Design Under Uncertainty Using Parallel Multiperiod Dynamic Optimization

Ian D. Washington and Christopher L. E. Swartz
Dept. of Chemical Engineering, McMaster University, Hamilton, ON, Canada L8S 4L7

DOI 10.1002/aic.14473
Published online May 23, 2014 in Wiley Online Library (wileyonlinelibrary.com)

A technique for optimizing dynamic systems under uncertainty using a parallel programming implementation is developed in this article. A multiple-shooting discretization scheme is applied, whereby each shooting interval is solved using an error-controlled differential equation solver. In addition, the uncertain parameter space is discretized, resulting in a multiperiod optimization formulation. Each shooting interval and period (scenario) realization is completely independent, thus a major focus of this article is on demonstrating potential computational performance improvement when the embedded dynamic model solution of the multiperiod algorithm is implemented in parallel. We assess our parallel multiperiod and multiple-shooting-based dynamic optimization algorithm on two case studies involving integrated plant and control system design, where the objective is to simultaneously determine the size of the process equipment and the control system tuning parameters that minimize cost, subject to uncertainty in the disturbance inputs. © 2014 American Institute of Chemical Engineers *AIChE J*, 60: 3151–3168, 2014
Keywords: dynamic optimization, multiple-shooting discretization, multiperiod formulation, design under uncertainty, parallel computing

Introduction

Dynamic optimization is an important tool for solving practical problems of model predictive control (MPC), moving horizon state estimation, optimal control, parameter estimation, and design with dynamic performance considerations. Practical large-scale dynamic optimization methodologies involve the use of direct techniques whereby partial or full discretization is applied to a dynamic system to yield an algebraic nonlinear program. Within this domain, there are three well-established practical classes of algorithms for continuous dynamic optimization applicable to large-scale systems, namely sequential (i.e., single shooting), simultaneous (i.e., full transcription), and multiple-shooting methods. Sequential methods solve an embedded differential-algebraic equation (DAE) model at each optimization iteration; simultaneous methods fully discretize the process model and then solve the optimization and model simultaneously; multiple shooting is a hybrid of sequential and simultaneous methods that independently solves the embedded model over a number of independent successive intervals, while an outer nonlinear programming (NLP) solver is used to satisfy the particulars of the desired optimization formulation and to ensure that continuity is achieved in the state trajectories between each successive shooting interval. Extensions to include mixed-integer formulations usually apply decomposition techniques and form an iterative solution scheme between master and primal subproblems.[1] All techniques have been applied successfully to a number of large-scale chemical process systems,[2–4] with more detail on the algorithmic aspects given in Biegler.[5]

In terms of advancing the state of direct dynamic optimization solution techniques such that larger plant-wide models can be used, recent work has focused on parallel implementations for: (1) the sensitivity computation within the sequential approach[6]; (2) the embedded model and sensitivity solution over each shooting interval within the multiple-shooting algorithm[7]; and (3) the structured linear algebra of the Karush–Kuhn–Tucker system solution for the interior-point (IP) NLP algorithm, used within the simultaneous method.[8] Furthermore, with scenario-based solution strategies for handling uncertainty, many past implementations have been serial with no exploitation of parallelism for the multiple periods/scenarios used. One exception where parallelization is used is a recent study by Zhu et al.,[9] who demonstrated a multiperiod NLP algorithm (structure exploiting parallel version of IPOPT) for the optimal operation of an air separation process under uncertainty. Their approach considers using a multiperiod algorithm to include parametric uncertainty within the disturbances (discretized into several possible realizations) as opposed to simply using nominal model inputs, and then applies large-scale NLP solution techniques where the underlying linear algebra structure is exploited using a parallelized solution strategy. In another direction, Ricardez-Sandoval[10] has used parallel techniques for Monte Carlo simulation within an integrated design and control framework, whereby independent function evaluations for dynamic model simulation are performed in parallel.

The particular application that we focus on in this work is the integration of design and control problem, which offers a systematic approach to address poor control performance,

Correspondence concerning this article should be addressed to C. L. E. Swartz at swartzc@mcmaster.ca.

violation of safety and environmental constraints, and degradation in economic performance through dynamic optimization-based design formulations. The design of chemical process plants is traditionally performed by first sizing the equipment via calculations under steady-state conditions, followed by the development of a control system structure and corresponding tuning in a subsequent step. It has been well recognized that using such an approach can lead to a process design with unfavorable process dynamics, which can be difficult to adequately control in the presence of unforeseen disturbances or process variability.[11] A more recent strategy has been to design the plant considering dynamic performance by integrating both design and control at the same stage, such that operational variability can be directly addressed at the design stage. The central idea to combining both design and control tasks into a single phase is to permit potential control performance limiting factors to be addressed simultaneously during system design so that designs that are likely to cause control difficulties can be avoided. Detailed discussions of integration techniques for design and control can be found in the comprehensive reviews by van Schijndel and Pistikopoulos,[12] Sakizlis et al.,[13] and more recently by Yuan et al.[14]

To construct a realistic design formulation, uncertainty must be explicitly embedded within the dynamic system that describes possible unknown exogenous disturbance inputs and/or model parameters. Our focus is on a stochastic optimization formulation, for which a straightforward route involves using a scenario-based or multiperiod optimization technique which describes uncertainty through a set of possible parameter scenarios based on a discrete probability distribution or discretized probability functions.

The focus of this article is on the underlying solution techniques utilized in systematic optimization approaches to the integration of design and control. More specifically, this article is on the development of a parallel computing implementation that utilizes a multiperiod approach to handle disturbance uncertainty, and a multiple-shooting dynamic optimization solution technique. The discretization of the infinite dimensional uncertainty space (multiperiod or multiscenario approach), when applied to large-scale DAEs, often yields large and potentially unwieldy systems of equations, which must be solved in an efficient manner for computational tractability. Fortunately, the resulting uncertainty realizations at the DAE level are independent within the state space, which allows the solution to be implemented in a highly parallel manner. Parallelization can be further exploited by adopting a parallel multiple-shooting algorithm for solving the dynamic optimization problem.[7] As with the sequential (single-shooting) approach, multiple shooting readily uses error-based step-size control through the DAE integration routine, but it is less prone to failure when optimization iterates yield values of the optimization variables that result in unstable dynamics. Our current algorithm decomposes the differential equations within the optimization formulation by partitioning both the number of periods and shooting intervals and off-loading each of these independent integration tasks to a parallel computing cluster, thus significantly speeding up each iteration of the NLP algorithm. The approach is well suited for use as a tool in existing optimization-based design and control frameworks,[15] or as a core solver in robust operational problems of MPC or open-loop robust optimal control.[16]

The article is laid out by first discussing the multiperiod formulation, followed by our dynamic optimization solution approach and implementation. Next, two case studies of different scale are presented to illustrate the computational performance of the algorithm. Finally, some concluding remarks are provided which highlight the benefits of the proposed algorithm.

Dynamic Multiperiod Optimization

Multiperiod or multiscenario optimization is a commonly used technique to approximate stochastic programs whereby an infinite dimensional (continuous) stochastic program is reformulated as a discrete-time problem such that the probability distribution is discretized at several points in the random variable space.[17] This type of formulation has been widely applied in the context of plant-wide design and scheduling of chemical processes under uncertainty.[18,19] Steady-state process models are typically used whereby the objective is to optimize an economic criterion while ensuring that the plant can operate under several possible conditions arising in a sequence of possibly different time periods.

The extension of multiperiod formulations to include system dynamics, and hence optimal control formulations, has been addressed by a number of researchers. Ruppen et al.[20] discuss batch reactor trajectory optimization under parametric uncertainty and utilize a simultaneous-based optimization scheme with a successive linear programming solution strategy to solve the resulting NLP. Bhatia and Biegler[21] discuss the design and scheduling of batch process plants and utilize sequential quadratic programming (SQP) for the resulting NLP. The authors' focus was primarily on the optimization formulation to facilitate the integration of design and scheduling. In a subsequent article, Bhatia and Biegler[22] develop and explore NLP algorithms to address the resulting large-scale NLP produced from the multiperiod formulation using IP techniques. From a design under uncertainty perspective, Mohideen et al.[15] proposed a framework for the integration of design and control which uses an iterative decomposition algorithm involving two optimization stages comprising a mixed-integer multiperiod dynamic optimization design problem and a feasibility analysis problem which is also a mixed-integer dynamic optimization problem.

Many of the past studies that utilize a multiperiod formulation with dynamic systems have generally considered a relatively small number of scenarios to approximate the expected value of the objective function and constraint satisfaction. To accurately capture the effects of uncertainty on systems with many uncertain parameters requires a large number of scenarios. The introduction of numerous scenarios, particularly in large-scale dynamic systems, raises the issue of computational tractability, which consequently requires an efficient solution implementation. In this work, we explore the use of parallel computing strategies.

Multiperiod formulation

For a dynamic system consisting of differential and algebraic equations over a fixed time horizon, a general stochastic dynamic program can be stated as

$$\begin{aligned}
&\min_{\mathbf{u}(t),\mathbf{p}} \mathcal{J}(\mathbf{u}(t),\mathbf{p}) := E_{\theta\in\Gamma}\{\phi(\mathbf{x}(t_f),\mathbf{z}(t_f),\mathbf{u}(t_f),\mathbf{v}(t_f),\mathbf{p},\boldsymbol{\theta},t_f)\}\\
&\text{st}: \dot{\mathbf{x}}(t)-\mathbf{f}_d(\mathbf{x}(t),\mathbf{z}(t),\mathbf{u}(t),\mathbf{v}(t),\mathbf{p},\boldsymbol{\theta},t)=\mathbf{0}\\
&\qquad \mathbf{f}_a(\mathbf{x}(t),\mathbf{z}(t),\mathbf{u}(t),\mathbf{v}(t),\mathbf{p},\boldsymbol{\theta},t)=\mathbf{0}\\
&\qquad \mathbf{x}(t_0)-\mathbf{h}_0(\mathbf{u}(t_0),\mathbf{v}(t_0),\mathbf{p},\boldsymbol{\theta},t_0)=\mathbf{0}\\
&\qquad \mathbf{g}(\mathbf{x}(t),\mathbf{z}(t),\mathbf{u}(t),\mathbf{v}(t),\mathbf{p},\boldsymbol{\theta},t)\le\mathbf{0}\\
&\qquad \mathbf{u}(t)\in U,\mathbf{p}\in P,t\in T
\end{aligned} \tag{P1}$$

where the expectation function is defined as

$$E_{\boldsymbol{\theta}\in\Gamma}\{\phi(\cdot)\}=\int_{\boldsymbol{\theta}\in\Gamma} P(\boldsymbol{\theta})\cdot\phi(\mathbf{x}(t_f),\mathbf{z}(t_f),\mathbf{u}(t_f),\mathbf{v}(t_f),\mathbf{p},\boldsymbol{\theta},t_f)\mathrm{d}\boldsymbol{\theta} \tag{1}$$

Furthermore, we define the sets $X\subseteq\mathbb{R}^{n_x}$ and $Z\subseteq\mathbb{R}^{n_z}$ for differential and algebraic state variables, $U=[\mathbf{u}^L,\mathbf{u}^U]\subset\mathbb{R}^{n_u}$ for the open-loop control variables, $V\subseteq\mathbb{R}^{n_v}$ for input disturbance variables, $P=[\mathbf{p}^L,\mathbf{p}^U]\subset\mathbb{R}^{n_p}$ for optimization design parameters, $\Gamma\subseteq\mathbb{R}^{n_\theta}$ for uncertain parameters, and $T=[t_0,t_f]\subset\mathbb{R}$ for the independent time domain. Accordingly, $\phi(\cdot): X\times Z\times U\times V\times P\times\Gamma\times T\mapsto\mathbb{R}$ represents a scalar objective function criterion, evaluated at the final time, that could possibly represent economic, state/output tracking, or a combination thereof; $\mathbf{f}_d(\cdot): X\times Z\times U\times V\times P\times\Gamma\times T\mapsto\mathbb{R}^{n_x}$ and $\mathbf{f}_a(\cdot): X\times Z\times U\times V\times P\times\Gamma\times T\mapsto\mathbb{R}^{n_z}$ represent the differential and algebraic functions, respectively, of the DAE model in semi-explicit form, assumed to be index-1 such that the Jacobian of $\mathbf{f}_a(\cdot)$ with respect to $\mathbf{z}(t)$ is nonsingular; $\mathbf{g}(\cdot): X\times Z\times U\times V\times P\times\Gamma\times T\mapsto\mathbb{R}^{n_g}$ are path inequality constraints (possibly including a subset of interior-point or endpoint inequality constraints on closed-loop control performance, open-loop inputs, and/or design parameters); $\mathbf{u}(t)\in U$ are time-variant control input variables; $\mathbf{v}(t)\in V$ are time-variant disturbance input variables (typically defined *a priori*); and $\mathbf{p}\in P$ are time-invariant model/design parameters; $P(\boldsymbol{\theta}): \Gamma\mapsto\mathbb{R}$ is a continuous probability density function that represents the uncertainty distribution and $\boldsymbol{\theta}\in\Gamma=\{\boldsymbol{\theta}\in\mathbb{R}^{n_\theta}:\boldsymbol{\theta}^L\le\boldsymbol{\theta}\le\boldsymbol{\theta}^U\}$ are time-invariant parameters that lie within this uncertainty region around some nominal/mean value.

The infinite dimensional formulation given by problem P1 can be discretized using a number of scenarios with an associated probability of occurrence (uncertainty space sampling). Accordingly, the objective function can be approximated as a weighted sum of different possible scenarios sampled from a particular distribution. As such, we can formulate problem P2 by considering $\boldsymbol{\theta}\in\Gamma\equiv\{\boldsymbol{\theta}^{(i)}\}_{i=1}^{n_s}$ in a multiperiod (or multiscenario) form defined by a specified number of uncertain parameter scenarios, n_s (realizations)

$$\begin{aligned}
&\min_{\mathbf{u}^{(i)}(t)\,\forall\, i\in\mathcal{S},\,\mathbf{p}} \mathcal{J}(\mathbf{u}^{(i)}(t),\mathbf{p})\\
&:=\textstyle\sum_{i\in\mathcal{S}} \mathsf{w}_i\cdot\phi_i(\mathbf{x}^{(i)}(t_f),\mathbf{z}^{(i)}(t_f),\mathbf{u}^{(i)}(t_f),\mathbf{v}^{(i)}(t_f),\mathbf{p},\boldsymbol{\theta}^{(i)},t_f)\\
&\text{st}: \quad \dot{\mathbf{x}}^{(i)}(t)-\mathbf{f}_d(\mathbf{x}^{(i)}(t),\mathbf{z}^{(i)}(t),\mathbf{u}^{(i)}(t),\mathbf{v}^{(i)}(t),\mathbf{p},\boldsymbol{\theta}^{(i)},t)=\mathbf{0}\\
&\qquad\qquad \mathbf{f}_a(\mathbf{x}^{(i)}(t),\mathbf{z}^{(i)}(t),\mathbf{u}^{(i)}(t),\mathbf{v}^{(i)}(t),\mathbf{p},\boldsymbol{\theta}^{(i)},t)=\mathbf{0}\\
&\qquad\qquad \mathbf{x}^{(i)}(t_0)-\mathbf{h}_0(\mathbf{u}^{(i)}(t_0),\mathbf{v}^{(i)}(t_0),\mathbf{p},\boldsymbol{\theta}^{(i)},t_0)=\mathbf{0}\\
&\qquad\qquad \mathbf{g}(\mathbf{x}^{(i)}(t),\mathbf{z}^{(i)}(t),\mathbf{u}^{(i)}(t),\mathbf{v}^{(i)}(t),\mathbf{p},\boldsymbol{\theta}^{(i)},t)\le\mathbf{0}\\
&\qquad\qquad \mathbf{u}^{(i)}(t)\in U,\ \mathbf{p}\in P,\ t\ \in T \quad \forall\, i\in\mathcal{S}
\end{aligned} \tag{P2}$$

where the states $\mathbf{x}^{(i)}(t)$, $\mathbf{z}^{(i)}(t)$, open-loop controls $\mathbf{u}^{(i)}(t)$, disturbances $\mathbf{v}^{(i)}(t)$, and uncertain parameters $\boldsymbol{\theta}^{(i)}$ are associated with a particular scenario i, and the design or model parameters $\mathbf{p}$ are uniform over all scenarios. The weight (or probability) associated with each scenario i is represented as $\mathsf{w}_i\in[0,1]$. This particular formulation, where we associate the control variables with each scenario, is of the form of a two-stage stochastic program. The parameters $\mathbf{p}$ constitute first-stage decisions, and the control inputs $\mathbf{u}^{(i)}(t)$ constitute second-stage decisions that can provide compensatory action in response to disturbance and parameter realizations. Due to the discretization approach, the NLP formulation (constraint gradient/incident matrices) has a distinct block structure whereby each scenario realization represents a block.[18] General NLP techniques to exploit block structured formulations and associated software tools are currently the subject of much research.[8,23] The approach proposed in this article does not focus on structure exploitation at the NLP level, but instead takes advantage of the decomposed model structure that is embedded within the multiple-shooting approach. This permits the use of existing NLP solvers, with expensive model function evaluations (i.e., DAE solution) handled in an efficient manner via parallelization.

A popular technique for generating scenarios is Monte Carlo sampling, which entails sampling from probability distributions describing the uncertain parameters. A realization for each parameter, taken together, constitutes a scenario, and each scenario is assigned an equal weight with the sum of weights equal to one.[24] Several alternative sampling techniques have been proposed, with the goal of maintaining good approximation accuracy with increased computational efficiency; some discussion of such approaches is given in Diwekar.[25]

Design and control formulation

In the context of design and control, the optimization parameters $\mathbf{p}$ defined in problem P2 refer to the equipment sizes (e.g., reactor, distillation column, or heat exchanger dimensions), controller tuning parameters and process setpoints. Furthermore, the objective function would typically be an economic index of the combined annualized capital and operating costs of the process plant. As a result, the particular form of the objective function differs slightly from problem P2, in that one is able to partition the overall cost into a design or capital cost portion, $C_{\mathrm{cap}}(\mathbf{p})$, as a function of the design parameters $\mathbf{p}$, and a control or operation cost portion, $C_{\mathrm{op}}(\mathbf{x}(t),\mathbf{z}(t),\mathbf{u}(t),\mathbf{v}(t),\mathbf{p},\boldsymbol{\theta},t)$, as a function of the system response $\mathbf{x}(t)$ or $\mathbf{z}(t)$, manipulated inputs $\mathbf{u}(t)$, specified disturbances $\mathbf{v}(t)$, design parameters $\mathbf{p}$, and uncertain parameters $\boldsymbol{\theta}$. Accordingly, the objective function can be stated as follows

$$\begin{aligned}
\mathcal{J}(\mathbf{p}) :=& C_{\mathrm{cap}}(\mathbf{p})+\textstyle\sum_{i\in\mathcal{S}}\mathsf{w}_i\cdot C_{\mathrm{op}}(\mathbf{x}^{(i)}(t_f),\,\mathbf{z}^{(i)}(t_f),\\
&\mathbf{u}^{(i)}(t_f),\,\mathbf{v}^{(i)}(t_f),\,\mathbf{p},\,\boldsymbol{\theta}^{(i)},\,t_f)
\end{aligned} \tag{2}$$

where the manipulated control inputs, $\mathbf{u}(t)$, are determined via an embedded control law and the time-varying disturbance inputs can be specified directly, via $\mathbf{v}(t)$, or parameterized through the time-invariant parameters, $\boldsymbol{\theta}$, as stepwise profiles according to

$$\mathbf{v}(t;\boldsymbol{\theta})=\mathbf{v}_0+\Delta\mathbf{v}\,\eta(t,t_{\mathrm{step}}) \tag{3}$$

where the uncertain parameters include the initial disturbance values $\mathbf{v}_0$ and the step magnitude values $\Delta\mathbf{v}$, and can be stated accordingly as $\boldsymbol{\theta}:=(\mathbf{v}_0^\top,\Delta\mathbf{v}^\top)^\top$. The function $\mathbf{v}(t;\boldsymbol{\theta})$ is defined for single-step disturbance inputs and approximates the disturbance trajectories as continuous profiles to avoid discontinuities within the model equations and thus avoid potential integration difficulties or the necessity to stop and restart the integration routine.[26] The approximation is performed using an appropriate smoothing function $\eta(t,t_{\mathrm{step}})$,

which is triggered at t_{step}. One possible smoothing function approximation is via the exponential function defined as

$$\eta(t, t_{\text{step}}) := [1+\exp(-\alpha(t-t_{\text{step}}))]^{-1} \in (0,1) \tag{4}$$

where $\alpha > 0$ is a tuning parameter used to adjust the rate of transition of the step and $t_{\text{step}} > t_0$ is the point in time to initiate the transition. Similar disturbance functions can be derived for multistep profiles or even sinusoidal profiles with an uncertain amplitude and/or frequency.[13]

The system inputs that are defined through the control laws are not independent decision variables. However, they remain scenario dependent, along with the other system states. Controller parameters such as the controller gain, reset time, set-point, and bias in a proportional integral (PI) control law can be treated as being uniform over all scenarios or scenario dependent (by defining a separate set of parameters for each scenario). The most appropriate setup would depend on the particular context of the application.

Optimization Solution Framework

As previously noted, direct methods used to solve dynamic optimization problems fall into three major categories: sequential, simultaneous, and multiple-shooting approaches. In this article, we focus on the multiple-shooting approach. It typically results in moderately sized NLP problems, allows for automatic control of integration accuracy through the use of DAE solvers, is less susceptible to failure due to unstable process dynamics than sequential approaches, and is well suited to parallelization.

In this section, we define the multiple-shooting approach, explore the extension to solve multiperiod formulations, and discuss our particular parallel implementation.

Multiple-shooting discretization

The multiple-shooting technique was made widely accessible through the MUSCOD software originally developed by Bock and Plitt[27] for optimal control using ordinary differential equation models, and subsequently adapted for DAE models.[28,29] The technique combines aspects of both the sequential (or single-shooting) and simultaneous methods in that one is able to expose the embedded model states to the optimization layer. The time horizon is partitioned into a number of shooting intervals. Within each interval, the inputs can be further parameterized and the DAE system integrated, much like the sequential method. The multiple-shooting technique entails introducing new optimization parameters to represent the state variable initial conditions at the beginning of each interval and new equality constraints to remove the discrepancy or defect between the state values at the final time from the previous interval and the initial time in the current interval. A general dynamic optimization formulation that utilizes the multiple-shooting discretization applicable to semi-explicit DAE models can be written according to problem P3. Note that to make the presentation of the multiple-shooting formulation less cumbersome, we exclude the functional presence of disturbance variables $\mathbf{v}(t)$ and uncertain parameters $\boldsymbol{\theta}$, as well as the index representing the scenario realizations.

$$\begin{aligned}
&\min_{\boldsymbol{w},\mathbf{p}} \mathcal{J}(\boldsymbol{w},\mathbf{p}) := \phi(\boldsymbol{\xi}_{0,n}, \boldsymbol{\mu}_{0,n}, \mathbf{p}, t_n) \\
&\text{st}: \dot{\mathbf{x}}(t) = \mathbf{f}_d(\mathbf{x}(t), \mathbf{z}(t), \mathcal{U}(t, \boldsymbol{\upsilon}_j), \mathbf{p}, t) && \forall t \in I_j,\ j=0,\ldots,n-1 \\
&\mathbf{0} = \mathbf{f}_a(\mathbf{x}(t),\ \mathbf{z}(t),\ \mathcal{U}(t, \boldsymbol{\upsilon}_j), \mathbf{p}, t) - \boldsymbol{\vartheta}(\boldsymbol{\gamma}_j, t) && \forall t \in I_j,\ j=0,\ldots,n-1 \\
&\mathbf{0} = \mathbf{h}_0(\boldsymbol{\upsilon}_0, \mathbf{p}, t_0) - \boldsymbol{\xi}_{0,0} \\
&\mathbf{x}(t_{j+1}; \boldsymbol{\xi}_{0,j}, \boldsymbol{\mu}_{0,j}, \boldsymbol{\upsilon}_j, \mathbf{p}) - \boldsymbol{\xi}_{0,j+1} = \mathbf{0} && \forall j=0,\ldots,n-1 \\
&\mathbf{f}_a(\boldsymbol{\xi}_{0,j}, \boldsymbol{\mu}_{0,j}, \boldsymbol{\upsilon}_j, \mathbf{p}, t_j) = \mathbf{0} && \forall j=0,\ldots,n \\
&\mathbf{g}(\boldsymbol{\xi}_{0,j}, \boldsymbol{\mu}_{0,j}, \boldsymbol{\upsilon}_j, \mathbf{p}, t_j) \le \mathbf{0} && \forall j=0,\ldots,n \\
&\boldsymbol{w} \in [\boldsymbol{w}^L, \boldsymbol{w}^U],\ \mathbf{p} \in [\mathbf{p}^L, \mathbf{p}^U]
\end{aligned} \tag{P3}$$

In formulation P3, $j=0,\ldots,n-1$ represent each shooting interval for a total of n nodes. The optimization parameters are partitioned into model or design parameters $\mathbf{p}$, and the shooting node parameters for differential/algebraic states and input variable parameters, defined collectively as $\boldsymbol{w} := (\boldsymbol{\xi}_{0,0}^\top, \boldsymbol{\mu}_{0,0}^\top, \boldsymbol{\upsilon}_0^\top, \ldots, \boldsymbol{\xi}_{0,n}^\top, \boldsymbol{\mu}_{0,n}^\top)^\top \in \mathbb{R}^{(n_x+n_z)(n+1)+n_\upsilon n}$. The continuous input vector can be defined using a parameterized function $\mathbf{u}(t) := \mathcal{U}(t, \boldsymbol{\upsilon}_j)$ based on a piecewise approximation within each shooting interval I_j, where $\boldsymbol{\upsilon}_j \in \mathbb{R}^{n_\upsilon}$ represent local polynomial coefficients. For example, it is common practice to use Lagrange interpolation polynomials (see, Biegler,[5] for a detailed discussion). Note that $\boldsymbol{\upsilon}_n$ is used in formulation P3 for notational simplicity, where $\boldsymbol{\upsilon}_n := \boldsymbol{\upsilon}_{n-1}$, and can be removed from the NLP. Additionally, we consider here a fixed end-time formulation where the objective function $\phi(\cdot)$ is represented in Mayer form, which typically only directly depends on the final model states $\boldsymbol{\xi}_{0,n}$, $\boldsymbol{\mu}_{0,n}$, parameters $\mathbf{p}$, and possibly the final time t_n. The DAE model, $\mathbf{F}(\cdot) := \{[\dot{\mathbf{x}}(t) - \mathbf{f}_d(\cdot)]^\top, [\mathbf{f}_a(\cdot) - \boldsymbol{\vartheta}(\boldsymbol{\gamma}_j, t)]^\top\}^\top$, is embedded within the NLP function evaluations and is solved using an appropriate DAE solver for $t \in I_j$, $j=0,\ldots,n-1$ with initial differential state conditions $\mathbf{x}(t_j) := \boldsymbol{\xi}_{0,j}$ and algebraic state conditions $\mathbf{z}(t_j) := \boldsymbol{\mu}_{0,j}$ for all intervals I_j, where the intervals are decoupled using the new parameters and are thus independent from each other. The particular formulation given here relies on the use of a relaxed form of the DAEs using a so-called relaxation function represented here according to the function $\boldsymbol{\vartheta}(\boldsymbol{\gamma}_j, t)$, where $\boldsymbol{\gamma}_j$ is functionally dependent on the shooting parameters at node j. This relaxed form also requires the addition of point equality constraints (for the algebraic model equations) in the NLP at each shooting node to ensure that the original model is obtained upon NLP convergence. More details are described in Leineweber et al.[29] and a recent discussion of possible representations of $\boldsymbol{\vartheta}(\boldsymbol{\gamma}_j, t)$ is provided by Houska and Diehl.[30] Using a relaxed embedded model prevents the necessity for repeated consistent initializations of the DAE model for each shooting interval when using adaptive DAE solvers.[31] In essence, the optimization parameters representing the shooting node algebraic states $\boldsymbol{\mu}_{0,j}$ will always be consistent with the embedded relaxed DAE model $\mathbf{F}(\cdot)$, thus removing the need to modify these variables before integrating the DAE system.

To remove the defect between the initial differential state parameters $\boldsymbol{\xi}_{0,j}$ and the previous interval's final integrated state values $\mathbf{x}(t_{j-1})$, continuity equality constraints are imposed at each shooting node j. The approach to handling inequality path constraints, $\mathbf{g}(\cdot)$, is to approximate them as interior point constraints at each shooting node, which avoids any otherwise necessary reformulation which is typically used in single-shooting approaches. In some cases, this

approximation may not suffice to remove inter-node constraint violation and in such cases the modeler can apply an end-point constraint to the integral of the constraint violation, such as proposed for single-shooting methods.[32] In the final optimization formulation posed to the NLP solver, the differential state variables $\mathbf{x}(t_{j+1})$ appear only within the continuity constraint evaluations, all other constraint and objective function evaluations are with respect to direct optimization parameters. As a result of the direct state variable presence, sensitivity analysis techniques are necessary when deriving the continuity constraint derivatives.

Objective and constraint function derivative evaluation

The gradient of the objective function (in Mayer form) can be evaluated directly at the final shooting node with respect to $\boldsymbol{w}_n := (\boldsymbol{\xi}_{0,n}^\top, \boldsymbol{\mu}_{0,n}^\top)^\top$ and $\mathbf{p}$, and can be stated as

$$\nabla_{\{\boldsymbol{w},\mathbf{p}\}}\mathcal{J}(\boldsymbol{w},\mathbf{p}) := (\mathbf{0}_{n_r}^\top, \mathbf{J}_n^{\boldsymbol{\xi}_{0,n}\top}, \mathbf{J}_n^{\boldsymbol{\mu}_{0,n}\top}, \mathbf{J}_n^{p\top})^\top \in \mathbb{R}^{(n_x+n_z)(n+1)+n_v n+n_p} \tag{5}$$

where $n_r=(n_x+n_z+n_v)n$, and $\mathbf{J}_n^{\boldsymbol{\xi}_{0,n}}$, $\mathbf{J}_n^{\boldsymbol{\mu}_{0,n}}$, and $\mathbf{J}_n^{p}$ represent first derivative vectors making up the NLP objective gradient evaluated at the end-point. These vectors can be either specified analytically or more suitably generated via automatic differentiation at the current optimization parameter iterates.

The multiple-shooting continuity equality constraints, including the initial conditions at t_0, can be defined as

$$\mathbf{c}_0(\boldsymbol{w}_0,\mathbf{p}) \equiv \mathbf{h}_0(\boldsymbol{\upsilon}_0,\mathbf{p}) - \boldsymbol{\xi}_{0,0} = \mathbf{0} \tag{6}$$

$$\mathbf{c}_{j+1}(\boldsymbol{w}_j,\boldsymbol{\xi}_{0,j+1},\mathbf{p}) \equiv \mathbf{x}(t_{j+1};\boldsymbol{\xi}_{0,j},\boldsymbol{\mu}_{0,j},\boldsymbol{\upsilon}_j,\mathbf{p}) - \boldsymbol{\xi}_{0,j+1} = \mathbf{0} \quad \forall j=0,\ldots,n-1 \tag{7}$$

where $\boldsymbol{w}_j := (\boldsymbol{\xi}_{0,j}^\top, \boldsymbol{\mu}_{0,j}^\top, \boldsymbol{\upsilon}_j^\top)^\top$ for $j=0,\ldots,n-1$. The algebraic consistency equality constraints and remaining inequality constraints represent NLP point constraints and can be defined at each shooting node according to

$$\mathbf{q}_j(\boldsymbol{w}_j,\mathbf{p}) \equiv \left(\mathbf{f}_a(\boldsymbol{w}_j,\mathbf{p})^\top, \mathbf{g}(\boldsymbol{w}_j,\mathbf{p})^\top\right)^\top \quad \forall j=0,\ldots,n-1 \tag{8}$$

$$\mathbf{q}_n(\boldsymbol{\upsilon}_{n-1},\boldsymbol{w}_n,\mathbf{p}) \equiv \left(\mathbf{f}_a(\boldsymbol{\upsilon}_{n-1},\boldsymbol{w}_n,\mathbf{p})^\top, \mathbf{g}(\boldsymbol{\upsilon}_{n-1},\boldsymbol{w}_n,\mathbf{p})^\top\right)^\top \tag{9}$$

The combined constraint vector for all NLP constraints can now be stated as

$$\mathbf{C}(\boldsymbol{w},\mathbf{p}) := \begin{bmatrix} \mathbf{c}_0(\boldsymbol{w}_0,\mathbf{p}) \\ \left(\mathbf{q}_0(\boldsymbol{w}_0,\mathbf{p})^\top, \mathbf{c}_1(\boldsymbol{w}_0,\boldsymbol{\xi}_{0,1},\mathbf{p})^\top\right)^\top \\ \vdots \\ \left(\mathbf{q}_j(\boldsymbol{w}_j,\mathbf{p})^\top, \mathbf{c}_{j+1}(\boldsymbol{w}_j,\boldsymbol{\xi}_{0,j+1},\mathbf{p})^\top\right)^\top \\ \vdots \\ \left(\mathbf{q}_{n-1}(\boldsymbol{w}_{n-1},\mathbf{p})^\top, \mathbf{c}_n(\boldsymbol{w}_{n-1},\boldsymbol{\xi}_{0,n},\mathbf{p})^\top\right)^\top \\ \mathbf{q}_n(\boldsymbol{\upsilon}_{n-1},\boldsymbol{w}_n,\mathbf{p}) \end{bmatrix} \in \mathbb{R}^{(n_x+n_q)(n+1)} \tag{10}$$

As a result of the stacked structure of both the constraint functions and optimization parameters $\boldsymbol{w} := (\boldsymbol{w}_0^\top, \ldots, \boldsymbol{w}_n^\top)^\top$, we maintain a sparse diagonal matrix structure within the constraint Jacobian matrix with respect to $\boldsymbol{w}$ and a block vector structure for the derivatives with respect to $\mathbf{p}$. This matrix can be defined as

$$\nabla_{\{\boldsymbol{w},\mathbf{p}\}}\mathbf{C}(\boldsymbol{w},\mathbf{p}) \equiv \begin{bmatrix} -\mathbf{I}_{n_x}\mathbf{0}_{n_x\times n_z}\mathbf{H}_0^{\upsilon_0} & & & & & & & \mathbf{H}_0^p \\ \mathbf{Q}_0^{\xi_{0,0}}\mathbf{Q}_0^{\mu_{0,0}}\mathbf{Q}_0^{\upsilon_0} & & & & & & & \mathbf{Q}_0^p \\ \mathbf{X}_1^{\xi_{0,0}}\mathbf{X}_1^{\mu_{0,0}}\mathbf{X}_1^{\upsilon_0} - \mathbf{I}_{n_x} & & & & & & & \mathbf{X}_1^p \\ & \ddots & & & & & & \vdots \\ & & \mathbf{Q}_j^{\xi_{0,j}}\mathbf{Q}_j^{\mu_{0,j}}\mathbf{Q}_j^{\upsilon_j} & & & & & \mathbf{Q}_j^p \\ & & \mathbf{X}_{j+1}^{\xi_{0,j}}\mathbf{X}_{j+1}^{\mu_{0,j}}\mathbf{X}_{j+1}^{\upsilon_j} - \mathbf{I}_{n_x} & & & & & \mathbf{X}_{j+1}^p \\ & & & \ddots & & & & \vdots \\ & & & & \mathbf{Q}_{n-1}^{\xi_{0,n-1}}\mathbf{Q}_{n-1}^{\mu_{0,n-1}}\mathbf{Q}_{n-1}^{\upsilon_{n-1}} & & & \mathbf{Q}_{n-1}^p \\ & & & & \mathbf{X}_n^{\xi_{0,n-1}}\mathbf{X}_n^{\mu_{0,n-1}}\mathbf{X}_n^{\upsilon_{n-1}} - \mathbf{I}_{n_x}\mathbf{0}_{n_x\times n_z} & & & \mathbf{X}_n^p \\ & & & & & \mathbf{Q}_n^{\upsilon_{n-1}} & \mathbf{Q}_n^{\xi_{0,n}} \; \mathbf{Q}_n^{\mu_{0,n}} & \mathbf{Q}_n^p \end{bmatrix} \tag{11}$$

where $\mathbf{H}_0^{\upsilon_0}$ and $\mathbf{H}_0^p$ are the first derivatives of $\mathbf{h}_0(\cdot)$ with respect to $\boldsymbol{\upsilon}_0$ and $\mathbf{p}$, respectively; while $\mathbf{Q}_j^{\mathcal{M}}$ represent point constraint first derivatives, at each shooting node, defined as

$$\mathbf{Q}_j^{\xi_{0,j}} := \frac{\partial \mathbf{q}_j(\boldsymbol{w}_j,\mathbf{p})}{\partial \boldsymbol{\xi}_{0,j}}; \mathbf{Q}_j^{\mu_{0,j}} := \frac{\partial \mathbf{q}_j(\boldsymbol{w}_j,\mathbf{p})}{\partial \boldsymbol{\mu}_{0,j}}; \quad \mathbf{Q}_j^{\upsilon_j} := \frac{\partial \mathbf{q}_j(\boldsymbol{w}_j,\mathbf{p})}{\partial \boldsymbol{\upsilon}_j}; \quad \mathbf{Q}_j^{p} := \frac{\partial \mathbf{q}_j(\boldsymbol{w}_j,\mathbf{p})}{\partial \mathbf{p}} \tag{12}$$

Again, the derivatives $\mathbf{H}_0^{\upsilon_0}$, $\mathbf{H}_0^p$, and $\mathbf{Q}_j^{\mathcal{M}}$ can be either specified analytically or generated via automatic differentiation. The matrices given by $\mathbf{X}_{j+1}^{\mathcal{M}}$ represent the embedded differential state parameter sensitivities, for each parameter $\mathcal{M} := \{\xi_{0,j}, \mu_{0,j}, \upsilon_j, p\}$, evaluated at the end of each shooting node, and are defined according to

$$\mathbf{X}_{j+1}^{\xi_{0,j}} := \frac{\partial \mathbf{x}(t_{j+1})}{\partial \boldsymbol{\xi}_{0,j}}; \quad \mathbf{X}_{j+1}^{\mu_{0,j}} := \frac{\partial \mathbf{x}(t_{j+1})}{\partial \boldsymbol{\mu}_{0,j}}; \quad \mathbf{X}_{j+1}^{\upsilon_j} := \frac{\partial \mathbf{x}(t_{j+1})}{\partial \boldsymbol{\upsilon}_j}; \quad \mathbf{X}_{j+1}^{p} := \frac{\partial \mathbf{x}(t_{j+1})}{\partial \mathbf{p}} \tag{13}$$

Many large-scale differential equation solvers provide state sensitivities via forward sensitivity analysis[33] or possibly adjoint (reverse) sensitivity analysis,[34] and the particular numerical methods/techniques used to compute this information are highly important in terms of both solution accuracy and speed when using large-scale chemical process engineering models.[6] Due to the direct presence of the state variables $\mathbf{x}(t_{j+1})$ in the continuity constraints, a natural approach to generate the optimization function derivatives is forward sensitivity analysis. We will forgo any further details of sensitivity analysis and automatic differentiation, and direct the reader to Biegler[5] and Griewank and Walther[35] for in depth discussions.

Combined multiperiod multiple-shooting NLP formulation

The fully discretized combined multiperiod multiple-shooting NLP formulation of problems P2 and P3 can now be stated according to

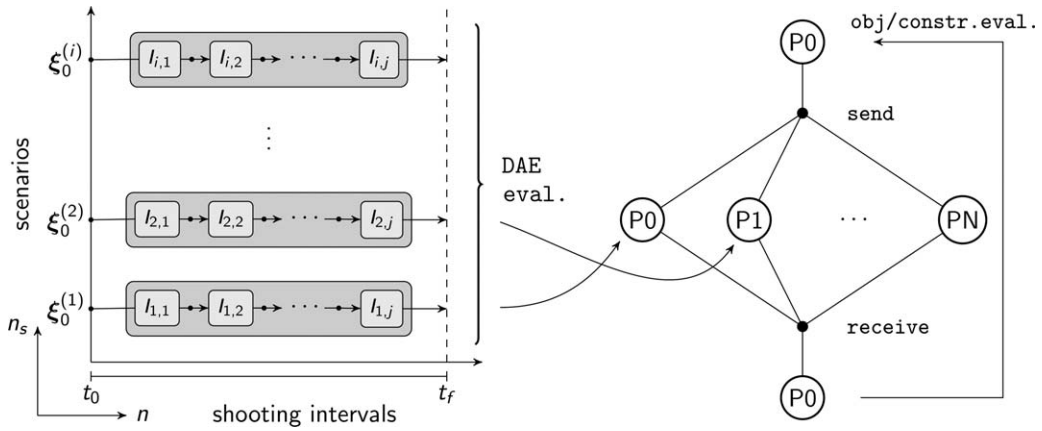

Figure 1. Implementation of DAE solution within the parallel multiperiod multiple-shooting dynamic optimization algorithm.

$$\begin{aligned} \min_{\boldsymbol{s},\mathbf{p}} \ \mathcal{J}(\boldsymbol{s},\mathbf{p}) \ &:= \sum\nolimits_{i\in\mathcal{S}} \mathsf{w}_i \cdot \phi(\boldsymbol{w}_n^{(i)},\mathbf{p},\boldsymbol{\theta}^{(i)}) \\ \text{st}: \ \mathcal{C}^L &\le \mathcal{C}(\boldsymbol{s},\mathbf{p}) \le \mathcal{C}^U \\ \boldsymbol{s} := (\boldsymbol{w}^{(1)\top},\ldots,\boldsymbol{w}^{(n_s)\top})^\top &\in [\boldsymbol{s}^L,\boldsymbol{s}^U],\ \mathbf{p}\in[\mathbf{p}^L,\mathbf{p}^U] \end{aligned} \tag{P4}$$

The constraint functions are equivalently stated, for all scenarios, as

$$\mathcal{C}(\boldsymbol{s},\mathbf{p}) \equiv \{\mathbf{C}^{(1)}(\boldsymbol{w}^{(1)},\mathbf{p},\boldsymbol{\theta}^{(1)})^\top,\ldots,\mathbf{C}^{(n_s)}(\boldsymbol{w}^{(n_s)},\mathbf{p},\boldsymbol{\theta}^{(n_s)})^\top\}^\top \tag{14}$$

where depending on the constraint type (equality or inequality), the vectors $\mathcal{C}^L$ and $\mathcal{C}^U$ are appropriately defined. For each scenario i, the concatenated shooting node parameters are defined as $\boldsymbol{w}^{(i)} := (\boldsymbol{w}_0^{(i)\top},\ldots,\boldsymbol{w}_n^{(i)\top})^\top$, where for each shooting interval we define $\boldsymbol{w}_j^{(i)} := (\boldsymbol{\xi}_{0,j}^{(i)\top},\boldsymbol{\mu}_{0,j}^{(i)\top},\upsilon_j^{(i)\top})^\top$. Again, the NLP as posed produces a highly sparse diagonal structure within the constraint Jacobian matrix, where for each scenario block we have an additional sparse matrix for each shooting interval. The Jacobian matrices can be stated accordingly as

$$\begin{aligned} &\nabla_{\boldsymbol{s}}\mathcal{C}(\boldsymbol{s},\mathbf{p}) \\ &:= \operatorname{diag}\left\{\nabla_{\boldsymbol{w}^{(1)}}\mathbf{C}^{(1)}(\boldsymbol{w}^{(1)},\mathbf{p},\boldsymbol{\theta}^{(1)}),\ldots,\nabla_{\boldsymbol{w}^{(n_s)}}\mathbf{C}^{(n_s)}(\boldsymbol{w}^{(n_s)},\mathbf{p},\boldsymbol{\theta}^{(n_s)})\right\} \end{aligned} \tag{15}$$

$$\begin{aligned} &\nabla_{\mathbf{p}}\mathcal{C}(\boldsymbol{s},\mathbf{p}) \\ &:= \left(\nabla_{\mathbf{p}}\mathbf{C}^{(1)}(\boldsymbol{w}^{(1)},\mathbf{p},\boldsymbol{\theta}^{(1)})^\top,\ldots,\nabla_{\mathbf{p}}\mathbf{C}^{(n_s)}(\boldsymbol{w}^{(n_s)},\mathbf{p},\boldsymbol{\theta}^{(n_s)})^\top\right)^\top \end{aligned} \tag{16}$$

Once constructed, the objective and constraint functions and corresponding objective gradient vector and block sparse constraint Jacobian matrices can be passed to an appropriately selected NLP solver. Typically, the multiple-shooting technique utilizes SQP-based algorithms, as these algorithms are often designed with efficient Lagrangian Hessian approximation schemes, thus allowing the user to provide only first-order derivatives. IP algorithms, on the other hand, are often best utilized with the explicit definition of second-order derivatives. Thus, to effectively apply an IP NLP solver to a multiple-shooting formulation, one would need to further generate second-order sensitivity information at an added computational expense.

Algorithm parallelization and implementation

The multiple-shooting algorithm benefits from a naturally parallel structure that does not require any additional decomposition techniques. This natural decoupling of each shooting interval is induced by the introduction of the optimization parameters $\boldsymbol{\xi}_{0,j},\boldsymbol{\mu}_{0,j}$ for $j=0,\ldots,n-1$, and allows the DAE in each shooting interval to be independently solved in parallel. Parallel multiple-shooting implementations have been explored to varying degrees by Kiehl[36] for ordinary differential equation (ODE) models, Leineweber et al.[7] for DAE models, Jeon[37] for DAE models with parallel adjoint sensitivity computation, and Bachmann et al.[38] within a Modelica modeling environment.

The main difference in our proposed optimization formulation and corresponding implementation is that we have incorporated an additional layer of parallelization in terms of the individual scenarios used within the multiperiod approach. This concept can be visualized in Figure 1 (illustrated using only the differential states), where for a single scenario realization i each integration task ($I_{i,j}$) for each shooting interval j over the entire time horizon is stacked into a vector and then this is repeated for all scenarios n_s. The result is a large $n\cdot n_s$ dimensional vector of independent integration tasks which can be broken up and solved in parallel using several processors (P_i for $i=0,\ldots,N$). However, for numerous integration tasks per DAE evaluation (e.g., 100–1000), it is more likely that the tasks are divided into several evenly distributed blocks (in terms of number, not computation work load) and then off-loaded to a parallel computing server for evaluation. To reap the full potential of the parallel computation, the processors must be loaded with a sufficient amount of computation work. Note that our implementation goes beyond the standard parallel multiple-shooting approach[36] which is confined to parallelizing only the shooting intervals.

A prototype implementation of the multiperiod multiple-shooting dynamic optimization algorithm was created in the Matlab scripting language where the NLP and DAE solutions were performed using third-party solvers. The parallelization of the DAE solution was implemented using the Matlab Parallel Computing Toolbox. Additionally, the codes were run on a server within the Shared Hierarchical Academic Research Computing Network (SHARCNET), which is a high-performance parallel computing network comprising

several universities within southern Ontario, Canada. Submission of the codes to the server was performed using the Matlab Distributed Computing Server (DCS) whereby the codes were sent to a scheduler and appropriately executed with a specified level of resource allocation (i.e., amount of RAM and number of processors). The NLP solution utilized the sparse SQP solver SNOPT (version 7.2-11),[39] while for the integration of the dynamic model we utilized the tools available from the SUNDIALS suite of solvers (version 2.5.0).[40] In particular, for explicit ODE models, our implementation uses the provided Matlab interface to CVODES, while for implicit DAE models we use IDAS. The particular focus of parallelization within our implementation is on the DAE and sensitivity solution as opposed to the NLP solution; the former is known to consume up to 90% of the total program computation time on serial machines.[41] Accordingly, the NLP solution is performed in serial (on a single processor), while for each function evaluation (objective and constraints), the embedded dynamic model and corresponding sensitivity solution is solved in parallel. Timing measurements were taken based on the wall clock time of: (1) the full program using Matlab's tic/toc technique; (2) the in-solver NLP solution; and (3) the parallel ODE/DAE and sensitivity solution (which includes the serial vector formation of the objective/constraints and sparse derivative matrices).

Illustrative Design and Control Case Studies

In this section, we apply the parallel multiperiod multiple-shooting approach to two different benchmark design and control problems of different magnitude and complexity. The purpose here is to assess potential performance improvements through a parallel computing implementation. Our first example uses a modified version of the Newell and Lee[42] evaporator benchmark model described by Kookos and Perkins,[43] where we consider determining an economically optimal design and controller tuning parameters subject to uncertain input flow rate and composition disturbances. The second example utilizes a larger model of a continuous binary distillation column discussed by Schweiger and Floudas.[26] Each example comprises a set of nonlinear DAEs describing the process model, path constraints on the state and closed-loop manipulated variables, and a nonlinear economic objective function that includes design and operating costs. Closed-loop control is achieved via multiloop PI controllers based on a fixed control structure.

For the particular multiperiod design and control formulations which embed continuous closed-loop PI controllers considered in this article, we define the first-stage variables as equipment dimensions and controller tuning parameters (i.e., proportional gain K_c, reset time τ_I, and output set-points y_{set}). The second-stage decision variables include the closed-loop manipulated variables $\mathbf{u}^{(i)}(t)$ over each scenario i, which are implicitly determined by the PI controller equations defined as

$$u^{(i)}(t)=\bar{u}^{(i)}+K_c(y^{(i)}(t)-y_{\text{set}})+\frac{K_c}{\tau_I}I^{(i)}(t) \tag{17}$$

$$I^{(i)}(t)=\int_{t_0}^{t}(y^{(i)}(\tau)-y_{\text{set}})\,d\tau \tag{18}$$

where the controller bias terms $\bar{u}^{(i)}$ are the explicit second-stage optimization decision variables for each scenario $i=1,\ldots,n_s$. These control actions allow for recourse once the uncertainty is resolved, and in this article we consider uncertainty solely within the disturbance inputs, $\mathbf{v}(t)$, which are defined as a function of each uncertain parameter $\boldsymbol{\theta}$ over all scenario realizations. The disturbance inputs are applied as step inputs to the system which require two uncertain parameters per disturbance trajectory representing the initial value before the step is applied (defined previously in vector form for all disturbance variables as $\mathbf{v}_0$) and a second parameter corresponding to the step magnitude (defined previously as $\Delta\mathbf{v}$). These uncertain parameters are generated from a uniform distribution by sampling between a specified upper and lower bound for $\mathbf{v}_0$ and $\Delta\mathbf{v}$. Throughout the article we adopt the terminology of a complete disturbance step realization (or vector of different disturbances) representing a single scenario realization. For example, 10 scenario realizations ($n_s=10$) of a set of two independent disturbance variables ($n_v=2$) would correspond to the generation of $2\cdot n_v\cdot n_s=40$ random variables.

Example 1: Evaporator Process Design

The evaporation process considered for this first example is depicted in Figure 2, where it is desired to remove a volatile liquid from a solution to yield a high concentration of a nonvolatile solute. The process is composed of a pressurized evaporation vessel, a separator or settling tank, a recirculation pump, and an overhead condenser (heat exchanger). For convenience, the chosen model equations used in this study are provided in Appendix A.

The optimization design parameters $\mathbf{p}$ comprise the evaporator and heat exchanger areas (combined with the heat-transfer coefficient, U; UA_1, UA_2), PI controller tuning parameters (K_{ci}, τ_{Ii}, $i=1,2$) and output variable set-points ($\bar{x}_2, \bar{P}_2$). The differential state variables $\mathbf{x}(t)$ are the evaporator product composition (x_2) and operating pressure (P_2), which are also considered as output variables to be controlled using the manipulated variables, $\mathbf{u}(t)$, of inlet steam pressure (P_{100}) and cooling water flow rate (F_{200}). Additionally, we consider two uncertain external input disturbances, $\mathbf{v}(t)$, of feed flow rate (F_1) and composition (x_1). A detailed listing of the variables shown in Figure 2 is provided in Table 1, while a further listing of variable and parameter specifications is provided in Appendix A.

The chosen design and control formulation can be described as a continuous multiperiod dynamic optimization problem according to the following equations

$$\min_{\mathbf{p}}\ \mathcal{J}(\mathbf{p}):=C_{\text{cap}}(\mathbf{p})+\sum\nolimits_{i\in\mathcal{S}}\mathsf{w}_i\cdot C_{\text{op}}(\mathbf{x}^{(i)}(t_f),\,\mathbf{u}^{(i)}(t_f),\,\mathbf{v}^{(i)}(t_f),\,\mathbf{p},\,\boldsymbol{\theta}^{(i)},\,t_f) \tag{19}$$

$$\text{st}:\ \text{ODE model (5 eqns.)} \tag{20}$$

$$\dot{\mathbf{x}}^{(i)}(t_0)=\mathbf{0} \tag{21}$$

$$P_{100}^{(i)}(t)=\bar{P}_{100}^{(i)}+K_{c_1}(x_2^{(i)}(t)-\bar{x}_2)+\frac{K_{c_1}}{\tau_{I_1}}I_1^{(i)}(t) \tag{22}$$

$$F_{200}^{(i)}(t)=\bar{F}_{200}^{(i)}+K_{c_2}(P_2^{(i)}(t)-\bar{P}_2)+\frac{K_{c_2}}{\tau_{I_2}}I_2^{(i)}(t) \tag{23}$$

$$F_1^{(i)}(t)=\bar{F}_1^{(i)}+\Delta\bar{F}_1^{(i)}\,\eta(t,t_{\text{step}}) \tag{24}$$

$$x_1^{(i)}(t)=\bar{x}_1^{(i)}+\Delta\bar{x}_1^{(i)}\,\eta(t,t_{\text{step}}) \tag{25}$$

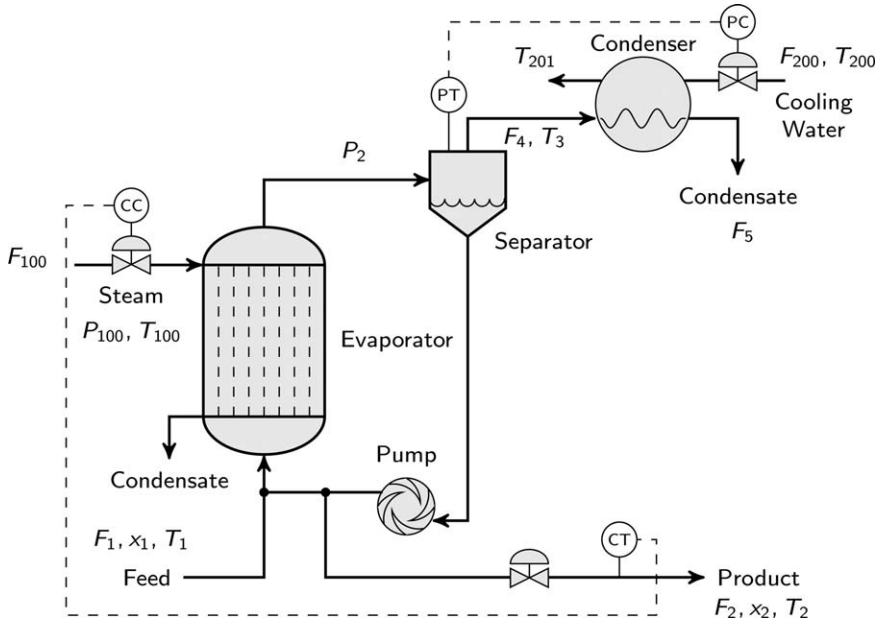

Figure 2. Case study 1: evaporator process schematic and control structure.

$$\varepsilon_{\text{ise}}^{(i)}(t_f) \leq \epsilon \tag{26}$$

$$x_2^L - x_2^{(i)}(t) \leq 0 \tag{27}$$

$$x_2^{(i)}(t) - x_2^U \leq 0 \tag{28}$$

$$P_2^L - P_2^{(i)}(t) \leq 0 \tag{29}$$

$$P_2^{(i)}(t) - P_2^U \leq 0 \tag{30}$$

$$P_{100}^{(i)}(t) - P_{100}^U \leq 0 \tag{31}$$

$$F_{200}^{(i)}(t) - F_{200}^U \leq 0 \qquad \forall t \in [t_0, t_f], i \in \mathcal{S} \tag{32}$$

We note that the formulation is continuous in the optimization parameters as we consider a fixed plant topology and control structure. The multiperiod algorithm that we consider comprises a single optimization stage (i.e., a single NLP formulation), in which a number of scenarios is introduced to approximate the uncertainty in the disturbance inputs and hence the stochastic nature of the problem. The investigation to be performed considers the effect on the computational performance of parallelization over uncertainty scenarios and multiple-shooting intervals.

Within the above formulation, the objective is to determine an economically optimal and control feasible design, where it is desired to keep the process at steady state prior to the input disturbance, and subsequently to drive the process back to the original steady state using feedback control (i.e., disturbance rejection). The formulation follows the structure of a two-stage stochastic program, where the first-stage costs correspond to the design, given here as the capital cost of the plant as a function of the design parameters ($C_{\text{cap}}(\mathbf{p})$), and the second-stage costs are dependent on the operating costs of the plant for a given design ($C_{\text{op}}(\cdot)$). The stochastic aspect of the problem is captured using a weighted deterministic formulation where the associated weights are specified equally for each scenario as $\mathsf{w}_i = 1/n_s$. The equality constraints within this formulation are the dynamic process model equations; the initial state variable conditions at time t_0 (i.e., steady-state constraints); the closed-loop PI controller equations; and the single-step disturbance approximation equations. The inequality constraints comprise an end-point constraint on the integrated squared error control performance metric (ε_{ise}) and path constraints on the state (x_2, P_2) and manipulated variables (P_{100}, F_{200}). The experiments performed considered running the optimization algorithm using a high and low number of scenarios for the uncertain disturbance variables $\boldsymbol{\theta} := (\mathbf{v}_0^\top, \Delta\mathbf{v}^\top)^\top$, where $\mathbf{v}_0 := (\bar{F}_1, \bar{x}_1)^\top$ and $\Delta\mathbf{v} := (\Delta\bar{F}_1, \Delta\bar{x}_1)^\top$. The idea is to emulate a high and low work load on the processors to examine its effect on the computation speedup and efficiency.

We make several additional remarks to elucidate the optimization formulation and solution procedure:

- The process model can be generally stated as a system of DAEs; however, for the examples shown in this article, the algebraic Eqs. 22–25 were eliminated via substitution of explicit algebraic variable expressions into the differential equations thus only requiring the solution of an explicit ODE system. The multiple-shooting discretization derivations given in previous sections are quite general and applicable to solving DAEs directly.
- The steady-state constraints were imposed by setting the explicit right-hand side of the ODEs to zero at t_0 (i.e., $\dot{\mathbf{x}}(t_0) = \mathbf{0} \mapsto \mathbf{f}_d(\overline{\mathbf{x}}, \overline{\mathbf{u}}, \overline{\mathbf{v}}, \mathbf{p}, t_0) = \mathbf{0}$).

Table 1. Case Study 1: Evaporator Model Variable Definitions

F_1: feed flow rate (kg/min)	T_1: feed temperature (°C)
F_2: product flow rate (kg/min)	T_2: product temperature (°C)
F_4: vapor flow rate (kg/min)	T_3: vapor temperature (°C)
F_5: condensate flow rate (kg/min)	T_{100}: steam temperature (°C)
F_{100}: steam flow rate (kg/min)	T_{200}: cooling water inlet temperature (°C)
F_{200}: cooling water flow rate (kg/min)	T_{201}: cooling water outlet temperature (°C)
x_1: feed composition (%)	x_2: product composition (%)
P_2: operating pressure (kPa)	P_{100}: steam pressure (kPa)

Table 2. Case Study 1: Evaporator Optimal Design and Control Parameters

	Initial Guess	Bounds	Optimal Solution		
Parameter ($\mathbf{p}$)[a]	$\mathbf{p}_g$	$\in [L, U]$	$n_s = 1$	$n_s = 5$	$n_s = 50$
$\mathcal{J}$ (\$/yr)	–	–	71,690.52	71,556.57	71,769.11
UA_1 [kW/K]	9.6	$\in$ [2,15]	4.652	4.795	4.932
UA_2 [kW/K]	6.8	$\in$ [2,15]	7.430	7.584	7.734
K_{c_1}	-20	$\in$ [-50,50]	-50	-50	-50
K_{c_2}	20	$\in$ [-50,500]	216.75	500	500
τ_{I_1}	2	$\in$ [1,5]	1.07	1	1
τ_{I_2}	2	$\in$ [1,5]	5.00	1	1
$\bar{x}_2$ [%]	29	$\in$ [25,30]	25.06	25.23	25.25
$\bar{P}_2$ [kPa]	49	$\in$ [40,80]	40.00	40.04	43.07

[a]Note, the controller bias parameters ($\bar{P}_{100}$, $\bar{F}_{200}$) are scenario-dependent optimization parameters, not shown. Initial guesses defined as $\bar{P}_{100} = 193.45$ kPa, $\bar{F}_{200} = 207.32\,\mathrm{kg/min}$.

- Bound constraints were introduced on the manipulated variables by adding appropriate inequality constraints at each shooting node.
- An end-point constraint on the integrated squared error control performance metric was introduced to influence the control performance. The metric is defined as

$$\varepsilon_{\mathrm{ise}}(t_f) := \int_{t_0}^{t_f} \{(x_2(t) - \bar{x}_2)^2 + (P_2(t) - \bar{P}_2)^2\} dt \qquad (33)$$

Using such an approach allows one to directly set a tolerance on the control performance via an inequality end-point constraint ($\varepsilon_{\mathrm{ise}}(t_f) \leq \epsilon$).

- The variables and constraints of this formulation can be broken down as follows: multiple-shooting equality constraints for n intervals (including initial constraints at t_0) of dimension $n_c := n_x n_s (n+1)$ (where n_x is the number of continuity constraints at the shooting nodes and n_s is the number of scenarios), similarly we have the same number of shooing node variables ($\boldsymbol{\xi}_0^{(i)}$); closed-loop control signals at t_0 ($P_{100}^{(i)}(t_0)$, $F_{200}^{(i)}(t_0)$) must be set to their controller bias values ($\bar{P}_{100}^{(i)}$, $\bar{F}_{200}^{(i)}$) via equality constraints of dimension $n_{h_1} = n_u n_s$, again we have the same number of decision variables as equations; finally, we have equality constraints at t_0 stating that the initial output variables be equal to their set-points of dimension $n_{h_2} = n_y n_s$. For this last set of equality constraints, the outputs are simply the states, which are already specified via an initial steady-state equality constraint; thus, these output constraints may be removed. The remaining constraints are inequalities for path or end-point constraints, and the main degrees of freedom for the formulation are the design variables UA_1, UA_2, K_{c_j}, τ_{I_j}, $j = 1, 2$, $\bar{x}_2$, and $\bar{P}_2$.
- To remove any obscurity in the solution timing results incurred by possible automatic derivative generation techniques, the first-order derivatives of the NLP objective/constraints with respect the optimization parameters were specified analytically and provided directly to the optimization algorithm.

Before considering the potential computation speedup via our parallel solution implementation, we first assess the optimization solution. Table 2 lists the design parameters, their initial guesses and lower and upper bounds as posed to the optimization solver. Additionally, stated are the final objective and parameter values for 1, 5, and 50 scenario realizations of the uncertain disturbance variables. In Figure 3, we illustrate the optimal solution trajectories for 5 scenarios, which corresponds to five sets of variable trajectories. For both the closed-loop inputs $\mathbf{u}(t)$ and controlled outputs $\mathbf{y}(t)$, we see that the discretized inequality path constraints are not violated, and for variables P_{100}, x_2, and P_2 the constraint becomes active for a short period of time when the output variables overshoot their set-points (see dashed lines in Figure 3). Furthermore, in Table 2, we see that for the desired control performance tolerance ϵ (specified *a priori*) and the chosen parameter bounds, the PI controller parameters K_c and τ_I converged to either an upper or lower bound at the optimal objective value solution. Relaxing the controller parameter bounds did not significantly change the results as the parameter values remained active at the bounds upon convergence to an optimal solution. This behavior is likely a consequence of the constraint on the integrated squared error where we set a relatively low tolerance value (e.g., $\epsilon = 0.1$) which would induce a rather aggressive control action (i.e., high gain magnitudes).

The parameters obtained using a single scenario realization represent nominal disturbance conditions and are presented for comparison to the more robust solutions which utilize multiple scenarios. There are a few notable aspects that can be observed from the optimal design solution. First, as the number of scenarios n_s is increased, the optimal surface area of both the evaporator (UA_1) and condenser (UA_2) also increases. This behavior reflects an increasing conservativeness of the design, with increasing scenarios, which implicitly supplement the necessary control actions to reject the uncertain disturbance inputs. In other words, in order for the control system to adequately handle all possible disturbance scenarios, the overall equipment surface areas must be increased at an economic penalty to dampen the disturbance effects. Another related observation is that to ensure the path constraints on the controlled variables are not violated, the output set-points (i.e., design solution) must back off from their bounds to adequately account for the anticipated control action used to reject the uncertain disturbance realizations.

For this case study, the algorithm was set up using $n = 30$ shooting intervals over a time horizon of 30 min and a low and high number of scenarios n_s defined according to Table 3. This corresponds to $n \cdot n_s$ independent integration tasks which are solved in parallel. Thus, we are parallelizing over both the shooting intervals and scenario realizations, by viewing each independent integration task as parallizable and appropriately grouping these tasks into blocks such that a relatively even amount of tasks are distributed among the available processors. The optimization problem is nonlinear and nonconvex, and is solved in serial using the SNOPT

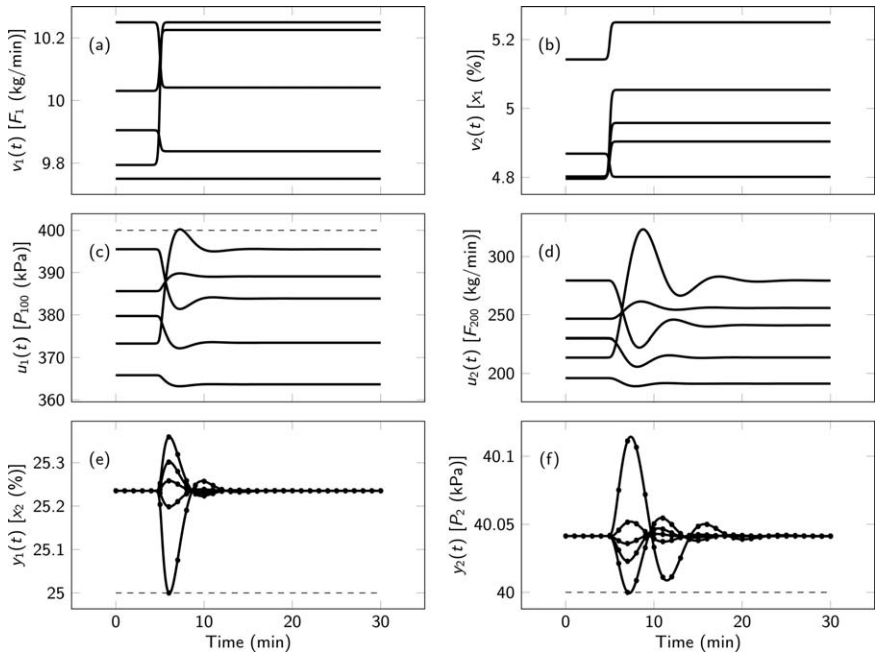

Figure 3. Case study 1: evaporator dynamic optimization trajectories for $n_s = 5$: (a)–(b) uncertain disturbance inputs v(*t*); (c)–(d) closed-loop inputs u(*t*); and (e)–(f) controlled outputs y(*t*).

solver. To solve the embedded dynamic model equations and associated sensitivity equations, we used the CVODES solver, where the model equations are posed as an explicit ODE. The design problem being considered has eight main degrees of freedom for the design and control variables (UA_1, UA_2, $\bar{x}_2$, $\bar{P}_2$, K_{c_j}, τ_{I_j}, $j=1,2$). Additionally, the PI controller bias parameters are introduced as degrees of freedom for each specific scenario. As we required the system to be at steady state initially, and each initial disturbance realization ($\bar{F}_1^{(i)}$, $\bar{x}_1^{(i)}$) is defined *a priori*, then the controller bias must be introduced for each scenario to appropriately compensate for the initial disturbance values.

The potential computation speedup attainable using the parallelized multiperiod multiple-shooting implementation will now be investigated. As previously discussed, the portion of the algorithm that we are parallelizing is the embedded DAE solution, as opposed to the NLP algorithm. To examine the potential speedup and efficiency of using more processors to evaluate the embedded dynamic model, we have selected the number of processors according to $n_p=2^k$ where $k = 0,2,3,4,5$. The computation speedup and processor efficiency are computed according to the following equations

$$\text{Speedup } (S) = \frac{\text{constraint/objective function eval. time w/} n_p = 1}{\text{constraint/objective function eval. time w/} n_p = 2^k} = \frac{T_{\text{serial}}}{T_{\text{parallel}}} \tag{34}$$

$$\text{Efficiency } (E) = \text{Speedup}/n_p = \frac{T_{\text{serial}}}{n_p\, T_{\text{parallel}}} \tag{35}$$

where we use the NLP constraint and objective function evaluation time (which includes the parallel DAE and sensitivity solution and a much smaller serial constraint/objective construction component) as a basis of performance measurement, as opposed to the total program wall clock time which includes the serial in-solver NLP solution time. In these definitions, T_{serial} represents the constraint/objective evaluation time using a single processor and T_{parallel} corresponds to the evaluation time using multiple processors. The computation speedup is one measure of potential performance improvement, and under ideal conditions where the program can be fully parallelized and the processors are sufficiently utilized, the addition of processors should reveal a parallel solution time corresponding to $T_{\text{parallel}} := T_{\text{serial}}/n_p$ and thus produce a linear speedup of $S = n_p$ (see 45 degree solid line in Figure 4a) and an efficiency of 100%. However, in practice these conditions are not usually observed, as (1) there is an inherent aspect of the program that is serial, and (2) the increase of processors invariably introduces greater communication overhead which impedes both the speedup and efficiency. For example, for a given amount of work governed by the number of integration tasks, the true parallel solution time approximates to $T_{\text{parallel}} := T_{\text{serial}}/n_p + T_{\text{overhead}}$, where T_{overhead} is the total cumulative amount of time spent communicating data between processors (i.e., parallel overhead). This

Table 3. Case Study 1: Evaporator Optimization Timings for Parallel Multiperiod Algorithm

NLP Size and Solution Statistics					Wall Clock Time (s) for n_p Processors[a]				
n_s	#vars[b]	n_h	n_g	#iter[c]	$n_p = 1$	$n_p = 4$	$n_p = 8$	$n_p = 16$	$n_p = 32$
5	793	795	315	37	318.3	83.5	41.1	22.8	17.7
50	7858	7950	3150	36	3158.2	799.2	387.6	189.3	110.8

[a]Average wall clock times over three experiments.
[b]Discretized formulation with $n = 30$ shooting intervals, n_h equality and n_g inequality constraints, respectively.
[c]No. of major iterations using SNOPT SQP solver with optimality, feasibility tolerance of 1×10^{-4} and 1×10^{-6}, respectively.

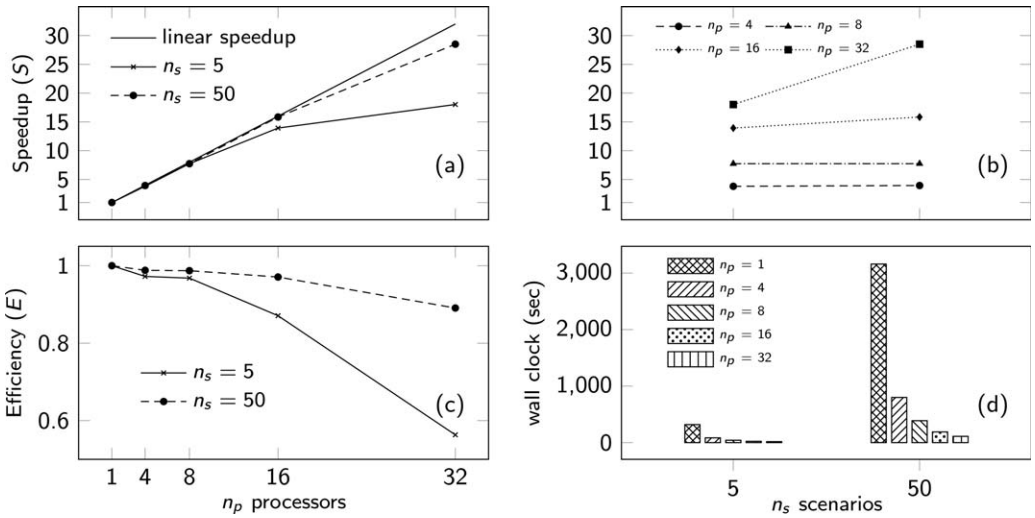

Figure 4. Case study 1: parallel multiperiod multiple-shooting algorithm speedup and efficiency with increasing number of processors and scenarios.

results in an observed speedup that is lower than the number of processors, with the relative drop off in speedup increasing as the number of processors increases. However, if we increase the amount of computation work, T_{overhead} will grow more slowly relative to T_{serial}/n_p, and allow for increased speedup and efficiency, and ultimately better utilization of the available resources. For a more detailed and in depth discussion on computation performance metrics and problem-size vs. machine-size dependencies, we refer the readers to Pacheco.[44]

In our particular implementation, the timing captures the dynamic model solution whereby the combined initial condition vector $\boldsymbol{\xi}_0$ is communicated in a balanced manner to the available processors, the computation is performed in parallel, and then the state variable solution at the end of each shooting node, $\mathbf{x}^{(i)}(t_{j+1})$ for all $j=0,\ldots,n-1$ and $i=1,\ldots,n_s$, is communicated back to the master processor. For example, if we have $n_p=4$ processors and $n \cdot n_s$ integration tasks, then the work load for each processor is $n \cdot n_s/4$ where any remaining tasks are distributed among the four processors. The time to communicate the data incurs overhead (denoted as T_{overhead}), which accumulates for each NLP function evaluation; thus, for the entire NLP solution there will have been performed two communication calls between master and slave processors to solve the entire embedded dynamic model for each objective/constraint evaluation.

Table 3 lists the optimization setup in terms of the number of scenarios used (n_s); the total number of discretized optimization variables (#vars); the total number of equality (n_h) and inequality (n_g) constraints, respectively. Additionally, we list the total number of major SQP iterations (as defined in Gill et al.[39]) and the average total program wall clock times (combined NLP and DAE solution timings) using an incremental number of computing processors. The optimization algorithm was run using a local parallel computing server within the SHARCNET cluster which uses two 12 core 2.2 GHz AMD Opteron chips and 32 GB of memory per computing node, and solution timings are reported based on an average of three experiments. Given a single node of 24 processors, we required the use of two computing nodes for the 32 processor trials. Note that the wall clock timings should be interpreted as the potential computation speedup from a relative perspective between each of the number of processors used and not as a reflection of the NLP or ODE solver solution speed. We make this statement since the results come from a prototyped implementation within the Matlab scripting language, which by its nature contains significant overhead not present in compiled programming languages. Nevertheless, we observe that when using a single processor with $n_s=5$ scenarios, the optimization algorithm requires approximately an average of 8.7 s per major SQP iteration and when increased to 16 processors the time per iteration decreased to about 0.6 s which indicates an average speedup of 15. When the number of scenarios is increased 10-fold we observe an improved relative speedup of approximately 16, indicating near linear speedup and close to perfect scalability of the embedded model solution.

Figure 4 provides plots of the speedup and efficiency and depicts the behavior when the number of scenarios is increased. Note that this figure is not directly based on the wall clock times reported in Table 3, and instead uses the cumulative average wall clock time of constraint/objective evaluation, as defined in Eqs. 34 and 35, excluding the serial in-solver NLP evaluation time. In Figures 4a and c, when we increase the number of processors n_p from 16 to 32 at a fixed amount of work ($n_s=5$), which doubles the overall communication overhead, we see a significant decrease in speedup and efficiency. This behavior occurs because the communication time is relatively large, for the given amount of computation work, compared to the serial solution time. When we increase the problem size from $n_s=5$ to $n_s=50$, we see significantly improved speedup and efficiency from $n_p=16$ to $n_p=32$. This is due to a relatively small increase in communication overhead T_{overhead} compared to the serial solution time $T_{\text{serial.}}$ Figure 4b further illustrates the behavior of speedup when we increase the computation work at a fixed number of processors. Here we see improved speedup using $n_p=16$ and $n_p=32$ when increasing n_s from 5 to 50, with an unnoticeable effect using lesser processors. Therefore, the effect of communication overhead can be reduced by introducing more computation work (i.e., greater loading of the processors). Ultimately, when utilizing 32 processors on a problem with 50 scenarios, we achieve a speedup in the dynamic model solution of approximately 29 times at 89% processor efficiency.

Example 2: Binary Distillation Process Design

The second example considered is the design and control of a continuous binary distillation column. Our purpose here is to provide further demonstration of our multiperiod multiple-shooting algorithm using a larger model and investigate the algorithm performance in terms of speedup and efficiency, and additionally to look at the timing breakdown between the embedded dynamic model and nonlinear program solution. The design and control problem follows in a similar manner to the previous case study, where we fix the control structure and solve an economic optimization formulation for the distillation column diameter and PI controller tuning parameters. For illustrative purposes, we consider a fixed/specified column height, tray dimensions, feed tray location, and reboiler and condenser surface areas. The process is illustrated in Figure 5, where F and z correspond to uncertain disturbance inputs for feed flow rate and composition, respectively; R and V represent the manipulated reflux and boil-up flow rates; and x_0 and x_{n_t+1} are controlled output compositions within the bottoms and distillate product.

The design and control optimization parameters of interest include the column diameter D_{col}, the PI controller tuning parameters K_{c_i}, τ_{I_i} for $i = 1,2$ and output set-points $\bar{x}_0$, $\bar{x}_{n_t+1}$. The dynamic model of the distillation column contains roughly 40 differential equations and a similar number of algebraic equations (see Appendix B for a detailed listing). The model is solved as an ODE by eliminating the algebraic equations via the explicit substitution of the algebraic state variables into the differential equations. The resulting design and control formulation is posed in a general manner as follows

$$\min_{\mathbf{p}} \quad \mathcal{J}(\mathbf{p}) := C_{\text{cap}}(\mathbf{p}) + \sum_{i\in\mathcal{S}} \mathsf{w}_i \cdot C_{\text{op}}(\mathbf{x}^{(i)}(t_f),\, \mathbf{u}^{(i)}(t_f),\, \mathbf{v}^{(i)}(t_f),\, \mathbf{p},\, \boldsymbol{\theta}^{(i)},\, t_f) \tag{36}$$

$$\text{st}: \quad \text{ODE model (40 eqns.)}$$

$$\dot{\mathbf{x}}^{(i)}(t_0) = \mathbf{0} \tag{37}$$

$$V^{(i)}(t) = \bar{V}^{(i)} + K_{c_1}(x_0^{(i)}(t) - \bar{x}_0) + \frac{K_{c_1}}{\tau_{I_1}} I_1^{(i)}(t) \tag{38}$$

$$R^{(i)}(t) = \bar{R}^{(i)} + K_{c_2}(x_{n_t+1}^{(i)}(t) - \bar{x}_{n_t+1}) + \frac{K_{c_2}}{\tau_{I_2}} I_2^{(i)}(t) \tag{39}$$

$$F^{(i)}(t) = \bar{F}^{(i)} + \Delta\bar{F}^{(i)}\, \eta(t, t_{\text{step}}) \tag{40}$$

$$z^{(i)}(t) = \bar{z}^{(i)} + \Delta\bar{z}^{(i)}\, \eta(t, t_{\text{step}}) \tag{41}$$

$$\varepsilon_{\text{ise}}(t_f)^{(i)} \le \epsilon \tag{42}$$

$$x_{n_t+1}^{L} - x_{n_t+1}^{(i)}(t_f) \le 0 \tag{43}$$

$$x_0^{(i)}(t_f) - x_0^{U} \le 0 \tag{44}$$

$$D_{\text{col}}^{\min}(\overline{V}^{(i)}) - D_{\text{col}} \le 0 \qquad \forall t \in [t_0, t_f], i \in \mathcal{S} \tag{45}$$

where we use an economic objective function, similar to the previous case study, consisting of the capital cost of the column tray/shell construction and the operating costs associated with the cooling water and steam flow rates used in the condenser and reboiler, respectively. Additionally, we pose equality path constraints for V and R; disturbance input equality constraints with uncertain parameters $\boldsymbol{\theta} := (\mathbf{v}_0^\top, \Delta\mathbf{v}^\top)^\top$, where $\mathbf{v}_0 := (\bar{F}, \bar{z})^\top$ and $\Delta\mathbf{v} := (\Delta\bar{F}, \Delta\bar{z})^\top$;

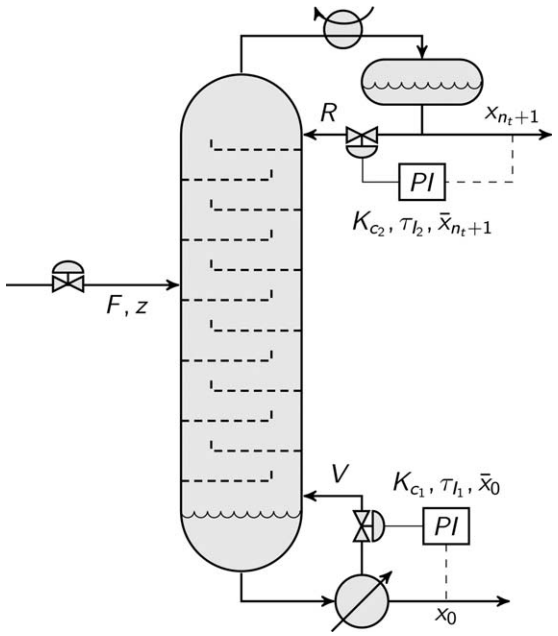

Figure 5. Case study 2: distillation process schematic and control structure.

a control performance inequality end-point constraint to explicitly influence the disturbance rejection; end-point constraints on the controlled output variables; and a column flooding constraint (Eq. 45) which ensures that the column maintains a minimum diameter based on the vapor boil-up rate. A detailed formulation of the objective function is provided in Appendix B.

The optimization design parameter initial guesses and bounds are listed in Table 4. Additionally stated are the optimal objective and parameter values for nominal disturbance conditions and uncertain disturbance realizations constructed from 10 and 20 scenarios. The objective function solution indicates the expected result that the economics worsen as uncertainty is introduced into the formulation. In other words, to account for possible uncertain disturbance realizations, the distillation column diameter must be increased from its nominal value at an economic expense. Also notable, from the listed parameter solution, are the nonunique values of the controller tuning parameters between scenario levels of 10 and 20. These nonunique values reflect the multiple possible tuning settings to adequately control the nonlinear system, which is a known behavior when including both the controller gain and integral reset time as degrees of freedom in the optimization formulation. Figure 6 provides the imposed disturbances realizations $\mathbf{v}(t)$ (with uncertain initial and final step values), the optimal manipulated variables $\mathbf{u}(t)$ and optimal output trajectories $\mathbf{y}(t)$ for a problem size of 10 scenario realizations per disturbance variable. Additionally, the multiple-shooting discretization consisted of $n = 25$ shooting nodes over a 50 minute time horizon, which amounts to total of $n \cdot n_s$ integration tasks $I_{i,j}$ distributed over the available processors.

With an optimal solution established, we now turn to assessing the potential performance improvement via parallelization of this much larger distillation model, relative to the previous evaporator model. In a similar manner to the previous case study, we provide in Table 5 the dimensions of the

Table 4. Case Study 2: Distillation Optimal Design and Control Parameters

	Initial Guess	Bounds	Optimal Solution		
Parameter ($\mathbf{p}$)[a]	$\mathbf{p}_g$	$\in [L, U]$	$n_s = 1$	$n_s = 10$	$n_s = 20$
$\mathcal{J}$ (\$/yr)	–	–	20,497.49	21,077.36	21,023.70
D_{col} [m]	0.75	$\in$ [0.1,2]	0.721	0.745	0.746
K_{c_1}	0.74	$\in$ [−50,50]	1.376	2.476	6.747
K_{c_2}	−4.0	$\in$ [−50,50]	−4.185	−5.113	−10.501
τ_{I_1}	3.5	$\in$ [1,9]	1.181	2.206	2.992
τ_{I_2}	8.0	$\in$ [1,9]	2.671	1.998	3.159
$\bar{x}_0$	0.01	$\in$ [0.005,0.05]	0.05	0.05	0.05
$\bar{x}_{n_t+1}$	0.97	$\in$ [0.85,1]	0.98	0.98	0.98

[a]Note, the controller bias parameters ($\bar{R}$, $\bar{V}$) are scenario-dependent optimization parameters, not shown. Initial guesses defined as $\bar{R}$=1.0024 kmol/min, $\bar{V}$= 1.5426 kmol/min.

discretized NLP formulation, the number of major SQP iterations required by SNOPT to meet the specified feasibility/optimality tolerances, and the total program wall clock times (combined NLP and DAE solution timings). The NLP formulation for $n_s = 20$ is about 2.5 times the size (in terms of variables/constraints) of the largest previous case study, and required, using a single processor, about 1.8 h of total computation time with an average solution time per SQP iteration of 5.4 min. Increasing the number of processors to 16, we observed a reduction in computation time to 0.3 h or an average of 0.9 min per iteration.

An assessment of the cumulative solution time for the dynamic model and sensitivity equations is provided in Figure 7. In a similar manner to the previous case study, this figure was constructed excluding the in-solver serial NLP solution time, which is present in the wall clock times reported in Table 5. The behavior of the speedup and efficiency profiles follows the same trends seen in the previous case study, where for the considered scenario levels of $n_s = 10$ and 20, we see an approximate linear speedup up to $n_p = 8$ followed by a slight decline to about $S \approx 13$ and $E > 80\%$ at $n_p = 16$ ending with a significant drop off at $n_p = 32$. In comparison to the largest example in the previous case study, where we used $n \cdot n_s = 1500$ integration tasks and saw near linear speedup, we are now using only 500 integration tasks (work per task is slightly greater due to a greater number equations). As a result, we see a slightly greater decline in speedup and efficiency at $n_p = 16$ and 32, due to a smaller fraction of parallel work load, which indicates that for greater efficiency we should either increase the work load or decrease the number of processors used. A related aspect when increasing the problem size (via an increase in scenario realizations) is the adverse effect created on the memory requirements within the serial NLP solver. The proper use and definition of sparse matrices within the constraint derivatives usually ensures efficient memory usage. However, one must be careful when using interpreted scripting languages (like Matlab), as memory can be a significant bottleneck if present in insufficient amounts relative to the necessary requirements for matrix and/or vector storage.

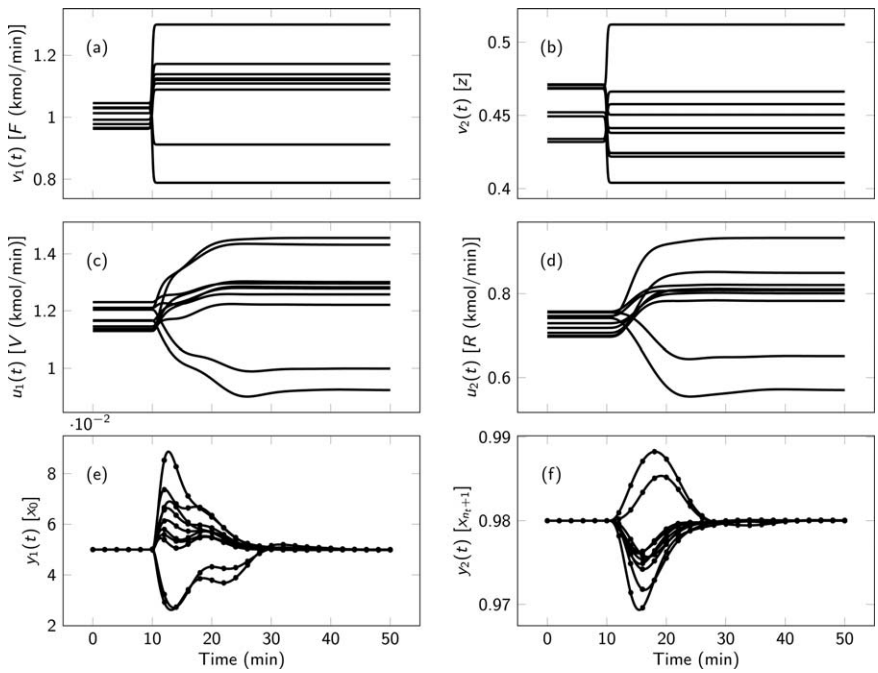

Figure 6. Case study 2: distillation dynamic optimization trajectories for $n_s = 10$: (a)–(b) uncertain disturbance inputs v(t); (c)–(d) closed-loop inputs u(t); and (e)–(f) controlled outputs y(t).

Table 5. Case Study 2: Distillation Optimization Timings for Parallel Multiperiod Algorithm

NLP Size and Solution Statistics					Wall Clock Time (s) for n_p Processors				
n_s	#vars[a]	n_h	n_g	#iter[b]	$n_p = 1$	$n_p = 4$	$n_p = 8$	$n_p = 16$	$n_p = 32$
10	10,167	10,190	40	29	4188.25	1760.24	867.35	584.86	508.31
20	20,327	20,380	80	20	6457.16	2385.95	1406.15	1067.57	880.29

[a]Discretized formulation with $n = 25$ shooting intervals, n_h equality and n_g inequality constraints, respectively.
[b]SNOPT SQP solver with optimality, feasibility tolerance of 1×10^{-4} and 1×10^{-5}, respectively.

An important computation aspect to highlight with this case study is the relative computation time between NLP and DAE and solution. Accordingly, we break down the timing measurements into the total ODE and sensitivity solution (for all shooting intervals and scenarios) which includes the complete NLP objective/constraint construction, and the remaining in-solver NLP solution time. Note, the ODE solution is the parallelized portion of the algorithm which should reflect a decreased time with increased processors. Figure 8 plots a breakdown in time between each of these solutions from which we observe that the serial NLP portion of the code remains relatively constant as processors are added, while the parallel DAE portion enjoys significant speedup. Increasing the number of processors further, beyond that shown, can certainly improve even more the DAE solution time; however, doing so without additionally increasing the problem size is at a cost of reduced efficiency. Introducing more processors is most beneficial if we increase the problem size and hence provide more computation work to the processors.

Conclusions

In this article, we have presented a novel parallel computing approach for large-scale dynamic optimization under uncertainty. A multiple-shooting approach was used for the DAE optimization, and the uncertain parameter space discretized to yield a multiperiod or multiscenario formulation. The DAE model and sensitivity equations corresponding to each shooting interval and scenario constitute independent integration tasks, well suited for parallel processing. The formulation was applied to two integrated design and control case studies, where the objective was to determine design and controller tuning parameters that minimize the combined annualized capital and operating cost of the plant subject to uncertain disturbance inputs and the dynamic process model. The solution of the dynamic model was parallelized by evenly distributing the independent integration tasks from each shooting interval and scenario realization over several processors. Through parallelization, the embedded dynamic model function evaluations were able to be solved with near

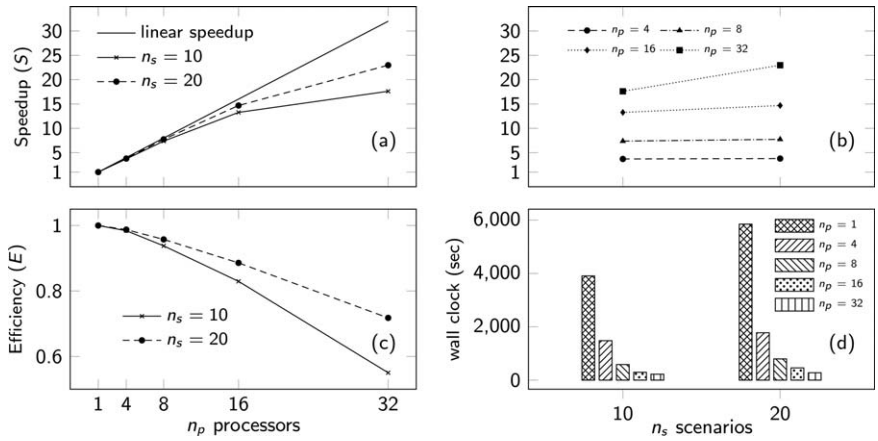

Figure 7. Case study 2: parallel multiperiod multiple-shooting algorithm speedup and efficiency with increasing number of processors and scenarios.

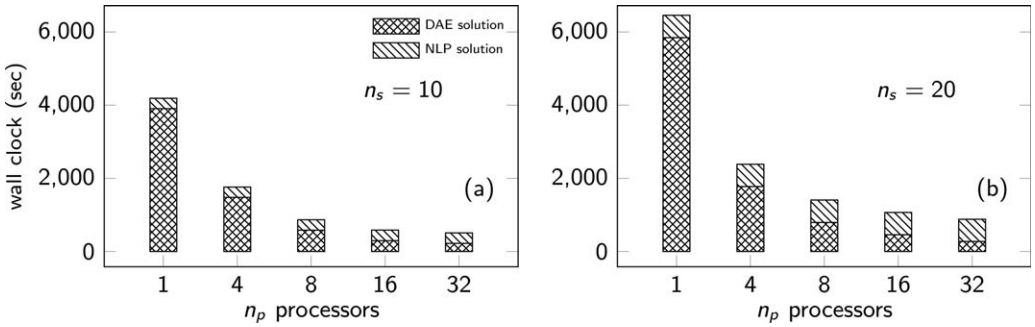

Figure 8. Case study 2: DAE and NLP solution wall clock timings with increasing number of processors and scenarios.

linear speedup as the number of processors were increased, before observing an expected drop off in efficiency. Ultimately, the cumulative model solution time could be reduced to less than half of the total optimization program run time. Additionally, speedup and efficiency results indicate an appropriate number of processors to use for a given problem size and how to most efficiently scale up the computation resources with problem size.

Acknowledgment

Funding for this work through the McMaster Advanced Control Consortium (MACC) is gratefully acknowledged.

Notation

$\mathbf{x}(t) \in \mathbb{R}^{n_x}$ = differential state variable vector
$\mathbf{z}(t) \in \mathbb{R}^{n_z}$ = algebraic state variable vector
$\mathbf{y}(t) \in \mathbb{R}^{n_y}$ = controlled output variable vector
$\mathbf{u}(t) \in \mathbb{R}^{n_u}$ = open- or closed-loop manipulated input variable vector
$\mathbf{v}(t) \in \mathbb{R}^{n_v}$ = input disturbance variable vector
$\mathbf{p} \in \mathbb{R}^{n_p}$ = model/design parameter vector
$\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ = uncertain parameter vector
$\boldsymbol{\xi}_{0,j} \in \mathbb{R}^{n_x}$ = differential state parameter vector at shooting node j
$\mu_{0,j} \in \mathbb{R}^{n_z}$ = algebraic state parameter vector at shooting node j
$\boldsymbol{\upsilon}_j \in \mathbb{R}^{n_v}$ = open-loop input coefficient parameter vector for shooting interval j
$\boldsymbol{w}_j \in \mathbb{R}^{n_{r1}}$ = shooting interval parameter vector: $n_{r1} := n_x+n_z+n_v$
$\boldsymbol{w} \in \mathbb{R}^{n_{r2}}$ = total shooting interval parameter vector $n_{r2} := (n_x+n_z)(n+1)+n_v n$
$\mathrm{s} \in \mathbb{R}^{n_{r3}}$ = total scenario/shooting parameter vector $n_{r3} := (n_x+n_z)(n+1)n_s+n_v n n_s$
$\mathbf{X}_{j+1}^{\mathcal{M}} \in \mathbb{R}^{n_x \times n_{\mathcal{M}}}$ = differential state sensitivity matrix wrt. $\mathcal{M}$ at node $j+1$
$\mathbf{Q}_j^{\mathcal{M}} \in \mathbb{R}^{n_q \times n_{\mathcal{M}}}$ = point constraint Jacobian matrix wrt. $\mathcal{M}$ at node j: $n_q := n_z + n_g$
$\mathbf{f}_d(\cdot) \in \mathbb{R}^{n_x}$ = explicit differential equation right-hand-side vector
$\mathbf{f}_a(\cdot) \in \mathbb{R}^{n_z}$ = algebraic residual equation vector
$\mathbf{h}_0(\cdot) \in \mathbb{R}^{n_x}$ = differential state initial condition function vector
$\mathbf{g}(\cdot) \in \mathbb{R}^{n_g}$ = path inequality constraint function vector
$\mathcal{U}(t, \boldsymbol{\upsilon}_j) \in \mathbb{R}^{n_u}$ = open-loop input polynomial approximation vector for shooting interval j
$\phi(\cdot) \in \mathbb{R}$ = scalar objective function criterion in Mayer form
$\eta(t) \in \mathbb{R}$ = scalar step approximation function
$\vartheta(\boldsymbol{\gamma}_j, t) \in \mathbb{R}^{n_z}$ = DAE relaxation vector
$\mathbf{c}_j(\boldsymbol{w}_j, \mathbf{p}) \in \mathbb{R}^{n_x}$ = multiple-shooting continuity constraint vector at node j
$\mathbf{q}_j(\boldsymbol{w}_j, \mathbf{p}) \in \mathbb{R}^{n_q}$ = combined point constraint vector at node j
$\mathbf{C}(\boldsymbol{w}, \mathbf{p}) \in \mathbb{R}^{n_{r4}}$ = total multiple-shooting constraint vector: $n_{r4} := (n_x+n_q)(n+1)$
$\mathcal{C}(\mathbf{s}, \mathbf{p}) \in \mathbb{R}^{n_{r5}}$ = total NLP constraint vector for all scenarios: $n_{r5} := (n_x+n_q)(n+1)n_s$

Literature Cited

1. Bansal V, Sakizlis V, Ross R, Perkins JD, Pistikopoulos EN. New algorithms for mixed-integer dynamic optimization. *Comput Chem Eng*. 2003;27(5):647–668.
2. Cervantes A, Biegler LT. Large-scale DAE optimization using a simultaneous NLP formulation. *AIChE J*. 1998;44(5):1038–1050.
3. Colantonio MC, Pytlak R. Dynamic optimization of large-scale systems: case study. *Int J Control*. 1999;72(9):833–841.
4. Fikar M, Latifi MA, Creff Y. Optimal changeover profiles for an industrial depropanizer. *Chem Eng Sci*. 1999;54(13–14):2715–2720.
5. Biegler LT. Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes. SIAM, Philadelphia, 2010.
6. Hartwich A, Stockmann K, Terboven C, Feuerriegel S, Marquardt W. Parallel sensitivity analysis for efficient large-scale dynamic optimization. *Optim Eng*. 2011;12(4):489–508.
7. Leineweber DB, Schafer A, Bock HG, Schloder JP. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part II: Software aspects and applications. *Comput Chem Eng*. 2003;27(2):167–174.
8. Laird CD, Wong AV, Akesson J. Parallel solution of large-scale dynamic optimization problems. In: Pistikopoulos EN, Georgiadis MC, Kokossis AC, editors. 21st European Symposium on Computer Aided Process Engineering – ESCAPE 21, Vol. 21 of *Computer Aided Process Engineering*. Elsevier, Oxford, UK, 2011:813–817.
9. Zhu Y, Legg S, Laird CD. Optimal operation of cyrogenic air separation systems with demand uncertainty and contractual obligations. *Chem Eng Sci*. 2011;66:953–963.
10. Ricardez-Sandoval LA. Optimal design and control of dynamic systems under uncertainty: A probabilistic approach. *Comput Chem Eng*. 2012;43(10):91–107.
11. Morari M, Perkins JD. Design for operations. In: Biegler LT, Doherty MF, editors. Foundations of Computer-Aided Process Design, Vol. 91 of *AIChE Symposium Series, No. 304*. CACHE AIChE, New York, 1995:105–114.
12. van Schijndel JMG, Pistikopoulos EN. Towards the integration of process design, process control & process operability: Current status and future trends. In: Malone MF, Trainham JA, Carnahan B, editors. Foundations of Computer-Aided Process Design. New York: American Institute of Chemical Engineers, 2000:99–112.
13. Sakizlis V, Perkins JD, Pistikopoulos EN. Recent advances in optimization-based simultaneous process and control design. *Comput Chem Eng*. 2004;28(10):2069–2086.
14. Yuan Z, Chen B, Sin G, Gani R. State-of-the-art and progress in the optimization-based simultaneous design and control for chemical processes. *AIChE J*. 2012;58(6):1640–1659.
15. Mohideen MJ, Perkins JD, Pistikopoulos EN. Optimal design of dynamic systems under uncertainty. *AIChE J*. 1996;42(8):2251–2272.
16. Huang R, Patwardhan SC, Biegler LT. Multi-scenario-based robust nonlinear model predictive control with first principle models. In: R M de Brito Alves CAO, Biscaia EC, editors. 10th International Symposium on Process Systems Engineering: Part A, Vol. 27 of *Computer Aided Chemical Engineering*. Elsevier, Oxford, UK, 2009:1293–1298.
17. Shapiro A. Stochastic programming approach to optimization under uncertainty. *Math Programming*. 2008;112(1):183–220.
18. Ierapetritou MG, Acevedo J, Pistikopoulos EN. An optimization approach for process engineering problems under uncertainty. *Comput Chem Eng*. 1996;20(6–7):703–709.
19. Sahinidis NV, Grossmann IE. Reformulation of multiperiod MILP models for planning and scheduling of chemical processes. *Comput Chem Eng*. 1991;15(4):255–272.
20. Ruppen D, Benthack C, Bonvin D. Optimization of batch reactor operation under parametric uncertainty—computational aspects. *J Process Control*. 1995;5(4):235–240.
21. Bhatia TK, Biegler LT. Dynamic optimization for batch design and scheduling with process model uncertainty. *Ind Eng Chem Res*. 1997;36(9):3708–3717.
22. Bhatia TK, Biegler LT. Multiperiod design and planning with interior point methods. *Comput Chem Eng*. 1999;23(7):919–932.
23. Gondzio J, Grothey A. Exploiting structure in parallel implementation of interior point methods for optimization. *Comput Manag Sci*. 2009;6(2):135–160.
24. Shapiro A. Monte Carlo sampling methods. In: Ruszczynski A, Shapiro A, editors. Stochastic Programming, Vol. 10 of *Handbooks in Operations Research and Management Science*. Elsevier, Oxford, UK, 2003:353–425.
25. Diwekar U. Introduction to Applied Optimization, 2nd ed. Springer, New York, 2008.
26. Schweiger CA, Floudas CA. Interaction of design and control: Optimization with dynamic models. In: Hager WW, Pardalos PM, editors. Optimal Control: Theory, Algorithms, and Applications. Kluwer Academic Publishers, New York, 1997:388–435.
27. Bock HG, Plitt KJ. A multiple shooting algorithm for direct solution of optimal control problems. In: Ninth IFAC World Congress. Permagon, Budapest, 1984:242–247.
28. Bock HG, Diehl M, Leineweber DB, Schloder JP. A direct multiple shooting method for real-time optimization of nonlinear DAE processes. In: Allgower F, Zheng A, editors. Nonlinear Model Predictive Control, Vol. 26 of *Progress in Systems and Control Theory*. Birkhauser Basel, Basel, Switzerland, 2000:245–267.
29. Leineweber DB, Bauer I, Bock HG, Schloder JP. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part I: Theoretical aspects. *Comput Chem Eng*. 2003;27(2):157–166.
30. Houska B, Diehl M. A quadratically convergent inexact SQP method for optimal control of differential algebraic equations. *Optimal Control Appl Methods*. 2013;34(4):396–414.
31. Brown P, Hindmarsh A, Petzold L. Consistent initial condition calculation for differential-algebraic systems. *SIAM J Sci Comput*. 1998;19(5):1495–1512.

32. Vassiliadis VS, Sargent RWH, Pantelides CC. Solution of a class of multistage dynamic optimization problems. 2. Problems with path constraints. *Ind Eng Chem Res*. 1994;33(9):2123–2133.
33. Caracotsios M, Stewart WE. Sensitivity analysis of initial value problems with mixed ODEs and algebraic equations. *Comput Chem Eng*. 1985;9(4):359–365.
34. Cao Y, Li S, Petzold L, Serban R. Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution. *SIAM J Sci Comput*. 2003;24(3):1076–1089.
35. Griewank A, Walther A. Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, 2nd ed. SIAM, Philadelphia, 2008.
36. Kiehl M. Parallel multiple shooting for the solution of initial value problems. *Parallel Comput*. 1994;20(3):275–295.
37. Jeon M. Parallel optimal control with multiple shooting, constraints aggregation and adjoint methods. *J Appl Math Comput*. 2005;19(1–2):215–229.
38. Bachmann B, Ochel L, Ruge V, Gerbremedhin M, Fritzson P, Nezhadali V, Erikkson L, Sivertsson M. Parallel multiple-shooting and collocation optimization with Open Modelica. In: Proceedings of the 9th International Modelica Conference, Linkoping University Press, Munich, Germany, 2012:659–668.
39. Gill PE, Murray W, Saunders MA. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Rev*. 2005;47(1):99–131.
40. Hindmarsh AC, Brown PN, Grant KE, Lee SL, Serban R, Shumaker DE, Woodward CS. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Trans Math Software*. 2005;31(3):363–396.
41. Keeping BR, Pantelides CC. A distributed memory parallel algorithm for the efficient computation of sensitivities of differential-algebraic systems. *Math Comput Simul*. 1998;44(6):545–558.
42. Newell RB, Lee PL. Applied Process Control: A Case Study. Prentice Hall, New Jersey, 1989.
43. Kookos IK, Perkins JD. An algorithm for simultaneous process design and control. *Ind Eng Chem Res*. 2001;40(19):4079–4088.
44. Pacheco PS. An Introduction to Parallel Programming. Morgan Kaufmann, New York, 2011.
45. Douglas JM. Conceptual Design of Chemical Processes. McGraw-Hill, New York, 1988.

Appendix A: Evaporator Model Equations

The evaporator model is based on the adaptation of the Newell and Lee[42] model given in Kookos and Perkins.[43] The dynamic closed-loop evaporator model is defined in terms of the differential, $\mathbf{x}(t)$, and algebraic, $\mathbf{z}(t)$, states; the closed-loop manipulated input variables $\mathbf{u}(t)$; the uncertain disturbance input variables, $\mathbf{v}(t)$; and the design parameters, $\mathbf{p}$. The variables and parameters are defined as

$$\mathbf{x}(t) := [x_2, P_2, I_1, I_2, \varepsilon_{\text{ise}}]^\top \tag{A1}$$

$$\mathbf{z}(t) := [T_2, T_3, T_{100}, Q_{100}, F_{100}, F_4, F_2, Q_{200}, T_{201}, F_5]^\top \tag{A2}$$

$$\mathbf{u}(t) := [P_{100}, F_{200}]^\top \tag{A3}$$

$$\mathbf{v}(t) := [F_1, x_1]^\top \tag{A4}$$

$$\mathbf{p} := [UA_1, UA_2, K_{c_1}, \tau_{I_1}, \bar{x}_2, K_{c_2}, \tau_{I_2}, \bar{P}_2]^\top \tag{A5}$$

The system of nonlinear first-order ordinary differential equations making up the model include: a balance on composition x_2; a relation balance for pressure P_2; integrated output error equations for I_1 and I_2; and a cumulative integrated squared error equation to determine $\varepsilon_{\text{ise}}(t_f)$

$$M\dot{x}_2 - (F_1 x_1 - F_2 x_2) = 0 \tag{A6}$$

$$C\dot{P}_2 - (F_4 - F_5) = 0 \tag{A7}$$

$$\dot{I}_1 - (x_2 - \bar{x}_2) = 0 \tag{A8}$$

$$\dot{I}_2 - (P_2 - \bar{P}_2) = 0 \tag{A9}$$

$$\dot{\varepsilon}_{\text{ise}} - \left((x_2 - \bar{x}_2)^2 + (P_2 - \bar{P}_2)^2\right) = 0 \tag{A10}$$

The algebraic equations used to explicitly define the algebraic variables, and subsequently eliminate them from the model, are defined as follows

$$T_2 := 0.5616\,P_2 + 0.3126\,x_2 + 48.43 \tag{A11}$$

$$T_3 := 0.507\,P_2 + 55 \tag{A12}$$

$$T_{100} := 0.1538\,P_{100} + 90 \tag{A13}$$

$$Q_{100} := UA_1(T_{100} - T_2) \tag{A14}$$

$$F_{100} := Q_{100}/\lambda_s \tag{A15}$$

$$F_4 := \frac{Q_{100} + F_1 C_p (T_1 - T_2)}{\lambda + C_p (T_3 - T_2)} \tag{A16}$$

$$F_2 := F_1 - F_4 \tag{A17}$$

$$Q_{200} := \frac{2C_p F_{200} UA_2 (T_3 - T_{200})}{2C_p F_{200} + UA_2} \tag{A18}$$

$$T_{201} := T_{200} + \frac{Q_{200}}{F_{200} C_p} \tag{A19}$$

$$F_5 := Q_{200}/\lambda \tag{A20}$$

Additionally, the algebraic controller and disturbance expressions are given as

$$P_{100} := \bar{P}_{100} + K_{c_1}(x_2 - \bar{x}_2) + \frac{K_{c_1}}{\tau_{I_1}} I_1 \tag{A21}$$

$$F_{200} := \bar{F}_{200} + K_{c_2}(P_2 - \bar{P}_2) + \frac{K_{c_2}}{\tau_{I_2}} I_2 \tag{A22}$$

$$F_1 := \bar{F}_1 + \Delta\bar{F}_1\,\eta \tag{A23}$$

$$x_1 := \bar{x}_1 + \Delta\bar{x}_1\,\eta \tag{A24}$$

Expressions for the evaporator capital and operating costs were obtained from Douglas.[45] In our calculations, we consider the annualized process capital cost defined based on the evaporator unit and the overhead condenser as a function of the overall surface areas for each unit, UA_1 and UA_2, respectively. The operating cost is based on the average steam and cooling water flow rates over a one-year period

$$C_{\text{evap}} := 5463\,(UA_1)^{0.65} \tag{A25}$$

$$C_{\text{cond}} := 2820\,(UA_2)^{0.65} \tag{A26}$$

$$C_{\text{steam}} := c_s\,\Delta H_{\text{vap}}\,F_{100}(t_f)\,T_A = 4890\,F_{100}(t_f) \tag{A27}$$

$$C_{cw} := c_{cw}\,\Delta H_{\text{cond}}\,F_{200}(t_f)\,T_A = 4.9\,F_{200}(t_f) \tag{A28}$$

The final capital and operating cost expressions are

$$C_{\text{cap}}(\mathbf{p}) := \frac{1}{\beta_p}(C_{\text{evap}} + C_{\text{cond}}) \tag{A29}$$

$$C_{\text{op}}(\mathbf{x}(t_f), \mathbf{u}(t_f), \mathbf{v}(t_f), \mathbf{p}, t_f) := \beta_t(C_{\text{steam}} + C_{cw}) \tag{A30}$$

where β_p and β_t are the payback period and tax fraction, respectively. We remark that the operating cost expression evaluated at t_f assumes a sustained period of steady-state operation at the final time conditions. Table A1 defines the model parameters.

Appendix B: Distillation Model Equations

The binary distillation model was adapted from Schweiger and Floudas.[26] The variables for the distillation model are defined

Table A1. Case Study 1: Evaporator Model Parameter Values

Description	Symbol	Value	Units
Cooling water heat capacity	C_p	0.07	kW min/(K kg)
Latent heat of evaporation (water)	λ	38.5	kW min/kg
Latent heat of steam (saturated)	λ_s	36.6	kW min/kg
Feed temperature	T_1	40	°C
Cooling water inlet temperature	T_{200}	25	°C
Evaporator liquid holdup	M	20	kg
Pressure equation parameter	C	4	kg min/kPa
Steam cost parameter	$c_s\,\Delta H_{\text{vap}}$	1.0×10^{-2}	\$/kg
Cooling water cost parameter	$c_{cw}\,\Delta H_{\text{cond}}$	1.0×10^{-5}	\$/kg
Payback period	β_p	3	yr
Tax rate fraction	β_t	1	–
Annual operating time	T_A	8150	h/yr
Initial feed flow rate interval	$\bar{F}_1$	[9.75, 10.25]	kg/min
Feed flow rate step interval	$\Delta\bar{F}_1$	[−0.25, 0.25]	kg/min
Initial feed composition interval	$\bar{x}_1$	[4.75, 5.25]	%
Feed composition step interval	$\Delta\bar{x}_1$	[−0.25, 0.25]	%
Product composition bounds	x_2	[25, 30]	%
Operating pressure bounds	P_2	[40, 80]	kPa
Steam pressure upper bound	P_{100}^U	400	kPa
Cooling water flow rate upper bound	F_{200}^U	600	kg/min

on a stage-wise basis where we label each stage starting from the reboiler at $n=0$, proceeding with each tray from $n=1,\ldots,n_t$, and ending with the condenser at $n=n_t+1$. The variables and parameters are defined as

$$\mathbf{x}_0(t) := [x_0, L_0]^\top \tag{B1}$$

$$\mathbf{x}_n(t) := [x_n, L_n]^\top \forall n \in \mathcal{N}\backslash\{0, n_t+1\} \tag{B2}$$

$$\mathbf{x}_{n_t+1}(t) := [x_{n_t+1}, D]^\top \tag{B3}$$

$$\mathbf{x}(t) := [\mathbf{x}_0(t)^\top, \ldots, \mathbf{x}_{n_t+1}(t)^\top, I_1, I_2, \varepsilon_{\text{ise}}]^\top \tag{B4}$$

$$\mathbf{z}_n(t) := [M_n^l, y_n]^\top \forall n \in \mathcal{N}\backslash\{n_t+1\} \tag{B5}$$

$$\mathbf{z}_{n_t+1}(t) := M_{n_t+1}^l \tag{B6}$$

$$\mathbf{z}(t) := [\mathbf{z}_0(t)^\top, \ldots, \mathbf{z}_{n_t+1}(t)^\top]^\top \tag{B7}$$

$$\mathbf{y}(t) := [x_0, x_{n_t+1}]^\top \tag{B8}$$

$$\mathbf{u}(t) := [R,\ V]^\top \tag{B9}$$

$$\mathbf{v}(t) := [F, z]^\top \tag{B10}$$

$$\mathbf{p} := [D_{\text{col}}, K_{c_1}, \tau_{I_1}, \bar{x}_0, K_{c_2}, \tau_{I_2}, \bar{x}_{n_t+1}]^\top \tag{B11}$$

where we use the index set $\mathcal{N} := \{0, \ldots, n_t+1\}$ to represent all stages in the column. A detailed explanation of the system variables is given in Table B1. The system of nonlinear first-order ordinary differential equations making up the model include: a relation balance for the liquid flow down the column for each stage; a composition balance for the light key component for each stage; integrated output error equations for I_1 and I_2; and a cumulative integrated squared error equation to determine $\varepsilon_{\text{ise}}(t_f)$

$$\tau_0\dot{L}_0 - (L_1 - (L_0+V)) = 0 \tag{B12}$$

$$\tau\dot{L}_n - (L_{n+1} - L_n + \mathcal{F}_n) = 0 \qquad \forall n \in \mathcal{N}\backslash\{0, n_t, n_t+1\} \tag{B13}$$

$$\tau\dot{L}_{n_t} - (R - L_{n_t}) = 0 \tag{B14}$$

$$\tau_{n_t+1}\dot{D} - (V - (R+D)) = 0 \tag{B15}$$

$$M_0^l\dot{x}_0 - (L_1(x_1 - x_0) + V(x_0 - y_0)) = 0 \tag{B16}$$

$$M_n^l\dot{x}_n - (L_{n+1}(x_{n+1} - x_n) + V(y_{n-1} - y_n) + \mathcal{F}ZX_n) = 0 \quad \forall n \in \mathcal{N}\backslash\{0, n_t, n_t+1\} \tag{B17}$$

$$M_{n_t}^l\dot{x}_{n_t} - (R(x_{n_t+1} - x_{n_t}) + V(y_{n_t-1} - y_{n_t})) = 0 \tag{B18}$$

$$M_{n_t+1}^l\dot{x}_{n_t+1} - V(y_{n_t} - x_{n_t+1}) = 0 \tag{B19}$$

$$\dot{I}_1 - (x_0 - \bar{x}_0) = 0 \tag{B20}$$

$$\dot{I}_2 - (x_{n_t+1} - \bar{x}_{n_t+1}) = 0 \tag{B21}$$

$$\dot{\varepsilon}_{\text{ise}} - \left((x_0 - \bar{x}_0)^2 + (x_{n_t+1} - \bar{x}_{n_t+1})^2\right) = 0 \tag{B22}$$

The algebraic equations used to explicitly define the algebraic variables are given as follows

$$y_n := \alpha x_n/(1 + x_n(\alpha - 1)) \forall n \in \mathcal{N}\backslash\{n_t+1\} \tag{B23}$$

$$M_n^l \approx M := \gamma_1 D_{\text{col}}^2 (h_{\text{weir}} + \gamma_2/D_{\text{col}}^{2/3}) \tag{B24}$$

$$\tau := \gamma_3 D_{\text{col}}^{4/3} \tag{B25}$$

$$\tau_0 = \tau_{n_t+1} := 100\,\tau \tag{B26}$$

$$M_0^l = M_{n_t+1}^l := 10\,M \tag{B27}$$

$$\mathcal{F}_n := \begin{cases} F & \text{if } n = n_f \\ 0 & \text{otherwise} \end{cases} \tag{B28}$$

Table B1. Case Study 2: Distillation Model Variable Definitions

x_n: liquid composition	y_n: vapor composition
L_n: liquid flow rate (kmol/min)	$I_{1,2}$: integrated controller error
z: feed composition	ε_{ise}: cumulative integrated squared error
F: liquid feed flow rate (kmol/min)	M_n^l: liquid molar holdup (kmol)
R: reflux flow rate (kmol/min)	D: distillate flow rate (kmol/min)
V: vapor boilup flow rate (kmol/min)	

Table B2. Case Study 2: Distillation Model Parameter Values

Description	Symbol	Value	Units
No. of trays	n_t	16	–
Feed tray location	n_f	8	–
Relative volatility	α	2.5	–
Weir height	h_{weir}	0.0254	m
Tray spacing	S_{tray}	0.5080	m
Tray holdup parameter	γ_1	6.0305	kmol/m^3
Tray holdup parameter	γ_2	0.008929	m$^{5/3}$
Time constant parameter	γ_3	0.05271	min m$^{-4/3}$
Flood constraint parameter	γ_4	0.6719	m min$^{1/2}$ kmol$^{-1/2}$
Steam cost	c_s	4.99×10^{-7}	\$/kJ
Cooling water cost	c_{cw}	1.23×10^{-8}	\$/kJ
Heat of vaporization	ΔH_{vap}	3.1×10^{4}	kJ/kmol
Heat of condensation	ΔH_{cond}	3.2×10^{4}	kJ/kmol
Initial feed flow rate interval	$\bar{F}$	[0.5, 1.5]	kmol/min
Feed flow rate step interval	$\Delta\bar{F}$	[−0.5, 0.5]	kmol/min
Initial feed composition interval	$\bar{z}$	[0.3, 0.6]	–
Feed composition step interval	$\Delta\bar{z}$	[−0.1, 0.1]	–
Condenser composition lower bound	$x^L_{n_t+1}$	0.98	–
Reboiler composition upper bound	x^U_0	0.05	–

$$\mathcal{F}ZX_n := \begin{cases} F(z-x_n) & \text{if } n=n_f \\ 0 & \text{otherwise} \end{cases} \tag{B29}$$

$$D^{\min}_{\text{col}} := \gamma_4 \bar{V}^{0.5} \tag{B30}$$

Additionally, the algebraic controller and disturbance expressions are given as

$$V := \bar{V} + K_{c_1}(x_0 - \bar{x}_0) + \frac{K_{c_1}}{\tau_{I_1}} I_1 \tag{B31}$$

$$R := \bar{R} + K_{c_2}(x_{n_t+1} - \bar{x}_{n_t+1}) + \frac{K_{c_2}}{\tau_{I_2}} I_2 \tag{B32}$$

$$F := \bar{F} + \Delta\bar{F}\,\eta \tag{B33}$$

$$z := \bar{z} + \Delta\bar{z}\,\eta \tag{B34}$$

Expressions for the distillation column capital and operating costs were again obtained from Douglas[45] and are defined in a simplified form as

$$C_{\text{tray}} := 95.5\, D^{1.55}_{\text{col}}\, H_{\text{col}} \tag{B35}$$

$$C_{\text{shell}} := 2928\, D^{1.066}_{\text{col}}\, H^{0.802}_{\text{col}} \tag{B36}$$

$$C_{\text{utility}} := (c_s\,\Delta H_{\text{vap}} + c_{cw}\,\Delta H_{\text{cond}})\,\bar{V}\,T_A = 7756\,\bar{V} \tag{B37}$$

where the column height is defined as $H_{\text{col}} = S_{\text{tray}}\, n_t$. The final capital and operating cost expressions (in \$/yr) are

$$C_{\text{cap}}(\mathbf{p}) := \frac{1}{\beta_p}(C_{\text{tray}} + C_{\text{shell}}) \tag{B38}$$

$$C_{\text{op}}(\mathbf{x}(t_f), \mathbf{u}(t_f), \mathbf{v}(t_f), \mathbf{p}, t_f) := \beta_t\, C_{\text{utility}} \tag{B39}$$

The model parameters used in defining the distillation model are listed in Table B2.

Manuscript received Nov. 19, 2013, and revision received Apr. 10, 2014.