I&EC research
Industrial & Engineering Chemistry Research

Article

pubs.acs.org/IECR

An Analytical Hessian and Parallel-Computing Approach for Efficient Dynamic Optimization Based on Control-Variable Correlation Analysis

Evgeny Lazutkin, Abebe Geletu, Siegbert Hopfgarten, and Pu Li*

Simulation and Optimal Processes Group, Institute for Automation and Systems Engineering, Technische Universität Ilmenau, P.O. Box 100565, 98684 Ilmenau, Germany

ABSTRACT: The approach of combined multiple-shooting with collocation is efficient for solving large-scale dynamic optimization problems. The aim of this work was to further improve its computational performance by providing an analytical Hessian and realizing a parallel-computing scheme. First, we derived the formulas for computing the second-order sensitivities for the combined approach. Second, a correlation analysis of control variables was introduced to determine the necessity of employing the analytical Hessian to solve an optimization problem. Third, parallel computing was implemented thanks to the nature of the combined approach, because the solutions of model equations and evaluations of both first-order and second-order sensitivities for individual time intervals are decoupled. Because these computations are expensive, a high speedup factor was gained through the parallelization. The performance of the proposed analytical Hessian, correlation analysis, and parallel computing is demonstrated in this article by benchmark problems including optimal control of a distillation column containing more than 1000 dynamic variables.

1. INTRODUCTION

The solution of large-scale dynamic optimization problems with differential-algebraic equations (DAEs) still poses a great challenge especially for online implementations such as nonlinear model predictive control and real-time parameter estimation.[1−3] Although many efficient approaches have been developed to address large-scale dynamic optimization problems, further studies are needed to enhance the efficiency of existing solution algorithms in the context of discretization, sensitivity calculation, and parallel computing.

Two major and widely used discretization methods are collocation on finite elements[4−6] and the multiple-shooting strategy.[7−9] In the collocation approach, the DAE system is fully discretized, leading to a large number of optimization variables in the nonlinear programming (NLP) formulation along with a corresponding number of equality constraints. On the other hand, the multiple-shooting approach decomposes the dynamic optimization problem into independent treatment in individual time intervals, thus allowing for the easy implementation of parallel computing.[10]

The control vector parametrization (CVP) method is known as a sequential approach.[11,12] Its basic idea is to discretize control variables over time intervals and obtain the states by solving DAEs by an integration scheme. A quasi-sequential approach was proposed by Hong et al.[13] in which only control variables are treated as decision variables and collocation on finite elements is used for discretization, leading to a two-stage solution strategy. An adaptive collocation scheme for this approach was introduced by Bartl et al.[14]

Recently, a combined multiple-shooting and collocation (CMSC) approach was proposed to exploit the advantages of both collocation and multiple-shooting.[15] In this approach, the discretization of model equations is based on the collocation method for individual time intervals, whereas optimality and state continuity are ensured in the resulting NLP in the context of the multiple-shooting method. It has been demonstrated that this approach is highly efficient for solving dynamic optimization problems.[15,16] In this study, the advantages of this approach are further exploited by introducing an analytical Hessian (AH) and implementing parallel computing.

It is known that the use of exact second-order sensitivities to provide an AH is beneficial for the solution of the resulting NLP problem. The results obtained using CVP indicate that exact second-order sensitivities, obtained by deriving and solving a differential equation system over the whole time horizon, can enhance robustness and convergence rate.[17−19] An extension of this method was made in the framework of collocation on finite elements and considering piecewise constant controls.[20,21] Moreover, there is wide interest in automatic sensitivity computation.[22,23] However, the computation of an AH is known to be highly expensive, which might counterbalance its advantages. Therefore, a method to examine the profitability of using an AH instead of a numerical approximation is highly desired.

Another way to enhance the efficiency of solving dynamic optimization problems is to decompose the resulting NLP problem and implement a parallel-computing strategy. The single-shooting method was implemented using parallel computation of first-order sensitivities.[24] Parallel computing was also implemented in the context of multiple-shooting involving DAE integration and sensitivity computation.[10] In the context of collocation on finite elements, parallel computing was realized by a Schur-complement decomposition within an

Received: June 30, 2015
Revised: November 3, 2015
Accepted: November 12, 2015
Published: November 12, 2015

DOI: 10.1021/acs.iecr.5b02369
Ind. Eng. Chem. Res. 2015, 54, 12086−12095

interior-point algorithm.[25] This decomposition method introduces coupling variables to ensure the continuity of state trajectories, leading to high computational costs. This approach was further improved by Kang et al.[26] by utilizing the block structure of the resulting NLP. In the work of Zavala et al.,[27] the optimizer was also adapted to the Schur-complement method to provide parallel computing for solving parameter estimation problems. More recently, Washington and Swartz[28] implemented a parallel-computing scheme in the context of multiple-shooting, but state trajectories in each time interval were computed using a numerical integrator.[29] In addition, the CMSC method was also implemented using parallel computing in an open-source environment, but with numerically approximated second-order derivatives.[30]

In summary, although much progress has been made in improving computation efficiency, further studies need to be carried out to explore the potential of the existing approaches. Therefore, the aim of this study is to extend the CMSC method in such a way that the computation efficiency can be enhanced in comparison to those of the existing approaches. This extension leads to following novel contributions: First, the formulas for analytical second-order sensitivities are derived in the context of the CMSC method. As a result, an AH can be used to improve the performance of the resulting NLP. However, the computational cost for an AH is higher than that for a numerical approximation such as the Broyden−Fletcher−Goldfarb−Shanno (BFGS) method. Therefore, as a second contribution, we introduce a new method for evaluating the profitability of using an AH by analyzing correlations of control (independent) variables of dynamic optimization problems. Correlation relationships are analyzed analytically by checking linear dependencies of the columns of the sensitivity matrix and numerically by calculating the angles between the corresponding columns. The latter is achieved by a simulation step with control signals in the form of a pseudorandom binary sequence. Consequently, the degree of difficulty for solving a dynamic optimization problem under consideration can be examined a priori. Third, by employing the decomposed structure of the CMSC method, parallel computing is implemented in Python for computing state values and both first- and second-order sensitivities using automatic differentiation. Through case studies, this parallel-computing strategy can speed up the computation considerably, particularly for large-scale problems.

The rest of the article is organized as follows: Section 2 presents the formulation of dynamic optimization problems, the transformation of such a problem into an NLP using the CMSC method, and the derivation of first- and second-order sensitivities in the CMSC framework. The method of correlation analysis for control variables is introduced in section 3. Section 4 presents a parallel-computing strategy and its numerical implementation. The computation efficiency of the proposed approach is demonstrated by case studies in section 5. In particular, a distillation column control problem with more than 1000 dynamic variables is solved, showing the capability of our approach for solving large-scale problems. The article ends with conclusions in section 6.

2. PROBLEM FORMULATION AND SOLUTION APPROACH

2.1. Problem Description. This work considers dynamic nonlinear optimization problems of the form

$$\min_{u(t)}\{J = M[x(t_f)] + \int_{t_0}^{t_f} L[x(t), z(t), u(t), t]\,dt\}$$

subject to

$$\dot{x}(t) = f[x(t), z(t), u(t), t], \quad x(t_0) = x_0$$

$$g[x(t), z(t), u(t), t] = 0, t_0$$

$$x_{min} \le x(t) \le x_{max}$$

$$z_{min} \le z(t) \le z_{max}$$

$$u_{min} \le u(t) \le u_{max} \tag{1}$$

where $x(t) \in \Re^{n_x}$ and $z(t) \in \Re^{n_z}$ represent the vectors of state and algebraic variables, respectively, and $u(t) \in \Re^{n_u}$ is the control vector. The time horizon is defined as $t \in [t_0, t_f]$, with initial time t_0 and final time t_f. The initial state $x(t_0)$ is usually known, whereas the final state $x(t_f)$ can be either fixed or free. The functions $f: \Re^{n_x} \times \Re^{n_z} \times \Re^{n_u} \times [t_0, t_f] \to \Re^{n_x}$ and $g: \Re^{n_x} \times \Re^{n_z} \times \Re^{n_u} \times [t_0, t_f] \to \Re^{n_z}$ in model equations, and the objective function J, consisting of a Mayer term $M: \Re^{n_x} \to \Re$ and a Lagrange term $L: \Re^{n_x} \times \Re^{n_z} \times \Re^{n_u} \times [t_0, t_f] \to \Re$, are assumed to be second-order-differentiable. Because of physical limits, controls $u(t)$, states $x(t)$, and algebraic variables $z(t)$ are usually bounded, as indicated by the box-constraint inequalities.

2.2. Transformation to NLP. In this study, we use the CMSC method to transform dynamic optimization problem 1 into a finite-dimensional NLP. The time horizon $[t_0, t_f]$ is divided into N intervals, and in each interval, the state and algebraic variables are treated using the multiple-shooting method. Subsequently, on each time interval, state and control variables are parametrized as initial values and piecewise constants, respectively; that is, $\mathbf{X}^p = [X_{p,0}, X_{p,1}, ..., X_{p,N}]$ and $\mathbf{V} = [V_0, V_1, ..., V_{N-1}]$. Equality constraints $X_{p,i+1} = X^c(t_{i+1}; X_{p,i}, V_i)$ are introduced to ensure continuity of the states between interval i and $i + 1$. Because the decision variables are $\mathbf{X}^p \in \Re^{n_x(N+1)}$ and $\mathbf{V} \in \Re^{n_u N}$, problem 1 is now transformed to the following NLP formulation

$$\min_{\mathbf{X}^p, \mathbf{V}}\{M(X_{p,N}) + \sum_{i=0}^{N-1} L_i(X_{p,i}, V_i)\}$$

subject to:

$$X_{p,i+1} = X^c(t_{i+1}; X_{p,i}, V_i), \quad i = 0,...,N-1$$

$$X_{p,0} = X_0$$

$$x_{min} \le \mathbf{X}^p \le x_{max}$$

$$u_{min} \le \mathbf{V} \le u_{max} \tag{2}$$

Note that model equations are not directly involved in the NLP formulation, because in each time interval, discretized model equations are solved in a Newton step to obtain state trajectories and both first- and second-order sensitivities as described in the following subsections. More details about the CMCS method can be found in Tamimi and Li.[15]

2.3. Computation of First-Order Sensitivities. In the CMSC method, for each time interval, a collocation method is employed. The state and algebraic variables are approximated inside each time interval by a linear combination of the Lagrange polynomials using the Radau scheme with N_c

DOI: 10.1021/acs.iecr.5b02369
Ind. Eng. Chem. Res. 2015, 54, 12086−12095

collocation points. After this, the state and algebraic variables X^c at all collocation points have implicit dependency on the decision variables X^p and V. Therefore, the discretized dynamic model equations for each time interval can be written in the compact form

$$[G(X^c(X^p, V), X^p, V)]_{N_{xz}\times 1} = \mathbf{0} \tag{3}$$

where $N_{xz} = (n_x + n_z)N_c$ and $X^c \in \Re^{N_{xz}}$. For the sake of brevity, the index for time intervals is dropped here. To obtain the first-order sensitivities for individual time intervals, two differentiation operators $\partial/\partial X^p$ and $\partial/\partial V$ are applied to eq 3, which yields

$$\left[\frac{\partial G}{\partial X^c}\right]_{N_{xz}\times N_{xz}}\left[\frac{\partial X^c}{\partial X^p}\right]_{N_{xz}\times n_x} = -\left[\frac{\partial G}{\partial X^p}\right]_{N_{xz}\times n_x} \tag{4}$$

$$\left[\frac{\partial G}{\partial X^c}\right]_{N_{xz}\times N_{xz}}\left[\frac{\partial X^c}{\partial V}\right]_{N_{xz}\times n_u} = -\left[\frac{\partial G}{\partial V}\right]_{N_{xz}\times n_u} \tag{5}$$

The construction of the linear equation systems for the first-order sensitivities can be done in a straightforward manner using automatic differentiation. Because of the CMSC method, eqs 4 and 5 have a sparse structure and thus can be solved for $\partial X^c/\partial X^p$ and $\partial X^c/\partial V$ by a linear sparse algebra algorithm.

2.4. Computation of Second-Order Sensitivities. In this subsection, we derive the formulas for computing second-order sensitivities for the CMSC method. It should be noted that, in general, there are two types of second-order sensitivities: derivatives of functions with respect to all variables (both states and controls) in the context of simultaneous approaches and derivatives of state variables with respect to controls in the context of (quasi-) sequential approaches. The CMSC method is a kind of sequential approach, and thus, we need to compute the second-order derivatives of states with respect to controls. In comparison with the approaches of previous studies,[17,21] where sensitivities have to be transferred from one time interval to the next (so-called global sensitivities) in the context of single shooting, we employ the advantage of the CMSC method that the computations of the second-order sensitivities in different time intervals are independent. The second-order sensitivities of the state continuity condition (i.e., the equality constraints of problem 2) are obtained simply as those on the last collocation point of the time intervals.

In each time interval, the first-order sensitivities $\partial X^c/\partial X^p$ and $\partial X^c/\partial V$ solved from eqs 4 and 5 have implicit dependency on the decision variables of only this interval. To clearly explain the derivation, the first-order sensitivity (matrix) eqs 4 is reformulated as a vector of linear equations. This is made by multiplying the matrix $\partial G/\partial X^c$ with the vector $[\partial X^c/\partial X^p]^k$ and the result is added with the vector $[\partial G/\partial X^p]^k$, where $k = 1, ..., n_x$ is the column number. This procedure is also done for eq 5, where $k = 1, ..., n_u$. As a result, eqs 4 and 5 can be rewritten in the compact form

$$\left[\Phi\left(X^c(X^p, V), X^p, V, \frac{\partial X^c}{\partial X^p}(X^p, V)\right)\right]_{N_{XZ}^X\times 1} = \mathbf{0} \tag{6}$$

$$\left[\Psi\left(X^c(X^p, V), X^p, V, \frac{\partial X^c}{\partial V}(X^p, V)\right)\right]_{N_{XZ}^U\times 1} = \mathbf{0} \tag{7}$$

where $N_{XZ}^X = N_{xz}n_x$ and $N_{XZ}^U = N_{xz}n_u$. Applying both differentiation operators $\partial/\partial X^p$ and $\partial/\partial V$ to eqs 6 and 7, the equations for second-order sensitivities can be expressed as follows

$$\left[\frac{\partial \Phi}{\partial\left(\frac{\partial X^c}{\partial X^p}\right)}\right]_{N_{XZ}^X\times N_{XZ}^X}\left[\frac{\partial^2 X^c}{\partial X^{p,2}}\right]_{N_{XZ}^X\times n_x} = -\left[\frac{\partial \Phi}{\partial X^c}\right]_{N_{XZ}^X\times N_{xz}}\frac{\partial X^c}{\partial X^p} - \left[\frac{\partial \Phi}{\partial X^p}\right]_{N_{XZ}^X\times n_x} \tag{8}$$

$$\left[\frac{\partial \Phi}{\partial\left(\frac{\partial X^c}{\partial X^p}\right)}\right]_{N_{XZ}^X\times N_{XZ}^X}\left[\frac{\partial^2 X^c}{\partial X^p \partial V}\right]_{N_{XZ}^X\times n_u} = -\left[\frac{\partial \Phi}{\partial X^c}\right]_{N_{XZ}^X\times N_{xz}}\frac{\partial X^c}{\partial V} - \left[\frac{\partial \Phi}{\partial V}\right]_{N_{XZ}^X\times n_u} \tag{9}$$

$$\left[\frac{\partial \Psi}{\partial\left(\frac{\partial X^c}{\partial V}\right)}\right]_{N_{XZ}^U\times N_{XZ}^U}\left[\frac{\partial^2 X^c}{\partial V \partial X^p}\right]_{N_{XZ}^U\times n_x} = -\left[\frac{\partial \Psi}{\partial X^c}\right]_{N_{XZ}^U\times N_{xz}}\frac{\partial X^c}{\partial X^p} - \left[\frac{\partial \Psi}{\partial X^p}\right]_{N_{XZ}^U\times n_x} \tag{10}$$

$$\left[\frac{\partial \Psi}{\partial\left(\frac{\partial X^c}{\partial V}\right)}\right]_{N_{XZ}^U\times N_{XZ}^U}\left[\frac{\partial^2 X^c}{\partial V^2}\right]_{N_{XZ}^U\times n_u} = -\left[\frac{\partial \Psi}{\partial X^c}\right]_{N_{XZ}^U\times N_{xz}}\frac{\partial X^c}{\partial V} - \left[\frac{\partial \Psi}{\partial V}\right]_{N_{XZ}^U\times n_u} \tag{11}$$

It can be seen that the matrix dimensions grow rapidly as the numbers of state and control variables increases, as indicated in eqs 8−11. In these equations, the partial derivative matrices can be generated by applying automatic differentiation. Note that eq 10 can be eliminated, because the sensitivities $\partial^2 X^c/(\partial X^p \partial V)$ and $\partial^2 X^c/(\partial V \partial X^p)$ are exactly the same. Then, the second-order derivatives can be calculated by solving the above linear matrix equations by a sparse linear solver, and the results provide an AH for the CMSC method.

However, there are two major obstacles to using an AH, especially for solving large-scale problems. First, in general, the generation of an AH is complicated. Nevertheless, it is quite straightforward using our approach as described above. Second, the computation time for each NLP iteration of an AH can be expensive; specifically, the computation time for each NLP iteration using an AH is higher than that using a BFGS approximation. Hence, an improvement in computation time for solving an NLP can be achieved only if the difference between the numbers of NLP iterations required when using an AH approach and a BFGS formula is sufficiently large. Therefore, in the next section, a method based on a priori simulation is proposed to examine whether an AH is beneficial for solving dynamic optimization problems.

3. CORRELATION ANALYSIS OF CONTROL VARIABLES

The basic idea of correlation analysis comes from parameter estimation problems where a model under consideration will be nonidentifiable if there are strong correlations between model parameters.[31−33] In such situations, the NLP solver will

DOI: 10.1021/acs.iecr.5b02369
Ind. Eng. Chem. Res. 2015, 54, 12086−12095

experience difficulties in attempting to converge to an optimal point, because the resulting sensitivity matrix tends to be singular or ill-conditioned.

Based on these considerations, we propose a novel approach to determine the use of an analytical or approximate Hessian by analyzing control-variable correlations of the dynamic optimization problem. This is done through a priori simulation using proper input control profiles. As in parameter estimation, the convergence to an optimal point will be slow if there are strong correlations between control variables. Such correlations can arise from improper modeling of the system.

Now, we investigate correlations of control variables in dynamic systems described by

$$\dot{x}(t) = f[\mathbf{x}(t), \mathbf{u}(t), t], \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{12}$$

where $\mathbf{x}(t) \in \Re^n$ and $\mathbf{u}(t) \in \Re^m$ are state and control vectors, respectively. The known initial state vector is given by $\mathbf{x}_0$. For the purpose of optimal control, the controls $\mathbf{u}(t)$ are independent variables and have to be determined by the optimization.

Mathematically, a correlation between u_i and u_j (U−U correlation) means that the two control variables have a functional relationship. The physical meaning of such a correlation is that the effects of u_i and u_j on the system will be compensated. A straightforward way to identify U−U correlations in a system described by eq 12 is to analyze the state sensitivity matrix to the controls ($\partial\mathbf{x}/\partial\mathbf{u}$). If there are linearly dependent columns in this matrix, then the corresponding controls are correlated.

Based on eq 12, the sensitivities of states to the controls can be expressed as

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\right) = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)\left(\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\right) + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right) \tag{13}$$

If the initial state of the system $\mathbf{x}_0$ is a steady state, the initial condition of eq 13 can be written as

$$\left(\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\bigg|_{t=t_0}\right) = -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\bigg|_{t=t_0}\right)^{-1}\left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\bigg|_{t=t_0}\right) \tag{14}$$

The solution of eq 13 leads to

$$\left(\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\right) = \exp\left[\int_{t_0}^{t}\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)\mathrm{d}\tau\right]\left(\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\bigg|_{t=t_0}\right) + \int_{t_0}^{t}\exp\left[\int_{\omega}^{t}\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)\mathrm{d}\omega\right]\left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right)\mathrm{d}\tau \tag{15}$$

Based on eqs 14 and 15, $\partial\mathbf{x}/\partial\mathbf{u}$ has a linear (integral) relation with $\partial\mathbf{f}/\partial\mathbf{u}$ from t_0 to t. Consequently, the corresponding columns in $\partial\mathbf{x}/\partial\mathbf{u}$ will be linearly dependent, that is, the corresponding control variables will be correlated, if, at any time, $\partial\mathbf{f}/\partial\mathbf{u}$ has the same linearly dependent columns. As a result, U−U correlations can be identified by analyzing the linear dependence of the columns of the function sensitivity matrix $\partial\mathbf{f}/\partial\mathbf{u}$, which is easy to achieve.

If the following equality holds true

$$\left(\frac{\partial \mathbf{f}}{\partial u_i}\right) = \alpha_{i,j}\left(\frac{\partial \mathbf{f}}{\partial u_j}\right) \tag{16}$$

then there is a correlation between u_i and u_j. If the coefficient $\alpha_{i,j}$ is a constant, it is a *structural* correlation, that is, this correlation does not depend on the control variables. If the coefficient $\alpha_{i,j}$ is a function of the control variables, $\alpha_{i,j}(u_i,u_j)$, it is a *practical* correlation and can then be resolved and remedied by selecting proper control profiles. In this way, the dependence of a U−U correlation of the controls can be identified. As a simple example, consider

$$\dot{x} = -(u_1 + u_2)x \tag{17}$$

Then $\partial f/\partial u_1 = \partial f/\partial u_2$; that is, u_1 and u_2 have a structural correlation. If we have

$$\dot{x} = -u_1u_2x \tag{18}$$

then $\partial f/\partial u_1 = (u_2/u_1)(\partial f/\partial u_2)$, which means that u_1 and u_2 are practically correlated. It is noted that this correlation analysis is not limited to handling differential equations. It can be easily extended to handle DAE systems.

However, the method of correlation analysis presented above can only explicitly determine whether there is or is not a correlation in a system under consideration, that is, it provides a result that two control variables are either 0% or 100% correlated. Nevertheless, control variables in a system are usually correlated between 0% (weak correlation) and 100% (strong correlation). Therefore, an index is needed to quantify the degree of a correlation between two controls. Such an index can readily be made by calculating the angles of the corresponding columns in the matrix of $\partial\mathbf{x}/\partial\mathbf{u}$. If the control profiles are given, one can obtain this matrix through simulation. Considering two columns in the matrix $\partial\mathbf{x}/\partial\mathbf{u}$, $v = [v_1, ..., v_n]^T$ and $w = [w_1, ..., w_n]^T$, the angle (i.e., collinearity) between the two columns can be calculated as

$$\Theta = \arccos\left(\frac{\sum_{i=1}^{n} v_i w_i}{\sqrt{\sum_{i=1}^{n} v_i^2 \sum_{i=1}^{n} w_i^2}}\right) \tag{19}$$

From the viewpoint of optimality conditions, a U−U correlation negatively affects the regularity conditions, that is, the Mangasarian−Fromovitz constraint qualification[34] (MFCQ), when the angle describing the collinearity of two columns is close to 0°. Then, the Jacobian matrix of the equality constraints in the NLP tends to be ill-conditioned, and hence, the NLP solver will need more iterations to converge. It is clear that the presence of correlated control variables will also negatively affect the convergence rate when the BFGS method is used, that is, the approximated Hessian might be insufficient and thus cause slow convergence of the optimization algorithm. Therefore, an AH is desired to accelerate the convergence if there is a U−U correlation.

Because an a priori simulation is needed to perform the correlation analysis, proper profiles of control inputs need to be selected. Here, the pseudorandom binary sequence (PRBS) is proposed as a stimulating signal. The PRBS has been widely used for system identification,[35,36] because it exhibits many preferred properties (e.g., persistently exciting, being white-noise-like, and having maximum power for a limited amplitude).[37,38] These properties are also advantageous for the purpose of identifying control-variable correlations, because the input signals should include a broad frequency band and both persistently and sufficiently excite the system within a limited time horizon.

DOI: 10.1021/acs.iecr.5b02369
Ind. Eng. Chem. Res. 2015, 54, 12086−12095

4. PARALLEL COMPUTATION AND SOFTWARE IMPLEMENTATION

Using a personal computer with a limited number of processors, a master processor is defined for the NLP solver, and the remaining possessors are defined as worker possessors each of which executes the computations for an equal number of time intervals. As soon as the NLP solver calls the multiprocessing function, the information will be distributed to the individual worker processors.

Computations for solving model eqs 3 using a Newton method and for calculating both the first-order sensitivities (see eqs 4 and 5) and the second-order sensitivities (see eqs 8, 9, and 11) are performed independently for individual time intervals. Parallel computing is implemented for performing these computations for each time interval using the standard multiprocessing Python module.[39]

It should be noted that the time taken for the communications between the master processor and the worker processors can be significant when too many worker processors are used to solve a small-scale problem. In such a case, the maximum speedup factor can be achieved when fewer worker processors are used.

The proposed solution approach described in section 2 is implemented based on the Modelica[40] and Optimica[41] models. Modelica is an object-oriented modeling environment commonly used for complex industrial systems, and the Optimica extension allows for the formulatation of optimization problems.

In this work, a toolchain for general usage is developed in such a way that the user needs only to define and input the dynamic optimization problem. The transformation from the dynamic optimization problem to an NLP is done completely automatically. Then, using JModelica,[42] the Modelica and Optimica models are transferred to a symbolic representation and made accessible in Python. CasADi[43] is used to perform the discretization of the DAEs, automatic differentiation, and calculation of analytical first- and second-order derivatives. In addition, CasADi is also employed to generate the Jacobian and Hessian and to solve the discretized DAEs. Finally, Ipopt[44] is used to solve the NLP problem, where options of either an AH or a BFGS approach can be chosen by the user.

5. CASE STUDIES

In this section, four dynamic optimization problems are considered as case studies and solved by the proposed approach. The computation times taken using different solution strategies are compared. For a simple presentation, the following abbreviations and symbols are used for all case studies: The number of variables in the NLP and the number of discretized state and algebraic variables are denoted by N_ω and N_ν, respectively. The number of nonzero elements in the Jacobian and Hessian are expressed by χ and ζ, respectively. The time horizon is divided into 60 equidistant time intervals for each problem. The Ipopt default convergence tolerance (10^{-8}) is used in solving the resulting NLP problem. The computation time is given in seconds. The four case studies consist of a continuous stirred-tank reactor (CSTR), a satellite (SAT), a combined-cycle power plant (CCPP), and a distillation column (DIST). The dimensions of these problems are given in Table 1, where the equality constraints correspond to state-variable continuity conditions and the inequality constraints are those imposed on discretized state and algebraic variables in the NLP solver. All computations were performed on up to six Intel I7 4.4 GHz cores with a Linux 64-bit operating system.

Table 1. Problem Dimensions

					constraints	
problem	N_ω	N_ν	χ	ζ	equality	inequality
CSTR	728	1440	6248	4680	488	–
SAT	668	1440	5768	3960	488	–
CCPP	670	24120	112150	3960	610	7210
DIST	7806	202680	6528006	–	7686	43380

5.1. Continuous Stirred-Tank Reactor. This problem was introduced by Luus[45] and further analyzed by Balsa-Canto et al.[18] The purpose of this optimization problem is to determine four control variables of the continuous stirred-tank reactor (CSTR) to maximize the economic benefit. The system dynamics is represented by several simultaneous chemical reactions. The dynamic optimization problem is formulated as[18]

$$\max_{u(t)} x_8(t_f) \tag{20}$$

subject to

$$\dot{x}_1(t) = u_4(t) - q x_1(t) - 17.6 x_1(t)\, x_2(t) - 23 x_1(t)\, x_6(t)\, u_3(t) \tag{21}$$

$$\dot{x}_2(t) = u_1(t) - q x_2(t) - 17.6 x_1(t)\, x_2(t) - 146 x_2(t)\, x_3(t) \tag{22}$$

$$\dot{x}_3(t) = u_2(t) - q x_3(t) - 73 x_2(t)\, x_3(t) \tag{23}$$

$$\dot{x}_4(t) = -q x_4(t) + 35.2 x_1(t)\, x_2(t) - 51.3 x_4(t)\, x_5(t) \tag{24}$$

$$\dot{x}_5(t) = -q x_5(t) + 219 x_2(t)\, x_3(t) - 51.3 x_4(t)\, x_5(t) \tag{25}$$

$$\dot{x}_6(t) = -q x_6(t) + 102 x_4(t)\, x_5(t) - 23 x_1(t)\, x_6(t)\, u_3(t) \tag{26}$$

$$\dot{x}_7(t) = -q x_7(t) + 46 x_1(t)\, x_6(t)\, u_3(t) \tag{27}$$

$$\dot{x}_8(t) = 5.8[q x_1(t) - u_4(t)] - 3.7 u_1(t) - 4.1 u_2(t) + q[23 x_4(t) + 11 x_5(t) + 28 x_6(t) + 35 x_7(t)] - 5 u_3^2(t) - 0.09 \tag{28}$$

$$q = u_1(t) + u_2(t) + u_4(t) \tag{29}$$

$$x(t_0) = [0.1883, 0.2507, 0.0467, 0.0899, 0.1804, 0.1394, 0.1046, 0.0]^T \tag{30}$$

$$0 \le u_1(t) \le 20 \tag{31}$$

$$0 \le u_2(t) \le 6 \tag{32}$$

$$0 \le u_3(t) \le 4 \tag{33}$$

$$0 \le u_4(t) \le 20 \tag{34}$$

$$t_0 \le t \le t_f, \quad t_0 = 0, \quad t_f = 0.2 \tag{35}$$

The Jacobian of the right-hand side $\mathbf{f}$ of model eqs 21−28 with respect to the control variables $\mathbf{u} = [u_1, u_2, u_3, u_4]^T$ is given by

DOI: 10.1021/acs.iecr.5b02369
Ind. Eng. Chem. Res. 2015, 54, 12086−12095

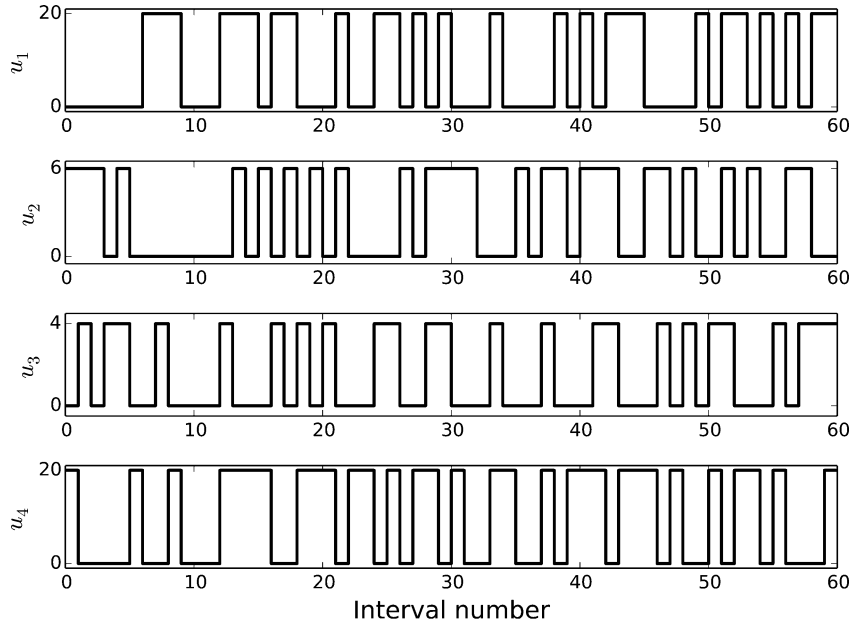

Figure 1. Control profiles generated from PRBSs for the CSTR model.

$$\frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \begin{bmatrix} -x_1(t) & -x_1(t) & -23x_1(t)\,x_6(t) & 1 - x_1(t) \\ 1 - x_2(t) & -x_2(t) & 0 & -x_2(t) \\ -x_3(t) & 1 - x_3(t) & 0 & -x_3(t) \\ -x_4(t) & -x_4(t) & 0 & -x_4(t) \\ -x_5(t) & -x_5(t) & 0 & -x_5(t) \\ -x_6(t) & -x_6(t) & -23x_1(t)\,x_6(t) & -x_6(t) \\ -x_7(t) & -x_7(t) & 46x_1(t)\,x_6(t) & -x_7(t) \\ -3.7 + \xi & -4.1 + \xi & -10u_3(t) & -5.8 + \xi \end{bmatrix} \tag{36}$$

where $\xi = 5.8x_1(t) + 23x_4(t) + 11x_5(t) + 28x_6(t) + 35x_7(t)$. It can be seen that the first, second, and fourth column are nearly linearly dependent. This means that the matrix $\partial\mathbf{f}/\partial\mathbf{u}$ tends to be ill-conditioned. Now, we calculate the angles between these columns using simulation, based on four PRBS signals as control profiles as shown in Figure 1. The calculated angles are given in Table 2, showing that there are strong U−U correlation pairs, namely, u_1−u_2, u_1−u_4, and u_2−u_4.

Table 2. Correlation Values between Controls for the CSTR Model

	u_1	u_2	u_3	u_4
u_1	0.0°	12.1°	123.6°	14.5°
u_2	12.1°	0.0°	123.4°	14.2°
u_3	123.6°	123.4°	0.0°	123.1°
u_4	14.5°	14.2°	123.1°	0.0°

Consequently, it will be favorable to solve this problem using an AH instead of the BFGS method. As predicted from the correlation analysis, the solution of this problem indicates that the analytical Hessian is indeed much more efficient than the BFGS method, as shown in Table 3. It can be seen that the convergence is very slow (1687 NLP iterations) using the BFGS method, whereas a significant improvement (22 NLP iterations) is achieved using the AH. This leads to a significant reduction in computation time. The convergence rate in terms of the objective-function value and the primal and dual infeasibilities (reported by Ipopt) is shown in Figure 2, where only first 100 iterations are plotted for the BFGS method. It can be clearly seen from Figure 2B,C that a feasible solution is difficult to achieve using the BFGS method. Moreover, it is interesting to note from Table 3 that the improvement in computation time by parallel computing (using six processors) is insignificant because the communication time takes a large portion of the total computation time.

Table 3. CSTR Problem Solutions

method	one CPU	six CPUs	objective	iterations
BFGS	112.407	93.119	21.82414	1687
AH	2.651	1.795	21.82414	22

5.2. Satellite Control. Optimal control of a rigid satellite[46,47] initially undergoing a tumbling motion is considered in this case study. The system dynamics is described by seven ordinary differential equations. The optimal control problem is formulated as follows

$$\min_{u(t)} \{\|x(t_f) - x_f\|^2 + 0.5\int_{t_0}^{t_f} \|u\|^2\, dt\} \tag{37}$$

subject to

$$\dot{x}_1 = 0.5(x_5x_4 - x_6x_3 + x_7x_2) \tag{38}$$

$$\dot{x}_2 = 0.5(x_5x_3 + x_6x_4 - x_7x_1) \tag{39}$$

$$\dot{x}_3 = 0.5(-x_5x_2 + x_6x_1 - x_7x_4) \tag{40}$$

$$\dot{x}_4 = -0.5(x_5x_1 + x_6x_2 + x_7x_3) \tag{41}$$

$$\dot{x}_5 = [(I_2 - I_3)x_6x_7 + T_{1s}u_1]I_1^{-1} \tag{42}$$

$$\dot{x}_6 = [(I_3 - I_1)x_7x_5 + T_{2s}u_2]I_2^{-1} \tag{43}$$

$$\dot{x}_7 = [(I_1 - I_2)x_5x_6 + T_{3s}u_3]I_3^{-1} \tag{44}$$

DOI: 10.1021/acs.iecr.5b02369
Ind. Eng. Chem. Res. 2015, 54, 12086−12095

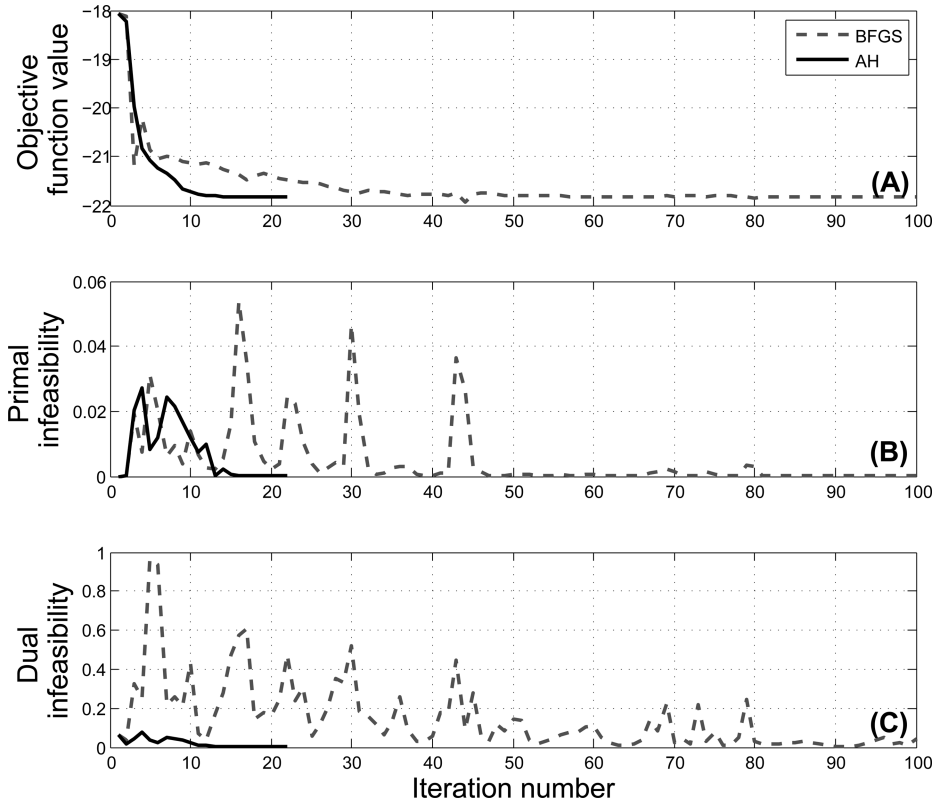

Figure 2. Convergence profiles during solution of the CSTR problem.

$$x(t_0) = [0, 0, 0, 0, 0.01, 0.005, 0.001]^T \quad (45)$$

$$x_f = [0.70106, 0.0923, 0.56098, 0.43047, 0, 0, 0]^T \quad (46)$$

$$t_0 \le t \le t_f, \quad t_0 = 0, \quad t_f = 100 \quad (47)$$

where $I_1 = 10^6$, $I_2 = 833333$, and $I_3 = 916677$ represent the principal moments of inertia and $T_{1s} = 550$, $T_{2s} = 50$, $T_{3s} = 550$ are the corresponding time constants. The Jacobian of the right-hand-side functions of eqs 38−44 with respect to the control variables is represented by

$$\frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ T_{1s}/I_1 & 0 & 0 \\ 0 & T_{2s}/I_2 & 0 \\ 0 & 0 & T_{3s}/I_3 \end{bmatrix} \quad (48)$$

From eq 48, it can be clearly seen that the columns are linearly independent, and thus, the controls are not correlated with each other. The angles between the columns calculated through simulation using PRBS signals as control profiles are given in Table 4, indicating very weak correlations between the controls. Consequently, it is expected that the problem can be efficiently solved using the BFGS method. As indicated in Table 5, although the number of iterations required using the BFGS method is greater, the computation time is less than that required using an AH. This is because the computation cost of the AH is higher than that of the BFGS method. The convergence rate is indicated in Figure 3. It can be seen that, to solve this problem, the convergence rate is almost the same when using either the AH or BFGS method.

Table 4. Correlation Values between Controls for the SAT Model

	u_1	u_2	u_3
u_1	0.0°	98.3°	95.9°
u_2	98.3°	0.0°	102.7°
u_3	95.9°	102.7°	0.0°

Table 5. SAT Problem Solutions

method	one CPU	six CPUs	objective	iterations
BFGS	0.591	0.464	0.46394578	13
AH	0.908	0.617	0.46394578	8

5.3. Combined-Cycle Power Plant. The model of a combined-cycle power plant was taken from in Casella et al.[48] and is available in the JModelica repository. It contains 10 differential states, 124 algebraic variables, and 1 control variable.

A correlation analysis for this problem is not relevant, because only one control variable is involved in the DAE system. Here, we solve this problem by the proposed approach and by a serial algorithm, namely, the collocation method from JModelica. The resulting computation times are shown in Figure 4, where the bold gray line shows the time for the solution of the problem using the collocation method.

It can be seen from Figure 4 that, when using four or six processors, our approach slightly outperforms the serial collocation method using the BFGS method. In addition, the computation time for applying an AH to this problem is higher than for using the BFGS method. This is because the difference between the numbers of NLP iterations required for AH (37

DOI: 10.1021/acs.iecr.5b02369
Ind. Eng. Chem. Res. 2015, 54, 12086−12095

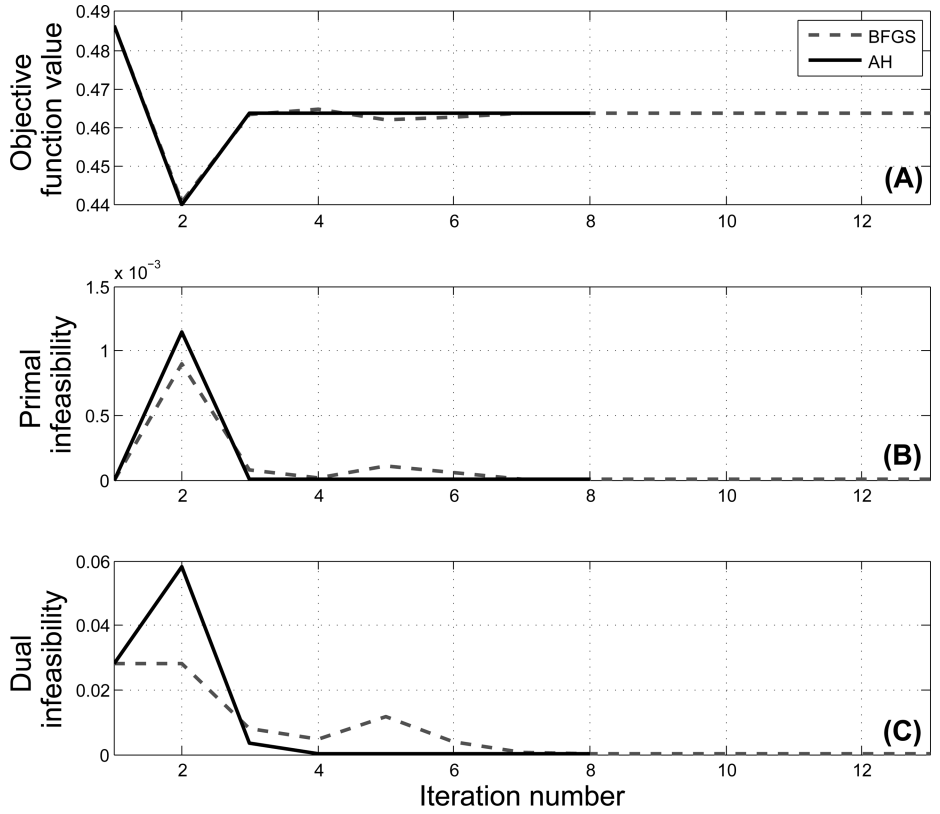

Figure 3. Convergence profiles during solution of the SAT problem.

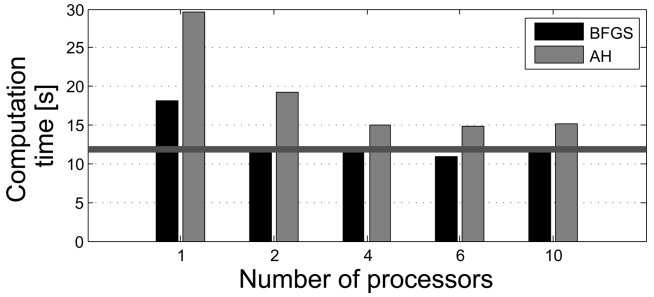

Figure 4. Computation time taken to solve the CCPP problem using parallel computing.

iterations) and for the BFGS method (49 iterations) is not significant. In this case, the computation cost for AH takes a large portion of the total computation time.

It is shown in Figure 4 that, when the number of processors used is increased from one to six, the total computation times for both methods decrease. However, more time is required for 10 processors than for six processors, because of the fast-growing communication time between the master and worker processors.

5.4. Distillation Column. This is a large-scale model available in JModelica. The original model was developed by Diehl,[49] and an extended version was coded in Modelica based on the work of Hedengren.[50] The distillation column has 40 trays for separating a mixture of methanol and *n*-propanol. This DAE model contains 125 differential states (molar vapor flux, temperature, liquid mole fractions for each tray, a reboiler, and a condenser), 1000 algebraic variables, and 2 control signals (volumetric reflux flow and heat input). The objective of this problem is to achieve given set points for temperatures and controls after a short reflux breakdown.

Using an a priori simulation with PRBSs as the control profiles, the results show that the angle between the two control variables is 116.19°. Thus, the two controls are weakly correlated, and therefore, the BFGS method is expected to achieve efficient computation.

We solved this problem again by our approach with the BFGS method and by the serial algorithm in the context of the collocation method available in JModelica. The results are shown in Figure 5, where the bold gray line shows the computation time using the collocation method.

It can be seen from Figure 5A that, as the number of processors for parallel computing is increased, the total

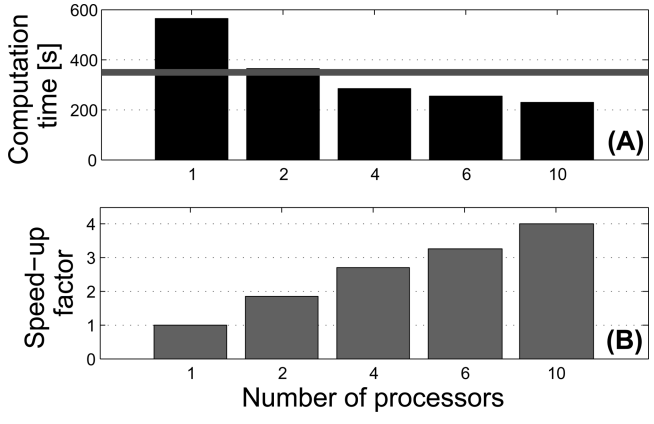

Figure 5. Computation time and speedup factor for solving the DIST problem using parallel computing.

DOI: 10.1021/acs.iecr.5b02369
Ind. Eng. Chem. Res. 2015, 54, 12086−12095

computation time decreases. In comparison to the collocation method, our approach takes less computation time when the number of processors is larger than two. The speedup factor obtained by using parallel computing is shown in Figure 5B, where a factor-of-4 speedup is achieved with 10 processors. Note that this speedup factor represents only the part that can be parallelized, that is, without including the time spent in Ipopt.

However, in the case of using a single processor, the time required for the solution by the proposed approach is much higher, because, unlike for the collocation method, model equations and sensitivities have to be solved in each NLP iteration. Therefore, it can be concluded that the proposed approach is suitable for solving large-scale problems using our parallel-computing strategy.

6. CONCLUSIONS

An efficient solution approach with analytical second-order derivatives and parallel computing based on the CMSC method is proposed. To accelerate the solution process, an AH for the CMSC method is derived. In addition, we proposed a novel approach for correlation analysis of control variables that can be used to identify, a priori, the degree of difficulty for solving dynamic optimization problems. This analysis can easily be done by examining the angles of the columns in the sensitivity matrix through simulation for which we propose to use PRBSs as control signals. A decision on whether to use a BFGS approximation or an AH can be made based on the result of this analysis. Nevertheless, a more detailed theoretical investigation of this method is left for future research.

Parallel computations in terms of time intervals are applied to reduce the computational costs. In comparison to previous algorithms, the parallel computation in this work was performed for the computations in individual time intervals on a stand-alone computer. The developed framework was tested on four dynamic optimization problems including a large-scale problem with more then 200000 variables after discretization. As a future work, the proposed framework will be made available as a web-based toolchain for wider public availability.

AUTHOR INFORMATION

Corresponding Author

*Tel.: +49 3677 69 1423. Fax: +49 3677 69 1434. E-mail: pu.li@tu-ilmenau.de.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

The authors acknowledge support of the ITEA2/EUREKA Cluster program by the European Commission [Project 11004, Model Driven Physical Systems Operation (MODRIO)] and financing from the German Ministry of Education and Research (BMBF, Förderkennzeichen: 01IS12022H). In addition, we thank the anonymous reviewers for their comments, which improved our article.

REFERENCES

(1) Schäfer, A.; Kühl, P.; Diehl, M.; Schlöder, J.; Bock, H. G. Fast reduced multiple-shooting method for nonlinear model predictive control. *Chem. Eng. Process.* **2007**, *46* (11), 1200−1214.

(2) Wang, Y.; Boyd, S. Fast model predictive control using online optimization. *IEEE Trans. Control Syst. Technol.* **2010**, *18* (2), 267−278.

(3) Houska, B.; Ferreau, H. J.; Diehl, M. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica* **2011**, *47* (10), 2279−2285.

(4) Cuthrell, J. E.; Biegler, L. T. Simultaneous optimization and solution methods for batch reactor control profiles. *Comput. Chem. Eng.* **1989**, *13* (1−2), 49−62.

(5) Biegler, L. T. Efficient solution of dynamic optimization and NMPC problems. In *Nonlinear Model Predictive Control*; Allgöwer, F., Zheng, A., Eds.; Progress in Systems and Control Theory Series; Springer Basel AG: Basel, Switzerland, 2000; Vol. *26*, pp 219−243.

(6) Biegler, L. T.; Cervantes, A. M.; Wächter, A. Advances in simultaneous strategies for dynamic process optimization. *Chem. Eng. Sci.* **2002**, *57* (4), 575−593.

(7) Bock, H. G.; Plitt, K. J. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings of the 9th IFAC World Congress*; Pergamon Press: New York, 1984; pp 242−246.

(8) Leineweber, D. B.; Schäfer, A.; Bock, H. G.; Schlöder, J. P. 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.

(9) Kirches, C.; Wirsching, L.; Bock, H. G.; Schlöder, J. P. Efficient direct multiple shooting for nonlinear model predictive control on long horizons. *J. Process Control* **2012**, *22* (3), 540−551.

(10) Leineweber, D. B.; Bauer, I.; Bock, H. G.; Schlöder, J. P. 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.

(11) Vassiliadis, V. S.; Sargent, R. W. H.; Pantelides, C. C. Solution of a class of multistage dynamic optimization problems. 1. Problems without path constraints. *Ind. Eng. Chem. Res.* **1994**, *33* (9), 2111−2122.

(12) Vassiliadis, V. S.; Sargent, R. W. H.; Pantelides, C. C. Solution of a class of multistage dynamic optimization problems. 2. Problems with path constraints. *Ind. Eng. Chem. Res.* **1994**, *33* (9), 2123−2133.

(13) Hong, W. R.; Wang, S. Q.; Li, P.; Wozny, G.; Biegler, L. T. A quasi-sequential approach to large-scale dynamic optimization problems. *AIChE J.* **2006**, *52* (1), 255−268.

(14) Bartl, M.; Li, P.; Biegler, L. T. Improvement of State profile accuracy in nonlinear dynamic optimization with the quasi-sequential approach. *AIChE J.* **2011**, *57* (8), 2185−2197.

(15) Tamimi, J.; Li, P. A combined approach to nonlinear model predictive control of fast systems. *J. Process Control* **2010**, *20* (9), 1092−1102.

(16) Lazutkin, E.; Geletu, A.; Hopfgarten, S.; Li, P. Modified multiple shooting combined with collocation method in JModelica.org with symbolic calculations. In *Proceedings of the 10th International Modelica Conference*; Linköping University Press: Linköping, Sweden, 2014; pp 999−1006.

(17) Vassiliadis, V. S.; Balsa-Canto, E.; Banga, J. R. Second-order sensitivities of general dynamic systems with application to optimal control problems. *Chem. Eng. Sci.* **1999**, *54* (17), 3851−3860.

(18) Balsa-Canto, E.; Banga, J. R.; Alonso, A. A.; Vassiliadis, V. S. Dynamic optimization of chemical and biochemical processes using restricted second-order information. *Comput. Chem. Eng.* **2001**, *25* (4−6), 539−546.

(19) Balsa-Canto, E.; Banga, J. R.; Alonso, A. A.; Vassiliadis, V. S. Dynamic optimization of distributed parameter systems using second-order directional derivatives. *Ind. Eng. Chem. Res.* **2004**, *43* (21), 6756−6765.

(20) Barz, T.; Kuntsche, S.; Wozny, G.; Arellano-Garcia, H. An efficient sparse approach to sensitivity generation for large-scale dynamic optimization. *Comput. Chem. Eng.* **2011**, *35* (10), 2053−2065.

(21) Barz, T.; Kraus, R.; Zhu, L.; Wozny, G.; Arellano-Garcia, H. Generation of discrete first- and second-order sensitivities for single shooting. *AIChE J.* **2012**, *58* (10), 3110−3122.

DOI: 10.1021/acs.iecr.5b02369
Ind. Eng. Chem. Res. 2015, 54, 12086−12095

(22) Li, S.; Petzold, L.; Zhu, W. Sensitivity analysis of differential-algebraic equations: a comparison of methods on a special problem. *Appl. Numer. Math.* **2000**, *32*, 161−174.
(23) Özyurt, D. B.; Barton, P. I. Large-scale dynamic optimization with the directional second order adjoint method. *Ind. Eng. Chem. Res.* **2005**, *44* (6), 1804−1811.
(24) Hartwich, A.; Stockmann, C.; Terboven, Ch.; Feuerriegel, S.; Marquardt, W. Parallel sensitivity analysis for efficient large-scale dynamic optimization. *Optim. Eng.* **2011**, *12* (4), 489−508.
(25) Word, D. P.; Kang, J.; Åkesson, J.; Laird, C. D. Efficient parallel solution of large-scale nonlinear dynamic optimization problems. *Comput. Optim. Appl.* **2014**, *59* (3), 667−688.
(26) Kang, J.; Cao, Y.; Word, D. P.; Laird, C. D. An interior-point method for efficient solution of block-structured NLP problems using an implicit Schur-complement decomposition. *Comput. Chem. Eng.* **2014**, *71*, 563−573.
(27) Zavala, V. M.; Laird, C. D.; Biegler, L. T. Interior-point decomposition approaches for parallel solution of large-scale nonlinear parameter estimation problems. *Chem. Eng. Sci.* **2008**, *63* (19), 4834−4845.
(28) Washington, I. D.; Swartz, C. L. E. Design under uncertainty using parallel multiperiod dynamic optimization. *AIChE J.* **2014**, *60* (9), 3151−3168.
(29) Hindmarsh, A. C.; Brown, P. N.; Grant, K. E.; Lee, S. L.; Serban, R.; Shumaker, D. E.; Woodward, C. S. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Trans. Math. Software* **2005**, *31* (3), 363−396.
(30) Bachmann, B.; Ochel, L.; Ruge, V.; Gebremedhin, M.; Fritzson, P.; Nezhadali, V.; Eriksson, L.; Sivertsson, M. Parallel Multiple-Shooting and Collocation Optimization with OpenModelica. In *Proceedings of the 9th International Modelica Conference*; Linköping University Press: Linköping, Sweden, 2012; pp 659−668.
(31) Chu, Y.; Hahn, J. Parameter set selection for estimation of nonlinear dynamic systems. *AIChE J.* **2007**, *53* (11), 2858−2870.
(32) McLean, K. A. P.; McAuley, K. B. Mathematical modelling of chemical processes obtaining the best model predictions and parameter estimates using identifiability and estimability procedures. *Can. J. Chem. Eng.* **2012**, *90* (2), 351−366.
(33) Li, P.; Vu, Q. D. Identification of parameter correlations for parameter estimation in dynamic biological models. *BMC Syst. Biol.* **2013**, *7*, 91.
(34) Nocedal, J.; Wright, S. J. *Numerical Optimization*; Springer Series in Operations Research and Financial Engineering; Springer: New York, 2006.
(35) Ljung, L. *System Identification: Theory for the User*; Prentice Hall PTR: Upper Saddle River, NJ, 1999.
(36) Haber, R.; Unbehauen, H. Structure identification of nonlinear dynamic systems - A survey on input/output approaches. *Automatica* **1990**, *26* (4), 651−677.
(37) MacWilliams, F. J.; Sloane, N. J. A. Pseudo-random sequences and arrays. *Proc. IEEE* **1976**, *64* (12), 1715−1729.
(38) Sarwate, D. V.; Pursley, M. B. Crosscorrelation properties of pseudo-random and related sequences. *Proc. IEEE* **1980**, *68* (5), 593−619.
(39) Hellmann, D. *The Python Standard Library by Example*; Addison-Wesley Professional: Boston, 2011.
(40) Fritzson, P. *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*, 2nd ed.; Wiley-IEEE Press: New York, 2014.
(41) Åkesson, J. Optimica—An extension of Modelica supporting dynamic optimization. In *Proceeding of the 6th International Modelica Conference*; Linköping University Press: Linköping, Sweden, 2008; pp 57−66.
(42) Åkesson, J.; Årzén, K.-E.; Gåfvert, M.; Bergdahl, T.; Tummescheit, H. Modeling and optimization with Optimica and JModelica.org-languages and tools for solving large-scale dynamic optimization problems. *Comput. Chem. Eng.* **2010**, *34* (11), 1737−1749.
(43) Andersson, J.; Åkesson, J.; Diehl, M. CasADi: A symbolic package for automatic differentiation and optimal control. *Lect. Notes Comput. Sci. Eng.* **2012**, *87*, 297−307.
(44) Wächter, A.; Biegler, L. T. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Math. Prog.* **2006**, *106* (1), 25−57.
(45) Luus, R. Application of dynamic programming to high-dimensional non-linear optimal control problems. *Int. J. Control* **1990**, *52* (1), 239−250.
(46) Junkins, J. L.; Turner, J. D. Optimal continuous torque attitude maneuvers. *J. Guid. Control* **1980**, *3* (3), 210−217.
(47) Robinett, R. D., III; Wilson, D. G.; Eisler, G. R.; Hurtado, J. E. *Applied Dynamic Programming for Optimization of Dynamical Systems. Advances in Design and Control*; SIAM: Philadelphia, 2005.
(48) Casella, F.; Donida, F.; Åkesson, J. Object-Oriented Modeling and Optimal Control: A Case Study in Power Plant Start-Up. In *Proceedings of the 18th IFAC World Congress*; International Federation of Automatic Control: Laxenburg, Austria, 2011; pp 9545−9554.
(49) Diehl, M. Real-Time Optimization for Large Scale Nonlinear Processes. Ph.D. Thesis, Universität Heidelberg, Heidelberg, Germany, 2001.
(50) Hedengren, J. D. A nonlinear model library for dynamics and control. In *CACHE News*; CACHE Corporation: Austin, TX, 2008.

DOI: 10.1021/acs.iecr.5b02369
Ind. Eng. Chem. Res. 2015, 54, 12086−12095