A Continuous Formulation for Logical Decisions in Differential Algebraic Systems using Mathematical Programs of Equilibrium Constraints

Submitted to: *Industrial & Engineering Chemistry Research*

October 19, 2012

Kody M. Powell (Corresponding Author)

kody.powell@utexas.edu

The University of Texas at Austin

Department of Chemical Engineering

200 E. Dean Keeton St. Stop C0400

Austin, TX 78712-1589

John D. Hedengren

Brigham Young University

Department of Chemical Engineering

Thomas F. Edgar

The University of Texas at Austin

Department of Chemical Engineering

ABSTRACT

This work presents a methodology to represent logical decisions in differential algebraic equation constrained optimization problems using a set of purely continuous algebraic equations. The formulations may be used when state variables trigger a change in process dynamics, and introduces a pseudo-binary decision variable, which is continuous, but will only take on values of either zero or one at the solution. This formulation enables dynamic optimization problems with logical disjunctions to be solved without using less computationally efficient methods, such as mixed integer programming or sequential solution methods. Several case studies are given to illustrate the value of this methodology including nonlinear model predictive control of a chemical reactor using a surge tank with overflow to buffer disturbances in feed flow rate. In this example, the continuous logic formulation of the problem solves over 300 times faster than the corresponding sequential method formulation.

1. INTRODUCTION

In the field of simulation, optimization, and control, models are ideally formulated as a set of continuous equations with continuous derivatives, so that solutions can be efficiently obtained using gradient-based solution algorithms, such as Newton's method. However, in many systems, the need frequently arises to include operators that may be discontinuous (such as the signum operator) or have discontinuous first derivatives (such as the absolute value operator). The introduction of such discontinuities into a model can have adverse impacts on the solver's ability to efficiently obtain an accurate solution due to the introduction of non-smooth gradients. Such problems have to be re-formulated and solved using a less desirable method.

In the field of dynamic optimization and control, optimization problems are particularly difficult, due to the high dimensionality of time-dependent problems, as model predictions and control actions for every time step must be prescribed by the solver. Furthermore, online applications require fast solution times so that control actions can be calculated and recommended within some pre-determined sampling period. The introduction of discontinuities further complicates matters, as some practitioners may resort to computationally expensive solution methods, such as Mixed Integer Nonlinear Programming (MINLP), in order to implement such disjunctive constraints.

Mathematical programs of equilibrium constraints (MPECs) have been proposed as a way to integrate non-smooth behavior into a set of simultaneous algebraic equations by the inclusion of complementarity conditions[1,2]. Complementarity, the requirement that at least one of a pair of variables be at some limit, provides a framework for representing disjunctive behavior using a set of continuous equations. MPECs using complementarity constraints have found use in optimization problems in the fields of structural mechanics[3,4], chemical and process engineering[5–7], electric power generation[8], and other fields [9,10].

Mathematical programs of complementarity constraints (MPCCs) are a subset of MPECs and can be used to represent non-smooth or discontinuous operators, such as absolute value, sgn, and min/max[11]. This work presents the formulation of a greater than or equal to (≥) and a less than or equal to (≤) operator, which can be used for if/then logic in a process model. The formulation is presented as a set of continuous algebraic equations. The equations are formulated in such a way, however, that only binary (0 or 1) solutions are obtained for certain variables. These pseudo-binary variables are then used to represent logical conditions within the model. This work does not present a detailed explanation of the convergence properties of MPECs, but rather puts forward a novel formulation that can be used by practitioners to represent logical statements within a continuous process model.

2. Motivation

2.1. Logical Disjunctions In Optimization

Logical expressions, such as the less than/equal to (≤) operator are typically introduced into optimization problems through the use of mixed integer programming, where certain variables are constrained at integer values. A general disjunctive program can be converted to an equivalent MINLP[12,13] and solved using various MINLP algorithms[14–16]. However, one drawback to MINLP formulations is that solution times grow exponentially with an increased number of discrete decisions[5]. When considering dynamic optimization problems, where the time domain is typically discretized and a set of decisions is required for each time, optimization problems can become especially large. When rapid solution is required, converting a large dynamic optimization problem with disjunctions to an MINLP problem may not be a tractable option. Therefore, the ability to embed logical statements or other disjunctive operators as sets of algebraic equations (or MPECs) while maintaining mathematical continuity, allows the problems

to be posed as standard nonlinear programming (NLP) problems, for which many efficient real-time solvers exist.

2.2. Sequential Solution Method

When converting a dynamic optimization problem into an NLP, two basic methodologies exist: sequential methods and simultaneous methods[17]. A sequential method employs a forward-stepping differential algebraic equation (DAE) or ordinary differential equation (ODE) solver, using a Runge-Kutta or similar numerical integration technique. Using this method, inputs at every time step are specified. The DAE solver then integrates forward one step at a time using the pre-specified inputs. The sequential method ensures that the state equations are satisfied at all times, as they are enforced by the DAE solver as integration transpires. Logical statements and other disjunctions are fairly easy to implement when using sequential methods, as the state equations can be altered at any point during the integration. For example, when a state variable reaches some limit that triggers a disjunction, a logical statement can be embedded into the DAE model ensuring that the change will be applied to future output from the model while that particular condition holds.

Sequential methods for solving DAE systems certainly have some advantages. When used to solve dynamic optimization problems, however, the disadvantages of sequential methods far outweigh these advantages. These methods are inefficient for large-scale optimization problems because they require simulating the model many times with different values of inputs (at every time step) in order to compute numerical approximations to gradient matrices so that new guesses can be calculated. The simulated solutions are continuously converged from initial values that are not optimal, leading to excessive CPU time that is devoted to intermediate solutions. The requirement to converge the model equations at every iteration also leads to a challenge for unstable systems. If the specified decision variables produce an unstable response, the iteration may fail to find an adequate search direction for the next iteration[18]. It is also difficult to enforce inequality constraints on state (or dependent) variables because the values of these variables at each time step are only obtained by forward integration using a set of pre-determined inputs, therefore, constraints cannot be directly imposed on these variables.

2.3. SIMULTANEOUS SOLUTION METHOD

Simultaneous solution methods are frequently used in industry for dynamic optimization and real-time control problems because they help to overcome many of the computational inefficiencies associated with sequential solution methods[19–21]. Simultaneous solution methods use collocation (more specifically, orthogonal collocation on finite elements[22,23]) to convert a DAE-constrained dynamic optimization problem to an NLP where the objective function is minimized and the constraint equations are solved simultaneously, making the algorithm much more computationally efficient. By comparison, a sequential method requires simulating through the differential constraint equations many times for every set of inputs[24].

The crux of a simultaneous solution method is the conversion of the DAE system to a system of purely algebraic equations using a collocation method. The differential equations are specified in (1) with time derivatives given as a function (f) of differential state variables (x), algebraic state variables (y), user-controlled inputs (u), and external inputs (p), each of which is a function of τ, a variable representing time, normalized to the range [0,1] over the time interval.

$$\dot{x}(\tau) = f\left(x(\tau), y(\tau), u(\tau), p(\tau)\right) \tag{1}$$

Conversion of these differential equations is done by representing differential state profiles in time by polynomial approximations, which are generated using Lagrange interpolation polynomials (Ω). These polynomials are formulated to exactly match the value of the derivatives when evaluated at the collocation points (τ_i). This relationship, assuming constant inputs over the time interval, is shown in (2), where the derivatives are approximated as the summation of f evaluated at each collocation point (τ_j) multiplied by the corresponding interpolation polynomial (Ω_j).

$$\dot{x}(\tau_i) = \sum_{j=1}^{N_C} \Omega_j(\tau_i) f\left(x(\tau_j), y(\tau_j), u(\tau_j), p(\tau_j)\right) \tag{2}$$

The Lagrange polynomials are formulated as shown in (3) and are of order N_C-1, where N_C is the number of collocation points used in the approximation over the time interval[25].

$$\Omega_j(\tau) = \prod_{k=1,k\neq j}^{N_C} \frac{\tau-\tau_k}{\tau_j-\tau_k} = \frac{\tau-\tau_1}{\tau_j-\tau_1}\frac{\tau-\tau_2}{\tau_j-\tau_2}\cdots\frac{\tau-\tau_{N_C}}{\tau_j-\tau_{N_C}} \tag{3}$$

The relationship in (2) holds exactly at the collocation points because each polynomial (Ω_j) in (3) is formulated to have a value of unity at the corresponding collocation point (τ_j) and a value of zero at all the other collocation points[25].

$$\Omega_j(\tau_i) = \begin{cases} 1, & \tau_i = \tau_j \\ 0, & \tau_i \neq \tau_j \end{cases} \tag{4}$$

With state derivatives guaranteed to exactly match at the collocation points, the state variables themselves are approximated by integrating (2).

$$\hat{\Omega}(\tau) = \int_0^1 \Omega(\tau)\, d\tau \tag{5}$$

This allows for the state values themselves to be approximated.

$$x(\tau_i) = x_0 + w\sum_{j=1}^{N_C} \hat{\Omega}_j(\tau_i) f\left(x(\tau_j), y(\tau_j), u(\tau_j), p(\tau_j)\right) \tag{6}$$

where $\hat{\Omega}_j$ is the integral of Ω_j, which is a polynomial of order N_C, x_0 is the value of the state variable at the beginning of the time interval, and h is the width of the time interval.

In order to ensure integration accuracy and that Ω is explicitly defined at the right end of the time interval (τ=1), Radau collocation points are used. The Radau collocation points are derived from Radau quadrature, which is similar to Gaussian quadrature, except that one collocation point is defined explicitly at one end (rather than having all points exclusively in the interior) of the time interval[26]. For dynamic optimization applications, the interval is 0 to 1, with the state values at 0 obtained from the previous interval, and with a collocation point set exactly at 1.

With an approximation for a single time interval defined, multiple time intervals can be joined together, with a separate polynomial representing each interval, or finite element. The initial condition for each time interval is given as the final condition of the previous time interval (C^0-continuity). Other quadrature methods propagate first derivatives (C^1-continuity) or higher p-order derivative information (C^p-continuity) across the interval boundaries[27] to achieve higher accuracy across intervals. Figure 1 illustrates the orthogonal collocation on finite elements discretization scheme. Each time interval (k) of length w contains N_C collocation points. The example in the figure uses N_C=3, but higher or lower orders of approximation also exist. The

approximation from finite element k would use the state value from the last collocation point (i=N_C) of element k-1 as its initial condition, as shown in (7). In (7), the subscripts (i and j) refer to the collocation point and the superscript (k) refers to the finite element number.

$$x_i^k(\tau_i) = x_{N_C}^{k-1} + w\sum_{j=1}^{N_C}\hat{\Omega}_j(\tau_i) f\left(x_j^k, y_j^k, u_j^k, p_j^k\right) \tag{7}$$

With the approximation in (7) completed, the differential equations are converted into algebraic equations, which can be solved by a nonlinear algebraic equation solver. Therefore, enforcing additional algebraic equality constraints (g) becomes possible, as these equations (8) can be included with the algebraic equations in (7).

$$g\left(x_j^k, y_j^k, u_j^k, p_j^k\right) = 0 \tag{8}$$

Nonlinear inequality constraints can also be included, as can upper and lower bounds on the variables themselves.

$$h\left(x_j^k, y_j^k, u_j^k, p_j^k\right) \le 0 \tag{9}$$

$$u_l \le u \le u_u \tag{10}$$

$$x_l \le x \le x_u \tag{11}$$

$$y_l \le y \le y_u \tag{12}$$

The ability to directly impose constraints on state variables is one of the advantages of a simultaneous solution method, as opposed to sequential method. The algebraic formulation of (6)-(12) lends itself quite well to inclusion in an optimization problem which can be converged by an NLP solver.

2.4. Embedding MPECs into Simultaneous Equations

One of the disadvantages of a simultaneous solution method compared to a sequential method is that it is much more difficult to embed disjunctive constraints or logical conditions. Because the model is solved as a set of simultaneous algebraic equations, the introduction of disjunctions would make it difficult to solve the equations by standard methods. However, with the ability to enforce algebraic constraints within a differential model, MPECs, which are formulated as sets of algebraic equations, can be embedded into the model to represent disjunctions. These MPECs take advantage of a complementarity condition that at least one of two constraints be active, as shown in (8), where $\perp$ is the complementarity operator, enforcing at least one of these constraints at all times[5,10].

$$0 \le \upsilon_+ \perp \upsilon_- \ge 0 \tag{13}$$

In this work, υ_+ and υ_- are referred to as complementarity variables. The condition in (13) can be maintained by using a number of different formulations, and the performance of each may depend on the solution algorithm used. The first option is to represent the complementarity as an equality constraint as in (14).

$$\upsilon_+ \upsilon_- = 0 \tag{14}$$

This equation requires that at least one of the pair υ_+ and υ_- be equal to zero. Alternatively, inequality constraints may also be used.

$$\upsilon_+ \upsilon_- \le 0 \tag{15}$$

or

$$\upsilon_+ \upsilon_- \le \varepsilon \tag{16}$$

where ε is a very small positive number, indicating that some error in this relationship may be tolerated in order to enhance the convergence properties of interior point NLP methods[5].

Using the complementarity condition, several different MPECs can be formulated to represent some commonly used functions. These sets of equations can be embedded into a DAE model and keep the model continuous and smooth, despite the fact that these operators represent non-smooth or discontinuous operators in standard practice.

2.4.1. ABSOLUTE VALUE OPERATOR

The absolute value operator

$$y = |x| \tag{17}$$

can be alternatively represented in a continuous optimization problem by embedding the following equations into the DAE or algebraic model:

$$x = \upsilon_+ - \upsilon_- \tag{18a}$$

$$\upsilon_+, \upsilon_- \ge 0 \tag{18b}$$

$$\upsilon_+ \upsilon_- = 0 \tag{18c}$$

$$y = \upsilon_+ + \upsilon_- \tag{18d}$$

In (18b), the complementarity variables are restricted to be nonnegative. Because the complementarity condition (18c) requires that at least one of these variables be zero, (18a)

represents the difference between two nonnegative values. When x is positive, υ_- must be zero in order to satisfy (18c). υ_+ is therefore positive and equal to *x*. Thus, the summation of υ_+ and υ_- in (18d) becomes equal to the absolute value of *x*. Similarly, for negative *x*, υ_- must be positive and υ_+ must be zero. The summation of these two nonnegative values (18d), therefore, will always be a positive number equal in magnitude to *x*[5].

2.4.2. MIN/MAX OPERATOR

The min and max operators, which select the minimum and maximum value, respectively, of two inputs (x_1 and x_2)

$$y = \min\left(x_1, x_2\right),\ z = \max\left(x_1, x_2\right) \tag{19}$$

can also be represented using MPEC formulations.

$$x_1 - x_2 = \upsilon_+ - \upsilon_- \tag{20a}$$
$$\upsilon_+, \upsilon_- \geq 0 \tag{20b}$$
$$\upsilon_+ \upsilon_- = 0 \tag{20c}$$
$$y = x_1 - \upsilon_+ \tag{20d}$$
$$z = x_1 + \upsilon_- \tag{20e}$$

In this formulation, if x_1 is greater than x_2, υ_+ will assume the difference between these values. υ_- will be zero in order to satisfy the complementarity condition (20c). The lesser of x_1 and x_2 will therefore be the higher number (x_1) minus the difference (υ_+) leaving *y* to be equal to the min of the two as specified in (20d). The greater number will be the higher number plus υ_-, which is zero in this case. Therefore, *z* will represent the max of the two numbers, as (20e) indicates[5].

2.4.3. SIGNUM OPERATOR

The signum operator gives an output of +1 for positive input and -1 for negative input.

$$y = \mathrm{sgn}\left(x\right) \tag{21}$$

This binary behavior can also take on a continuous representation by using an MPEC formulation.

$$x = \upsilon_+ - \upsilon_- \tag{22a}$$

$$\upsilon_+, \upsilon_- \geq 0 \tag{22b}$$

$$\upsilon_+ \upsilon_- = 0 \tag{22c}$$

$$\upsilon_+ (1-y) + \upsilon_- (1+y) = 0 \tag{22d}$$

As (22) indicates, when x is positive, υ_+ will also be positive and equal in magnitude to x. Because υ_- will be zero, y will have to equal +1 in order to satisfy (22d). Similarly, when x is negative, y will be equal to -1, as a positive value of υ_- and a zero value of υ_+ will enforce this in (22d)[5].

3. MPEC Formulations to Represent Logical Statements

Because MPECs provide a continuous formulation to represent some disjunctive relationships, it is possible to represent some logical behavior within a model using similarly constructed MPECs which take advantage of the complementarity relationships described above. For instance an MPEC can be used to represent a binary variable, which is 1 when some condition is true and 0 otherwise. This binary variable can then be integrated into the model equations such that certain equations only hold true under the logical conditions dictated by the MPEC. The remainder of this section discusses the development of a greater than/equal to ($\geq$) and a less than/equal to ($\leq$) operator. Section 4 will then discuss the methodology for implementing such logic into a set of DAEs.

3.1. Greater Than and Less Than Operators

With only a slight modification of (22), the MPEC can be constructed so as to produce a 1 for a positive input (x) and a 0 for a negative input. Here, the variable δ is introduced to represent the binary output of this MPEC.

$$\delta = \begin{cases} 1 \text{ if } x>0 \\ 0 \text{ if } x<0 \end{cases} \tag{23}$$

The MPEC formulation is very similar to the signum operator, with only a slight modification made in the fourth equation. As (24d) indicates, the output of this MPEC can be customized to yield various constants, depending on the terms added to or subtracted from δ.

$$x = \upsilon_+ - \upsilon_- \tag{24a}$$

$$\upsilon_+, \upsilon_- \geq 0 \tag{24b}$$

$$\upsilon_+ \upsilon_- = 0 \tag{24c}$$

$$\upsilon_+(1-\delta)+\upsilon_-(\delta)=0 \tag{24d}$$

Using the formulation in (24), δ becomes a pseudo-binary variable, one which is continuous, but can only assume values of zero or one at the solution for negative or positive values of x, respectively.

3.2. Greater Than/Equal To and Less Than/Equal To Operators

Careful inspection of (24) reveals a major shortcoming. When x=0, both complementarity variables are simultaneously equal to zero. This means that (24d) will be satisfied by any value of δ, as the system has an infinite number of solutions in this case. The MPEC equations must therefore be modified in order to give the system the discrete switching behavior that is desired with no ambiguity for any value of x.

$$\delta=\begin{cases}1 \text{ if } x\geq 0\\ 0 \text{ if } x<0\end{cases} \tag{25}$$

Adding a second complementarity condition to the set of equations is proposed to overcome the issue of ambiguity when x=0. This equation (26d) contains a third complementarity variable, υ_0, and is designed such that υ_0 will take on some finite (albeit still ambiguous) value when υ_+ and υ_- are simultaneously zero, due to the input, x, being equal to zero.

$$x=\upsilon_+-\upsilon_- \tag{26a}$$
$$\upsilon_+,\upsilon_-\geq 0 \tag{26b}$$
$$\upsilon_+\upsilon_-=0 \tag{26c}$$
$$\left(\upsilon_+^2+\upsilon_-^2\right)\upsilon_0=0 \tag{26d}$$
$$\upsilon_+(1-\delta)+\upsilon_0(1-\delta)+\upsilon_-(\delta)=0 \tag{26e}$$

In (26e) a third term is added for the case that only υ_0 is nonzero (which occurs when x=0). However, some ambiguity still exists in this formulation, namely, that all complementarity variables may simultaneously be zero when x is zero, thereby satisfying (26e), regardless of the value of δ. In order to prevent this occurrence, υ_+ and υ_- are squared in (26d) order to ensure that these squared terms converge to zero at a faster rate, leaving υ_0 at some nonzero value. With zero values for υ_+ and υ_- and a finite value for υ_0, the (1-δ) term multiplying υ_0 must equal zero, giving δ a value of 1 when x=0. Changing the δ term in (26e) will obviously affect what δ

converges to in this case, meaning that the MPEC can be formulated so that δ takes on some other, user-determined, value. The same holds true for the terms multiplying υ_+ and υ_- if other outputs are desired for positive and negative values for *x*, respectively.

An alternate formulation using only equality constraints is used for testing the convergence properties of this logical MPEC. The non-negativity constraints in (26b) are removed and these constraints are instead enforced by squaring the complementarity variables in the first equation (27a). Note that this is a system of four equations and four unknowns, with x being considered an external input to this system.

$$x = \upsilon_+^2 - \upsilon_-^2 \tag{27a}$$

$$\upsilon_+ \upsilon_- = 0 \tag{27b}$$

$$\left(\upsilon_+^2 + \upsilon_-^2\right)\upsilon_0 = 0 \tag{27c}$$

$$\upsilon_+ \left(1-\delta\right) + \upsilon_0 \left(1-\delta\right) + \upsilon_- \left(\delta\right) = 0 \tag{27d}$$

This system of equations is evaluated for convergence properties using Newton's method for solving systems of nonlinear equations. The system exhibits no issues with convergence for positive and negative values of *x*, with δ converging to 1 and 0, respectively, as desired. The predominant concern is obtaining a distinct desired solution when *x* is zero. Newton iterations for this scenario are shown in Figure 2 and 3. As Figure 2 illustrates, υ_+ and υ_- converge to zero as expected. The other complementarity variable, υ_0, however, remains at its initial guess value, as the squared terms in (27c) converge to zero in order to satisfy (27a). This finite value for υ_0, however, forces δ to converge exactly to 1 in order to satisfy (27d), rather than leaving this value ambiguous, as the formulation in (24) would have.

4. Continuous Logic in Dynamic Systems

Using the collocation scheme combined with the logical MPEC framework developed in the previous section, dynamic systems of equations with logical conditions can be simulated using only a set of continuous algebraic equations. This is done by embedding a logical MPEC into the DAE system. The pseudo-binary variable, δ, from this MPEC can be multiplied with the model equations, meaning that some equations will hold only when δ =1. Two simulation examples are used to illustrate how this is done.

4.1. TANK WITH OVERFLOW

A simple example to illustrate the need for representing logic in a DAE model is that of a simple tank with overflow, shown in Figure 4. While the dynamics of this system are trivial, the equations representing the dynamic behavior of the tank change dramatically when the tank reaches its overflow limit. The system, as posed in (28), can be represented as a simple ODE combined with a logical expression determining when the tank overflows.

$$\frac{dV}{dt} = Q_{in} - Q_{out} - Q_{over} \tag{28a}$$

$$Q_{over} = \begin{cases} Q_{in} - Q_{out} \text{ if } V \geq V_{max} \ \& \ Q_{in} > Q_{out} \\ 0 \text{ otherwise} \end{cases} \tag{28b}$$

where V is the tank volume, Q_{in} is the flow into the tank, Q_{out} is the flow out of the tank, and Q_{over} is the flow exiting the tank as overflow, when the tank volume exceeds its capacity, V_{max}. While the system simple in (28) is very simple, the logical statement (28b) prevents it from being solved using a standard simultaneous solution method. However, by including the algebraic equations representing the greater than/equal to logic MPEC, this system can be solved using a simultaneous solution method. This DAE system translated into a continuous logic formulation using an MPEC with complementarity constraints is given in (29), where (29e-29h) represent the additional algebraic equations introduced by the logical MPEC.

$$\frac{dV}{dt} = Q_{in} - Q_{out} - Q_{over} \tag{29a}$$

$$\left(1 - \delta_{hi}\right) Q_{over} = 0 \tag{29b}$$

$$Q_{over} \geq 0 \tag{29c}$$

$$V \leq V_{max} \tag{29d}$$

$$V - V_{max} = \upsilon_{+}^{2} - \upsilon_{-}^{2} \tag{29e}$$

$$\upsilon_{+}\upsilon_{-} = 0 \tag{29f}$$

$$\left(\upsilon_{+}^{2} + \upsilon_{-}^{2}\right)\upsilon_{0} = 0 \tag{29g}$$

$$\upsilon_{+}^{2}\left(1 - \delta_{hi}\right) + \upsilon_{-}^{2}\left(\delta_{hi}\right) + \upsilon_{0}\left(1 - \delta_{hi}\right) = 0 \tag{29h}$$

In this formulation, δ_{hi} is a pseudo-binary variable that is equal to one when the tank is full and zero when it is not full. When the tank is not full, (29b) will ensure that Q_{over} is zero. When the tank is full, Q_{over} will take on whatever value necessary to satisfy the material balance (29a). However, Q_{over} must be restricted to non-negative values in order to prevent negative values of Q_{over} from satisfying (29a) when the tank is not full. The MPEC tests whether the

quantity V-V_{max} is greater than or equal to zero. However, in order to enhance convergence properties, V is also restricted by (29d), so that V cannot exceed its limit. Alternatively, this constraint can be imposed solely by the MPEC equations. However, this may lead to poor convergence properties of the system. Convergence is also enhanced in this case by squaring υ_+ and υ_- in (29g) and (29h), forcing the squared terms to converge more quickly so that υ_0 remains near its initial guess in the event that the system is at its volume limit.

In order to demonstrate the ability of (29) to accurately represent a logic-dependent dynamic system, the set of equations with pre-specified inputs (Q_{in} and Q_{out}) is solved using a DAE solution package known as Advanced Process Monitor, or APMonitor [28]. This software package allows a user to define a model using both differential and algebraic equations. The software performs the collocation to convert the differential equations to algebraic equations and the problem is converted to a set of nonlinear algebraic equations. For optimization, an NLP problem would be solved. Because the system is still a continuous set of equations, APMonitor computes the gradient matrices with automatic differentiation, ensuring accuracy and fast solution times. The APOPT solver, which uses an active set method, demonstrates the best convergence as the problem is solved assuming some set of constraints to be active, which works well with inequality constraints such as (29c) and (29d).

The results of the simulation are shown in Figures 5 through 8. As Figures 5 and 6 illustrate, the overflow (Q_{over}) remains at zero until the tank fills. Once the tank fills, the logical condition that Q_{over}=0 is nullified as δ_{hi}=1, allowing Q_{over} to take on whatever positive value is needed to satisfy (29a). The complementarity variables (Figure 8) are well behaved, with υ_- equaling zero when the tank is at the high limit and υ_0 equaling zero when the tank is not at the high limit. The positive complementarity variable (υ_+) is always zero as the system is prevented from exceeding the high limit by (29d).

Remarkably, the results in Figures 5-8 illustrate that logic can be embedded into a dynamic system using only continuous algebraic equations to model the system. While convergence for the formulation in (29) is obtained, there are many variations of the MPEC formulation, some of which do not display the same ability to converge consistently. When

implementing similarly-formulated MPECs, it may be necessary to explore various formulations to determine which will be the most robust for the application and choice of solver.

4.2. POWER FLOW SYSTEM

The logical MPEC's performance is also tested in a power flow system (shown in Figure 9) with a photovoltaic solar panel, a battery, a load (represented by a building), and the electric grid. This system assumes simple dynamics for the battery (30a). Energy balances are computed around the photovoltaic panel and the load in order to obtain (30b) and (30c), respectively. A logic-based operating strategy is applied in order to specify the system's operation. Using this strategy, the maximum amount of solar power is delivered to the load by using the battery. When solar power available (q_{PV}) exceeds the demand (q_{load}), the battery (whose state of charge is represented by E_{batt}) is charged. When the battery reaches its capacity (E_{max}), the excess power is delivered to the grid with flow q_3. Conversely, when the battery is void of charge, power must be imported from the grid to the load with flow q_4. This logic is specified in (30d) and 30e). The variables q_1 and q_2 represent the power delivered to and extracted from the battery, respectively.

$$\frac{dE_{batt}}{dt} = q_1 - q_2 \tag{30a}$$

$$0 = q_{PV} - q_1 - q_3 \tag{30b}$$

$$0 = q_{load} - q_2 - q_4 \tag{30c}$$

$$q_3 = \begin{cases} q_{PV} - q_1 \text{ if } E_{batt} \geq E_{max} \ \& \ q_{PV} > q_1 \\ 0 \text{ otherwise} \end{cases} \tag{30d}$$

$$q_4 = \begin{cases} q_{load} - q_2 \text{ if } E_{batt} \leq 0 \ \& \ q_{load} > q_2 \\ 0 \text{ otherwise} \end{cases} \tag{30e}$$

Conversion of the model to continuous form requires two sets of logical MPEC equations representing the logical decisions of (30d) and (30e). This requires two sets of pseudo-binary (δ) and complementarity variables (υ), which are assigned the subscripts hi and lo, corresponding to the full (30d) and empty (30e) battery charge conditions, respectively. When converted to continuous logic form, (30) becomes (31).

$$\frac{dE_{batt}}{dt} = q_1 - q_2 \tag{31a}$$

$$0 = q_{PV} - q_1 - q_3 \tag{31b}$$

$$0 = q_{load} - q_2 - q_4 \tag{31c}$$

$$E_{min} \leq E_{batt} \leq E_{max} \tag{31d}$$

High limit MPEC eqns corresponding to (30d)

$$\left(1-\delta_{hi}\right)q_3 = 0 \tag{31e}$$

$$q_3 \geq 0 \tag{31f}$$

$$E_{max} - E_{batt} = \upsilon_{hi+}^2 - \upsilon_{hi-}^2 \tag{31g}$$

$$\upsilon_{hi+}\upsilon_{hi-} = 0 \tag{31h}$$

$$\left(\upsilon_{hi+}^2 + \upsilon_{hi-}^2\right)\upsilon_{hi,0} = 0 \tag{31i}$$

$$\upsilon_{hi+}^2\left(1-\delta_{hi}\right) + \upsilon_{hi-}^2\left(\delta_{hi}\right) + \upsilon_{hi,0}\left(1-\delta_{hi}\right) = 0 \tag{31j}$$

Low limit MPEC eqns corresponding to (30e)

$$\left(1-\delta_{lo}\right)q_4 = 0 \tag{31k}$$

$$q_4 \geq 0 \tag{31l}$$

$$E_{min} - E_{batt} = \upsilon_{lo+}^2 - \upsilon_{lo-}^2 \tag{31m}$$

$$\upsilon_{lo+}\upsilon_{lo-} = 0 \tag{31n}$$

$$\left(\upsilon_{lo+}^2 + \upsilon_{lo-}^2\right)\upsilon_{lo,0} = 0 \tag{31o}$$

$$\upsilon_{lo+}^2\left(1-\delta_{lo}\right) + \upsilon_{lo-}^2\left(\delta_{lo}\right) + \upsilon_{lo,0}\left(1-\delta_{lo}\right) = 0 \tag{31p}$$

The continuous logic formulation for the power flow system is demonstrated using a simulation with pre-determined q_{pv} and q_{load} over a 24-hour time horizon, which is shown in Figure 10. Hourly time intervals are used in the simulation. As the figure shows, the supply (q_{pv}) and demand (q_{load}) do not perfectly coincide, with the available solar power peaking near midday and the demand peaking later in the afternoon, requiring the system to use battery energy storage in order to maximize the power delivered to the load from the solar panel. As Figures 11 through 13 illustrate, at the beginning of the day, there is no charge in the battery (indicated by δ_{lo}=1) and the demand exceeds the load, forcing power to be drawn from the grid. As the solar power picks up, the battery charges until it reaches its capacity (indicated by δ_{hi}=1). When this occurs, the logic dictates that the excess power be exported from the solar panel to the grid, indicated by the positive values for q_3 in Figure 11. At the end of the day, the solar power is diminished, the battery completely discharges, and power is again imported from the grid.

The power flow example again demonstrates the value of using MPECs to represent logical decisions in a DAE system. Embedding this logic in the form of continuous algebraic equations allows the system to be solved using the simultaneous method, which has been proven to significantly increase computational efficiency as compared to a sequential method.

5. CONTINUOUS LOGIC IN AN NMPC PROBLEM

As a demonstration of the value of integrating logic into a simultaneous solution method, a nonlinear model predictive control (NMPC) problem is solved for a continuous stirred tank reactor (CSTR), which carries out the reaction:

$$A + B \rightarrow 2C \tag{32}$$

The objective of the controller is to regulate the concentration of component C (C_C) using the heat input to the reactor (q_{heat}) and the flow rate of component B (Q_B) as manipulated variables. The system is subject to disturbances in the flow of component A ($Q_{A,in}$) and is equipped with a surge tank to buffer out the effects of sudden increases in Q_A. However, in the case that the volume of fluid in the surge tank exceeds the tank capacity, the surge tank will overflow and a sudden increase in the flow of A will enter the CSTR as shown in Figure 14. NMPC in this scenario can monitor the level in the surge tank (h) and the flow of A coming into the surge tank so that sudden disturbances due to surge tank overflow can be anticipated and accounted for pre-emptively by the controller. The model requires a built-in logical statement as in (28) to represent the tank overflow condition.

In the MPC problem, the outflow from the bottom of the surge tank ($Q_{A,out}$) is proportional to the square root of the height (h) in the tank (34), with the dynamics of the tank represented by a simple material balance (33). The model requires a built-in logical statement to represent the tank overflow condition (35).

$$A_{tank}\frac{dh}{dt} = Q_{A,in} - Q_{A,out} - Q_{A,over} \tag{33}$$

$$Q_{A,out} = C\sqrt{h} \tag{34}$$

$$Q_{A,over} = \begin{cases} Q_{A,in} - Q_{A,out} & \text{if h} \geq h_{max} \;\&\; Q_A > Q_{A,out} \\ 0 \text{ otherwise} \end{cases} \tag{35}$$

The CSTR is assumed to be at constant volume so that the total inlet flow equals the flow out (Q_{out}) at all times (36).

$$Q_{out} = Q_{A,out} + Q_{A,oover} + Q_B \tag{36}$$

The kinetics in the tank are first order in both A and B and the rate law (37) has temperature dependence subject to the Arrhenius equation, where R_A is the rate of reaction of component A,

k_0 is the reaction rate constant, E_A is the activation energy, R is the ideal gas constant, T is the temperature in the tank, C_A and C_B are the concentrations of component A and B, respectively.

$$R_A = k_0 e^{\frac{-E_A}{RT}} C_A C_B \tag{37}$$

The tank temperature is determined by an energy balance on the tank (38), where q_{heat} is the rate that heat is delivered to the tank, V is the CSTR volume, ρ and C_P are the density and the heat capacity, respectively of the fluid in the system, and the subscript 0 refers to the fluid before it enters the tank. The components A, B, and C are all assumed dilute so that their concentrations do not affect the density, heat capacity, or overall material balances of the solution. This assumption also permits neglecting heat of reaction in the energy balance.

$$\rho V C_P \frac{dT}{dt} = \rho C_P Q_{out} \left(T_0 - T\right) + q_{heat} \tag{38}$$

Material balances on each component are also computed, giving three more differential equations (39)-(41), where C_C is the concentration of component C.

$$V \frac{dC_A}{dt} = \left(Q_{A,out} + Q_{A,over}\right) C_{A0} - Q_{out} C_A - R_A V \tag{39}$$

$$V \frac{dC_B}{dt} = Q_B C_{B0} - Q_{out} C_B - R_A V \tag{40}$$

$$V \frac{dC_C}{dt} = 2R_A V - Q_{out} C_C \tag{41}$$

The MPC problem seeks to minimize deviations from the set point for C_C subject to disturbances in $Q_{A,in}$ without making drastic control moves. To achieve this trade-off, a quadratic performance index is used where the squared deviations at the end of each time interval are weighted differently (10 for set point deviations and 1 for manipulated variable changes) and summed to create a performance index to be minimized. This yields the dynamic optimization problem in (42), which is subject to the system model in (33) through (41) and inequality constraints on the inputs.

$$\min_{Q_B, q_{heat}} = \sum_{i=1}^{N_t} 10\left(C_{C,i} - C_{C,SP}\right)^2 + \sum_{i=1}^{N_t} 1\left(Q_{B,i} - Q_{B,i-1}\right)^2 + \sum_{i=1}^{N_t} 1\left(q_{heat,i} - q_{heat,i-1}\right)^2 \tag{42a}$$

Subject to (33)-(41)

$$0 \le Q_B \le Q_{B,max} \tag{42b}$$

$$0 \le q_{heat} \le Q_{heat,max} \tag{42c}$$

A first order hold is used for the manipulated variables (MVs) where the value of these variables is held constant over each time interval. A total of N_t time intervals are used in the model prediction. As Figure 14 shows, the controller checks the most recent state measurements (concentrations and temperature in the CSTR and fluid height in the surge tank) and disturbance measurements (flow of A) at each time step in order to update the model and ensure accurate future predictions. The model with built-in logic for surge tank overflow allows the controller to anticipate large influxes of flow and proactively account for this disturbance.

The optimization problem posed in (42) is solved using both a sequential and a simultaneous solution method. In this problem N_t=30 over 1 minute time intervals with a control horizon equal to the prediction horizon of 30 minutes. With two MVs, the optimization problem has 60 degrees of freedom in total. The sequential method version of the problem uses an optimization solver (FMINCON) in MATLAB [29], which takes pre-determined values of the inputs, simulates the system using an explicit ODE integrator (ODE45), computes the objective function and uses this information to construct numeric approximations to the gradient matrices to compute a new search direction for the next iteration. The sequential method also uses if/then logic as in (28b) to describe the changing dynamics of the surge tank. This methodology requires simulating through the entire time horizon of the system model thousands of times in order to generate the gradient matrices and iterate.

The simultaneous version of the problem is solved using APMonitor with the greater than/equal to MPEC described in (29), which, combined with the built-in orthogonal collocation scheme in APMonitor, allows the problem to be expressed entirely as a set of algebraic equations and inequality constraints, which can be solved using an NLP. The APOPT solver is again used to obtain a solution to this NLP. This method does not require multiple simulations of the system model as it solves the constraints of the system simultaneously subject to minimization of the objective function. As opposed to the sequential approach, the simultaneous method converges the equation residuals only once at the optimal solution.

The MPC problem is solved with the system initially at steady state with $Q_{A,in}$=$Q_{B,in}$=0.5 m^3/min and C_C exactly on set point at 3 mol/m^3. At time t=0, however, a step change disturbance

is introduced, changing $Q_{A,in}$ to 0.8 m^3/min. The results from each solution method showing the controlled variable (CV) and the MVs are shown in Figure 15. As the figure shows, despite the introduction of a large disturbance, the CV is maintained very near its set point in each case. There are slight differences in the solutions using the sequential method relying more heavily on Q_B control moves and the simultaneous method relying more heavily on q_{heat} control moves. The optimization results of the two methods are compared in Table 1, which shows that the sequential method produces a slightly better objective function, but requires 300 times more computational effort. The simultaneous method has negligible computation time for an MPC problem with a one minute time interval, indicating that this MPC scheme could be implemented with no concerns on completing the solution within the required cycle time. In contrast, it would be difficult to implement a real-time MPC application with the sequential method due to the computation time exceeding the time interval in used in the MPC problem.

Table 1: Computational results from the sequential and simultaneous solution methods. Computations for each method are executed using an Intel ® Core 2 Duo ™ (2.54 GHz) processor with 4 GB RAM.

	Sequential	Simultaneous
Objective function value	0.0094	0.0108
System model evaluations	3,336	1
Computation time (s)	331.6	1.1

The profiles of some relevant state variables are shown in Figure 16 for the simultaneous solution method. As these plots indicate, the continuous logic formulation produces the desired switching behavior with no issues. As the surge tank reaches its overflow condition, the tank overflows but otherwise, $Q_{A,over}$=0. In this MPC application, it is invaluable to have the overflow condition represented in the model, as it allows the controller to anticipate large interruptions to the operation of the CSTR. While the disturbance it introduced at t=0, its major impact is not observed until t=18 min when the tank overflows. The model however, allows for this change to be predicted and control moves to be made pre-emptively. As Figure 15 shows, more drastic control moves are made several minutes before the tank overflows. Predicting this occurrence with a logic-embedded model allows the system to effectively maintain its set point despite the large change in operating conditions.

6. Conclusions and Future Work

This work demonstrates how logical expressions based on a greater than/equal to (≥) or less than/equal to (≤) operator can be used in NMPC. As opposed to prior work, the new method is well-conditioned at the switching point, leading to a unique solution with improved convergence properties. These equations, known as MPECS, can be embedded into a DAE model using only continuous algebraic equations. The MPECs take advantage of complementarity conditions, requiring that at least one of a set of two inequality constraints be active at all times. Two simulation examples have been presented to demonstrate the viability of using MPECs to represent these logical decisions. The examples, as presented, demonstrate rapid and accurate convergence, illustrating how a logical operating scheme can be simulated using an efficient simultaneous solution method.

In addition to simulation, an NMPC problem is also solved using the formulation developed in this work. The simultaneous solution method combined with the continuous logic formulation is compared to a sequential method using simple if/then logic. The results show that the methods produce nearly equivalent solutions. However, the simultaneous method with continuous logic is 300 times faster in obtaining a solution. The continuous logic formulations allow implementation of logical statements into a model without having to resort to the less efficient sequential method for real-time NMPC or dynamic optimization calculations. The model including the dynamics and the logical statements are implemented as a continuous system of algebraic equations, which can be solved with efficient NLP solvers.

While the examples posed in this work demonstrate the potential of using MPECs for logical decisions, this nascent topic requires much more research to be a viable method for solving optimization problems with such decisions. One of the key challenges to overcome is the non-convexity that is characteristic of many problems with logical decisions like this, which causes the optimizer to converge to local solutions. Furthermore, the mathematical properties of logical MPECs must be studied to provide a better understanding of how these problems are handled by various solvers and what can be done to further enhance performance. In particular, in the examples in this paper, the logical conditions are dependent on pre-determined inputs. Optimality is more difficult to obtain when the logical statements depend on the decision variables, with the optimizer typically finding a feasible solution and stopping. This issue is one

that requires further understanding of how a solver deals with continuous logic dependent on decision variables. This paper presents the concept of using MPECs to represent logical decisions when using a simultaneous solution method so that this concept may be explored for other applications.

7. WORKS CITED

1. Movahedian, N.; Nobakhtian, S. Necessary and Sufficient Conditions for Nonsmooth Mathematical Programs with Equilibrium Constraints. *Nonlinear Analysis: Theory, Methods & Applications* **2010**, *72*, 2694–2705.
2. Yin, H.; Ding, F.; Zhang, J. Active Set Algorithm for Mathematical Programs with Linear Complementarity Constraints. *Applied Mathematics and Computation* **2011**, *217*, 8291–8302.
3. Tangaramvong, S.; Tin-Loi, F. An FE-MPEC Approach for Limit Load Evaluation in the Presence of Contact and Displacement Constraints. *International Journal of Solids and Structures* **2012**, *49*, 1753–1763.
4. Tangaramvong, S.; Tin-Loi, F.; Senjuntichai, T. An MPEC Approach for the Critical Post-collapse Behavior of Rigid-plastic Structures. *International Journal of Solids and Structures* **2011**, *48*, 2732–2742.
5. Baumrucker, B. T.; Renfro, J. G.; Biegler, L. T. MPEC Problem Formulations and Solution Strategies with Chemical Engineering Applications. *Computers & Chemical Engineering* **2008**, *32*, 2903–2913.
6. Raghunathan, A. U.; Biegler, L. T. Mathematical Programs with Equilibrium Constraints (MPECs) in Process Engineering. *Computers & Chemical Engineering* **2003**, *27*, 1381–1392.
7. Raghunathan, A. U.; Soledad Diaz, M.; Biegler, L. T. An MPEC Formulation for Dynamic Optimization of Distillation Operations. *Computers & Chemical Engineering* **2004**, *28*, 2037–2052.
8. Gabriel, S. A.; Leuthold, F. U. Solving Discretely-constrained MPEC Problems with Applications in Electric Power Markets. *Energy Economics* **2010**, *32*, 3–14.
9. Baumrucker, B. T.; Biegler, L. T. MPEC Strategies for Optimization of a Class of Hybrid Dynamic Systems. *Journal of Process Control* **2009**, *19*, 1248–1256.
10. Baumrucker, B. T.; Biegler, L. T. MPEC Strategies for Cost Optimization of Pipeline Operations. *Computers & Chemical Engineering* **2010**, *34*, 900–913.
11. Hedengren, J. D. MPEC: Mathematical Programs with Equilibrium Constraints http://apmonitor.com/wiki/index.php/Apps/MpecExamples.
12. Björkqvist, J.; Westerlund, T. Automated Reformulation of Disjunctive Constraints in MINLP Optimization. *Computers & Chemical Engineering* **1999**, *23, Supplement*, S11–S14.
13. Grossmann, I. E. Review of Nonlinear Mixed-Integer and Disjunctive Programming Techniques. *Optimization and Engineering* **2002**, *3*, 227–252.
14. Grossmann, I. E.; Türkay, M. Solution of Algebraic Systems of Disjunctive Equations. *Computers & Chemical Engineering* **1996**, *20, Supplement 1*, S339–S344.
15. Liu, G. S.; Zhang, J. Z. A New Branch and Bound Algorithm for Solving Quadratic Programs with Linear Complementarity Constraints. *Journal of Computational and Applied Mathematics* **2002**, *146*, 77–87.
16. Sawaya, N. W.; Grossmann, I. E. A Cutting Plane Method for Solving Linear Generalized Disjunctive Programming Problems. *Computers & Chemical Engineering* **2005**, *29*, 1891–1913.
17. Biegler, L. T. An Overview of Simultaneous Strategies for Dynamic Optimization. *Chemical Engineering and Processing: Process Intensification* **2007**, *46*, 1043–1053.
18. Tanarkit, P.; Biegler, L. T. Stable Decomposition for Dynamic Optimization. *Industrial & Engineering Chemistry Research* **1995**, *34*, 1253–1266.

19. Hedengren, J. D.; Allsford, K.; Ramlal, L. Moving Horizon Estimation and Control for an Industrial Gas Phase Polymerization Reactor. *Proceedings of the American Control Conference* **2007**, 1353–1358.
20. Leibman, M. J.; Edgar, T. F.; Lasdon, L. S. Efficient Data Reconciliation and Estimation for Dynamic Processes Using Nonlinear Programming Techniques. *Computers & Chemical Engineering* **1992**, *16*, 963–986.
21. Spivey, B. J.; Hedengren, J. D.; Edgar, T. F. Constrained Nonlinear Estimation for Industrial Process Fouling. *Industrial and Engineering Chemistry Research* **2010**, *49*, 7824–7831.
22. Carey, G. F.; Finlayson, B. A. Orthogonal Collocation on Finite Elements. *Chemical Engineering Science* **1975**, *30*, 587–596.
23. Finlayson, B. A. Orthogonal Collocation on Finite Elements—progress and Potential. *Mathematics and Computers in Simulation* **1980**, *22*, 11–17.
24. Bequette, B. W. Nonlinear Control of Chemical Processes: a Review. *Ind. Eng. Chem. Res.* **1991**, *30*, 1391–1413.
25. Zavala, V. M. Computational Strategies for the Optimal Operation of Large-Scale Chemical Processes. PhD Dissertation, Carnegie Mellon University: Pittsburgh, PA, 2009.
26. Biegler, L. T.; Cervantes, A. M.; Wächter, A. Advances in Simultaneous Strategies for Dynamic Process Optimization. *Chemical Engineering Science* **2002**, *57*, 575–593.
27. Hughes, T. J. R.; Reali, A.; Sangalli, G. Efficient Quadrature for NURBS-based Isogeometric Analysis. *Computer Methods in Applied Mechanics and Engineering* **2010**, *199*, 301–313.
28. Hedengren, J. D. *Advanced Process Monitor*; Advanced Process Monitor, 2012.
29. *MATLAB*; The MathWorks Inc., 2011.

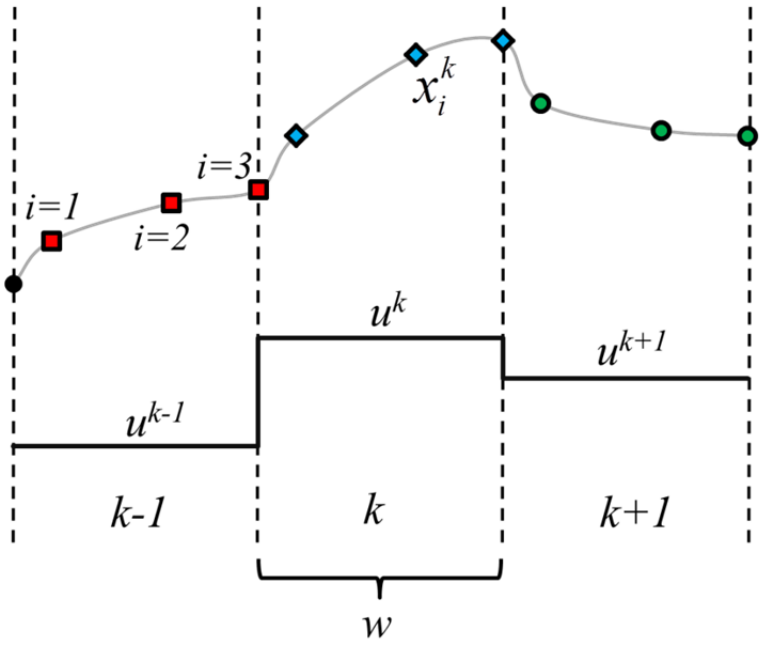

Figure 1: A schematic illustrating the orthogonal collocation on finite elements discretization with a first-order hold assumed for inputs (*u*) in each element (*k*). The differential state variables (*x*) are approximated at each of the collocation points, denoted by *i*. The points are represented using different shapes and colors, which help distinguish one finite element from another.

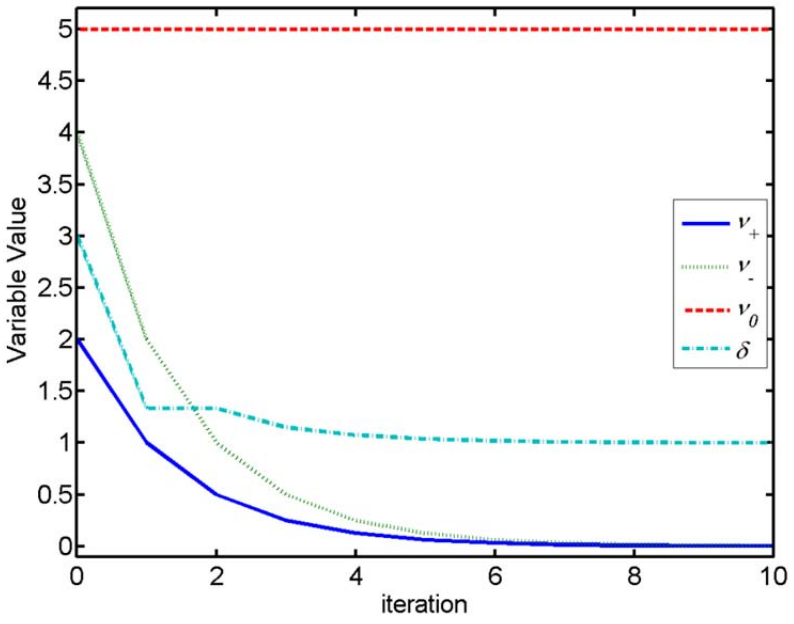

Figure 2: A plot showing the convergence of the greater than/equal to logic MPEC when *x*=0. As the plot shows, δ converges to 1 as desired.

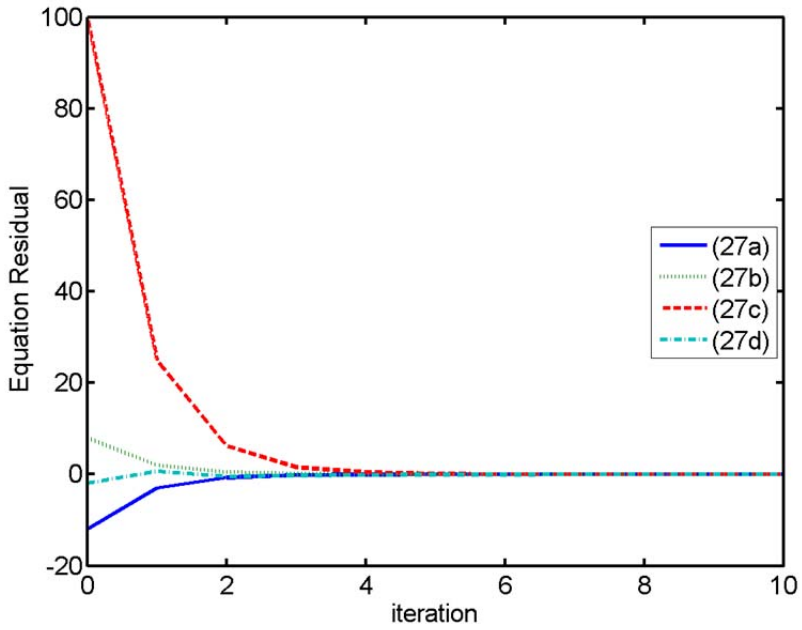

Figure 3: A plot of residuals when solving (27) using Newton's Method when x=0.

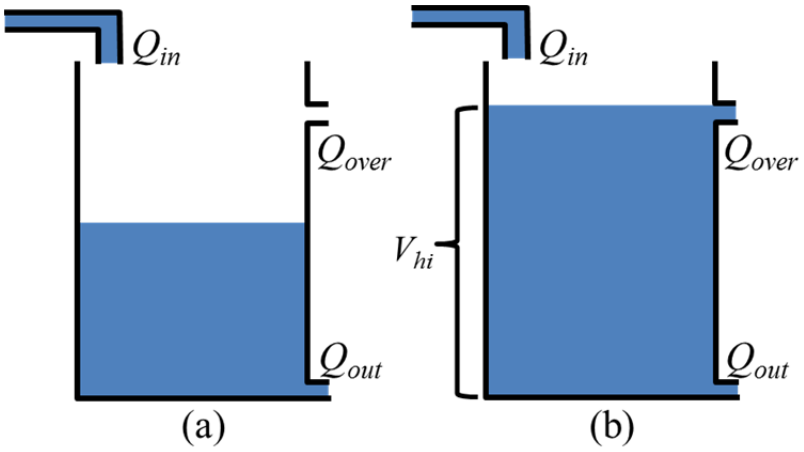

Figure 4: A schematic showing how the dynamic equations representing a simple tank change when the tank overflows.

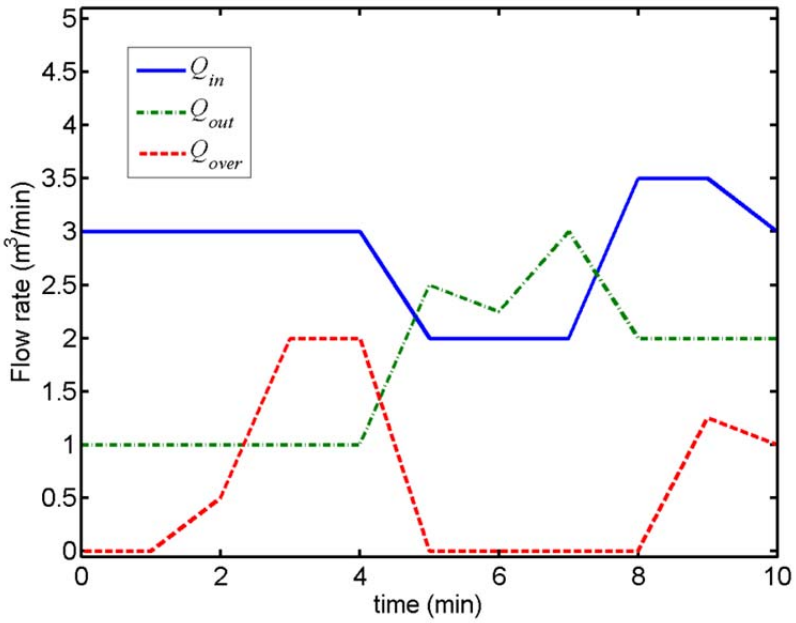

Figure 5: Flow rates in and out of the tank overflow system. Q_{in} and Q_{out} are the model inputs. Q_{over} is a dependent variable, subject to the logical condition of the tank being at its overflow limit.

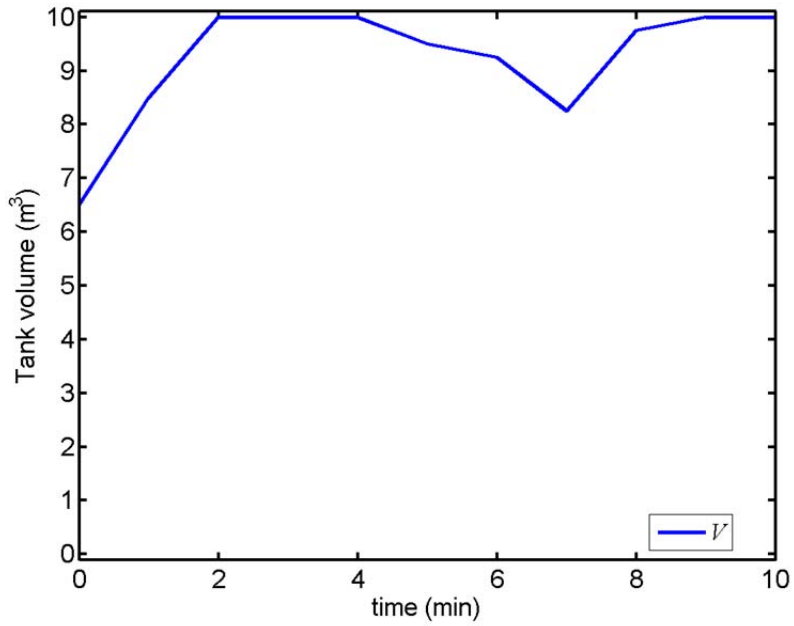

Figure 6: Tank volume with a high limit (V_{max}) of 10 m^3. If the tank volume reaches this limit, overflow may ensue.

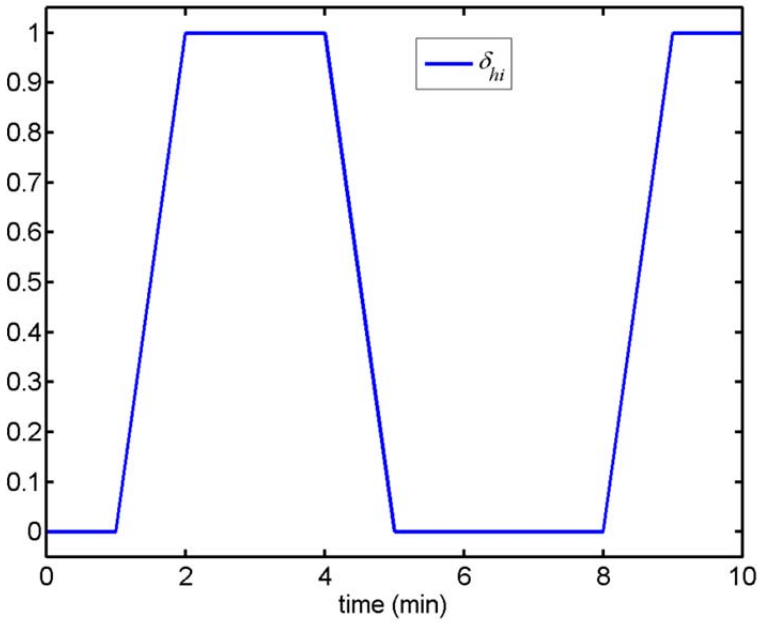

Figure 7: The pseudo-binary variable, δ_{hi}, which is a continuous variable that takes on values of 1 (tank full) and 0 (tank empty) at the solution.

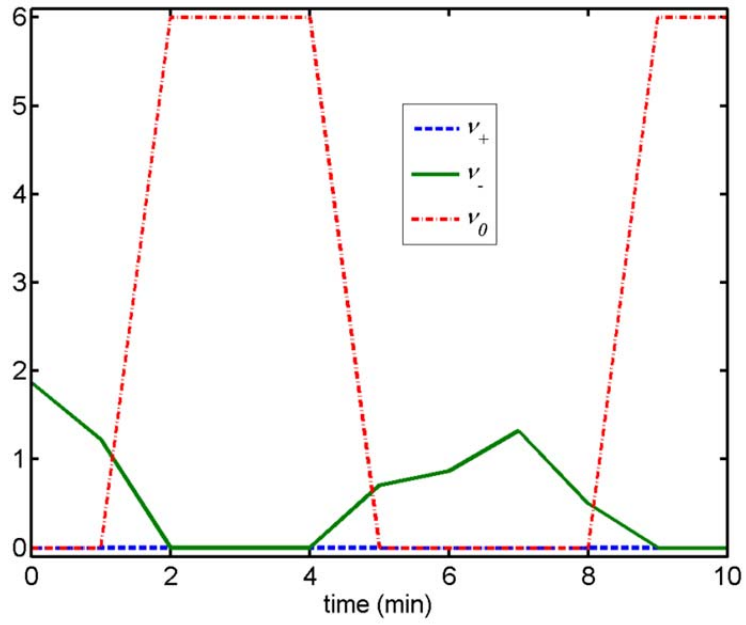

Figure 8: Complementarity variables used in the tank overflow system.

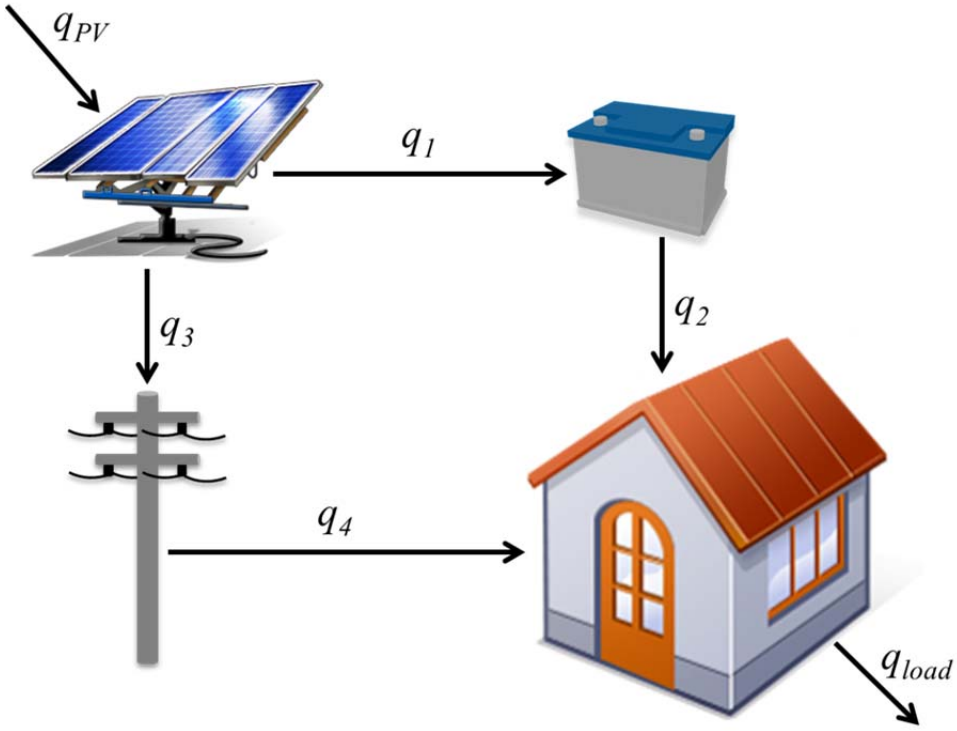

Figure 9: Schematic for the power flow example with photovoltaic panel, battery, electric grid, and a load (represented by the building) with the corresponding flows defined between these elements.

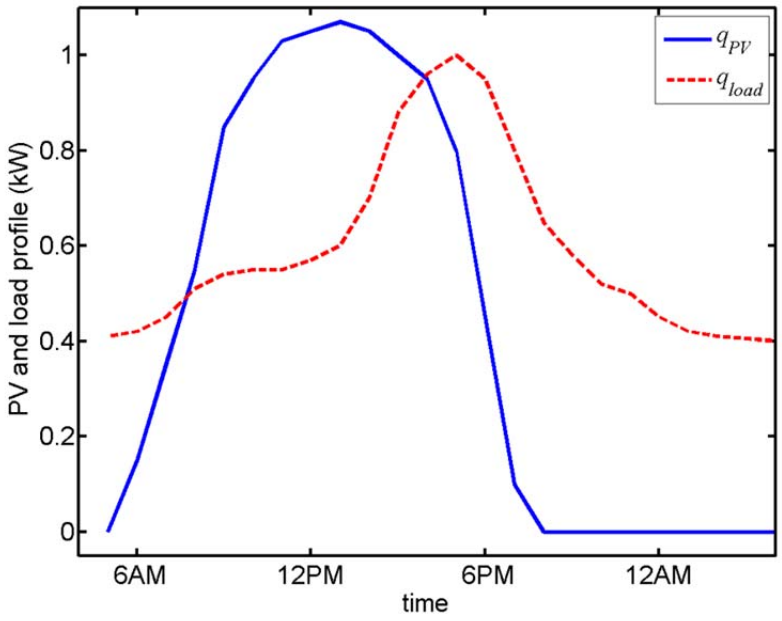

Figure 10: Inputs to the power flow model with q_{pv} (the electric power flow entering the photovoltaic panel) and q_{load} (the power demand of the building).

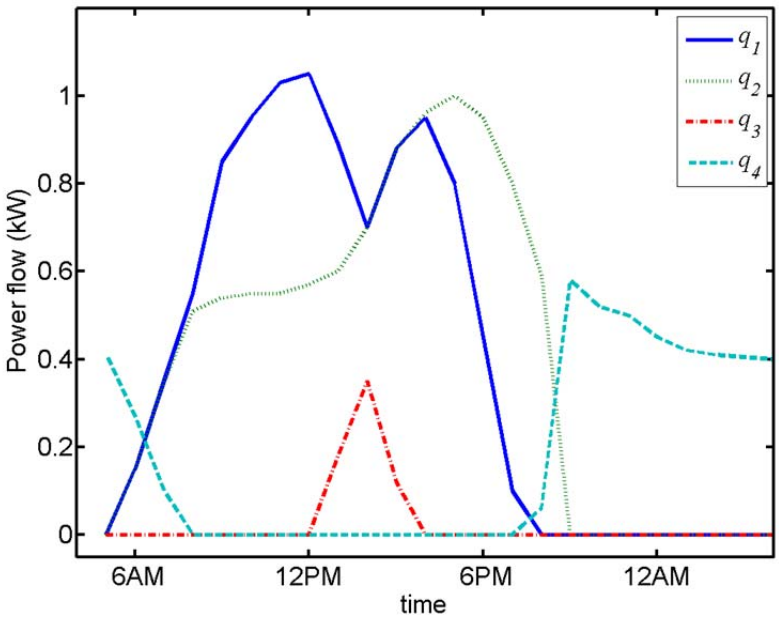

Figure 11: Flows in the power network illustrating the viability of the continuous logic MPEC formulation.

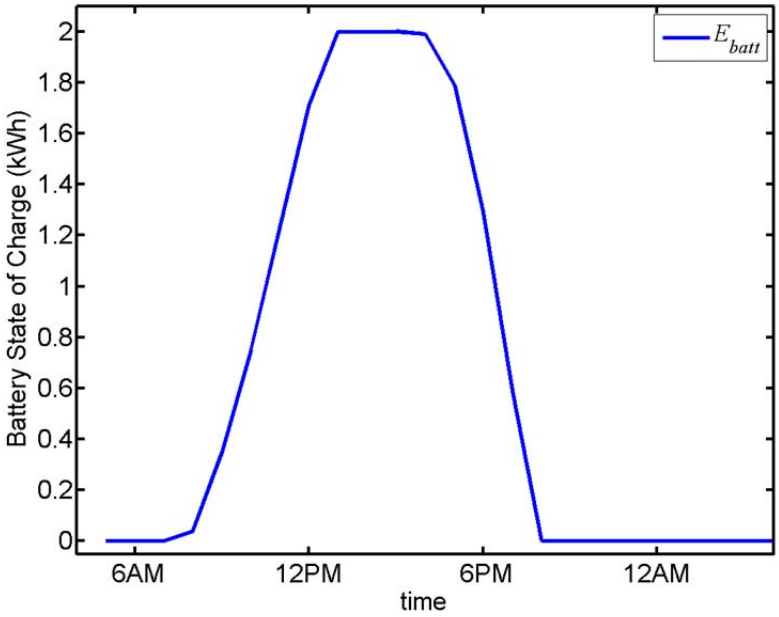

Figure 12: State of charge (kWh) of the battery with an upper limit of 2 kWh.

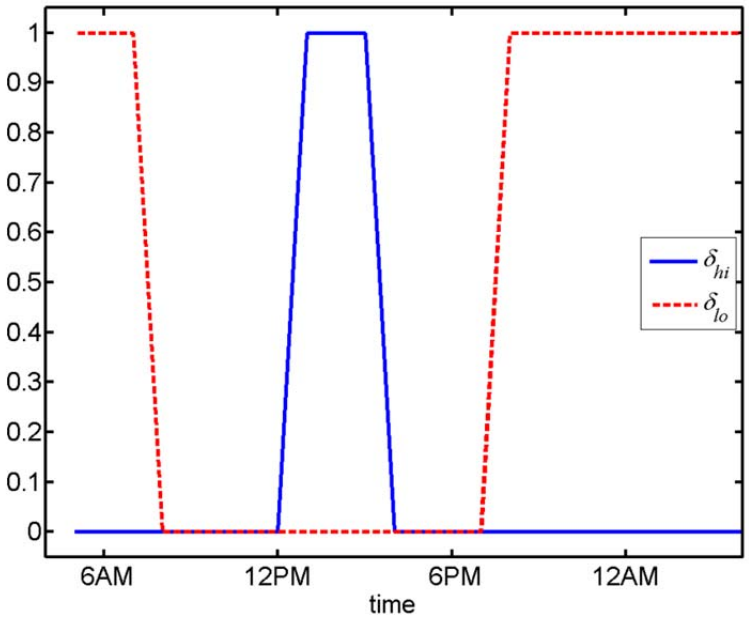

Figure 13: Pseudo-binary variables indicating a fully charged battery (δ_{hi}) and a fully discharged battery (δ_{lo}).

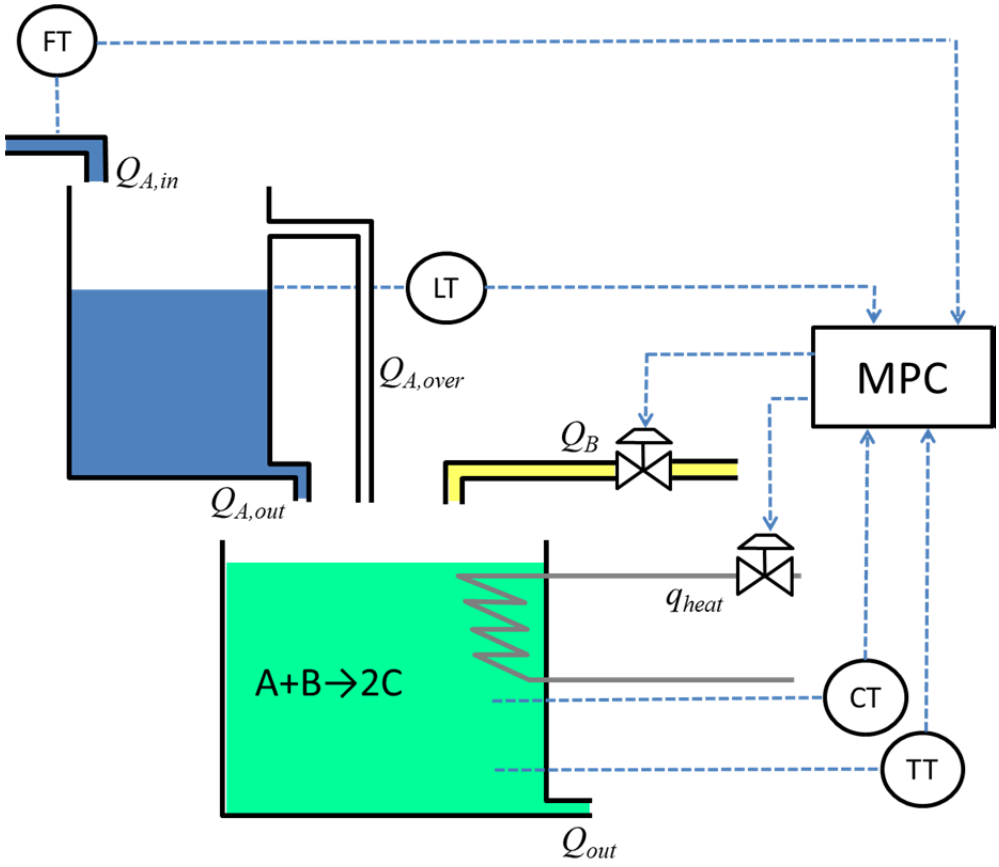

Figure 14: A schematic showing the MPC scheme of a CSTR and surge tank with overflow.

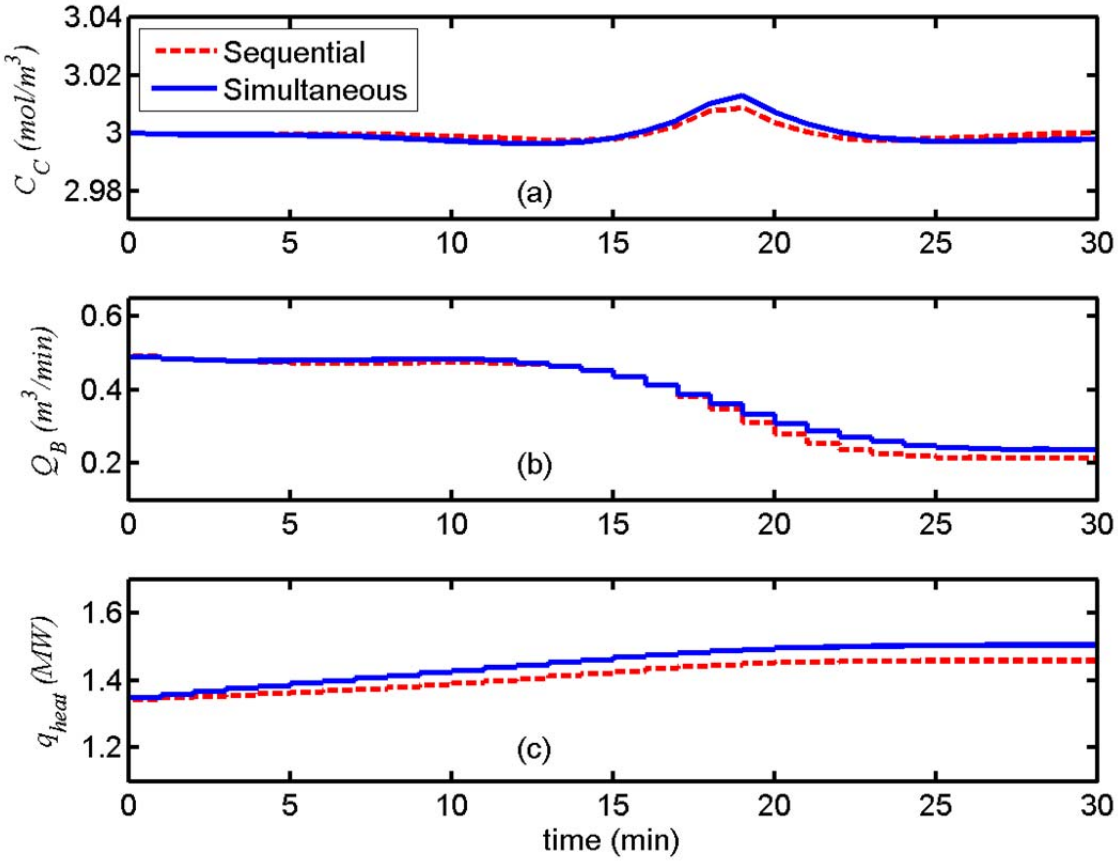

Figure 15: Results from the CSTR with surge tank nonlinear MPC problem showing the solution from the sequential method (red dotted line) with the simultaneous method (blue solid line), where C_C is the controlled variable with a set point of 3 mol/m^3 (a), Q_B and q_{heat} are manipulated variables subject to a zero-order hold.

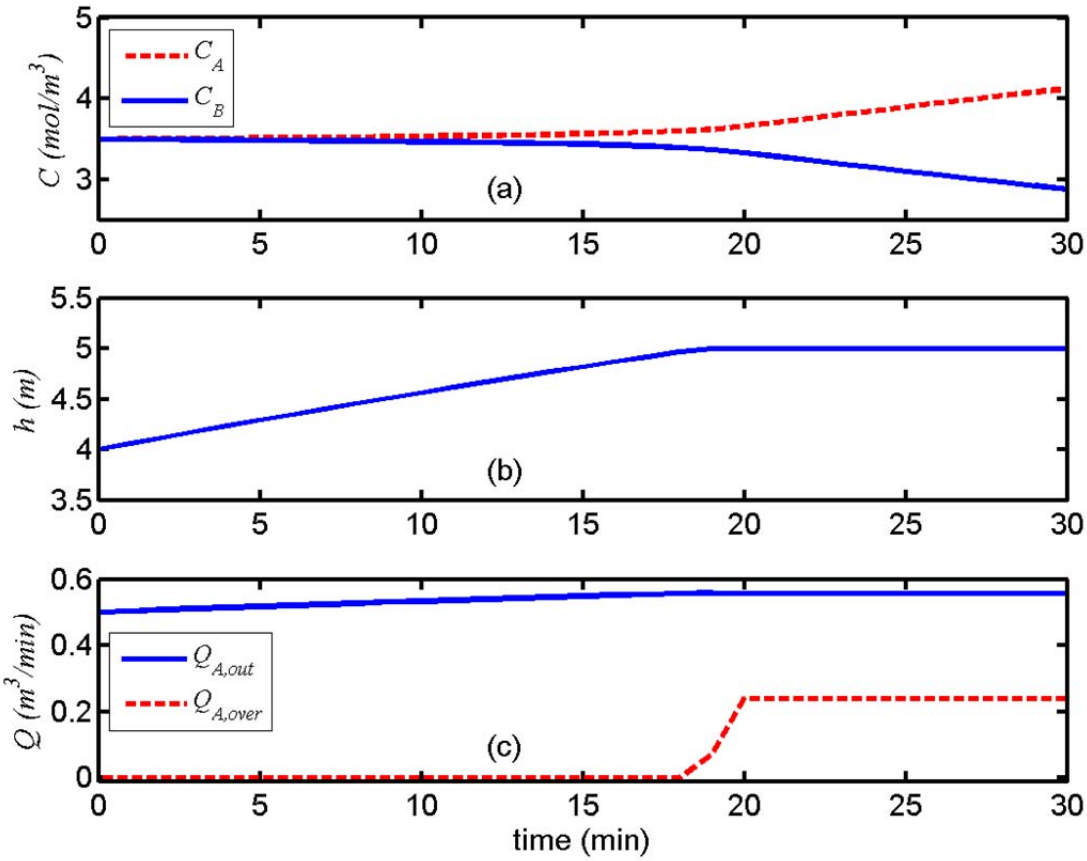

Figure 16: Results of the CSTR MPC problem showing other differential and algebraic state variables with time including the compositions of A and B (a), height of fluid in the surge tank (b), and flow from the surge tank (c).