic graphs for numerical constraint solving. *J Global Optim.* 2009;45(4):499–531.
28. Baharev A, Kolev L, Rév E. Computing multiple steady states in homogeneous azeotropic and ideal two-product distillation. *AIChE J.* 2011;57(6):1485–1495.
29. Knüppel O. PROFIL/BIAS—a fast interval library. *Computing.* 1994;53(3–4):277–287.
30. Melquiond G, Pion S, Brönnimann H. *Interval Arithmetic Library*. 2006. Available at: http://www.boost.org/doc/libs/1_49_0/libs/numeric/interval/doc/interval.htm.
31. Griewank A, Walther A. *Evaluating Derivatives*, 2nd ed. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2008.

Appendix A

Table A1. The Fluid Properties Used in the Subsea Separator Model

		Fluid Properties
API	35	Oil gravity (API)
SG_G	0.6	Specific gravity of gas
SG_W	1.0	Specific gravity of water
ρ_W°	1000	Density of water (kg/m^3) at standard conditions

Table A2. The Physical Design Specifications of the Subsea Separator Model

		Physical Design Specifications
k_{GLS}	$0.5/60 \approx 8.33\times10^{-3}$	GLS performance constant
k_{LLS}	$0.01/60 \approx 1.67\times10^{-4}$	LLS performance constant
C_{v1}	$1.0/60 \approx 1.67\times10^{-2}$	V-1 sizing coefficient (kg/Pa$^{1/2}$s)
C_{v2}	0.1675	V-2 sizing coefficient (kg/Pa$^{1/2}$s)
L_{GLS}	$\in \{4.0, 5.0\}$	Length (m) of GLS (varies by case)
R_{GLS}	$\in \{0.4, 0.6\}$	Radius (m) of GLS (varies by case)
P_{GLS}	4×10^6	Operating pressure (Pa) of GLS
L_{LLS}	5.0	Length (m) of LLS
R_{LLS}	0.8	Radius (m) of LLS
P_{LLS}	4×10^6	Operating pressure (Pa) of LLS
H_{LLS}	0.6	Liquid level in the LLS (m)

Table A3. The Input Conditions for the Subsea Separator Model

		Input Conditions
P_{well}	5.52×10^6	Wellhead pressure (Pa)
ξ_{G1}	$\in$ [0,1]	Mass fraction of gas, uncertain
ξ_{W1}	$\in$ [0,1]	Mass fraction of water, uncertain
ξ_{O1}	$\in$ [0,1]	Mass fraction of oil, uncertain

Table A4. The Control Settings for the Subsea Separator Model

		Control Settings
u_1	$\in$ [0,1]	Valve V-1 opening
u_2	$\in$ [0,1]	Valve V-2 opening

Table A5. The Symbols Used Throughout This Article and Their Descriptions

Other Symbols	
C_{vi}	Valve i constant (kg/Pa$^{1/2}$s)
g_a	Acceleration of gravity (m/s^2)
G^{max}	Maximum allowable gas fraction (by mass) in oil product
k_i	Separator i performance constant
H_j	Fluid level in separator j
η	Auxiliary optimization variable
L_j	Length of separator j
$\dot{m}_i$	Mass flow rate in stream i (kg/s)
$\mathbf{p}$	Uncertainty variable
P_i	Pressure of stream i (Pa)
R_j	Radius of separator j
SG_j	Specific gravity of component j
$\mathbf{u}$	Control variable
U	Interval bounding control variable
V_j	Total liquid volume in separator j (m^3)
V_{oil}	Volume of oil/gas mixture phase in LLS (m^3)
$\mathbf{x}$	Implicit function of the controls and uncertainty
X	Interval bounding the implicit function
ξ_{ji}	Mass fraction of component j in stream i
$\boldsymbol{\xi}_i$	$(\xi_{Gi},\xi_{Wi},\xi_{Oi})$ vector of mass fractions for stream i
ρ_i	Density of material in stream i (kg/m^3)
$\mathbf{z}$	Vector of internal state variables

Table A6. Forward-Backward Constraint Propagation was Applied with 100 Iterations to X^0 Resulting in the Refined Interval X^{FB}. Interval-Newton with Interval Gauss–Seidel was then Applied to X^{FB} Resulting in the Refined Interval X^N After Five Iterations.

	Interval Method Comparison		
	X^0	X^{FB}	X^N
ξ_{G4}	$[9.46305\times10^{-3}, 0.36]$	[0.080707,0.154937]	[0.11869,0.11869]
ξ_{W4}	[0.24,0.375]	[0.316898,0.344735]	[0.330492,0.330492]
ξ_{O4}	[0.4,0.625]	[0.500327,0.602395]	[0.550819,0.550819]
$\dot{m}_3$	[0,304.517]	[205.261,256.99]	[231.61,231.61]
$\dot{m}_4$	[541.364,845.881]	[588.891,640.62]	[614.271,614.271]
H_{GLS}	[0.462165,0.7992]	[0.546875,0.647172]	[0.59503,0.59503]
ξ_{G7}	$[8.6697\times10^{-3}$, 0.348991]	[0.0762948,0.148717]	[0.113166,0.113166]
ξ_{O7}	[0.651009,0.99133]	[0.851283,0.923705]	[0.886834,0.886834]
$\dot{m}_6$	[203.011,203.011]	[203.11,203.11]	[203.11,203.11]
$\dot{m}_7$	[338.352,642.869]	[338.352,437.609]	[381.528,381.528]
$\dot{m}_8$	[0,304.517]	[0,73.4416]	[29.7311,29.7311]

Manuscript received May 6, 2013, and revision received Mar. 7, 2014.