MODELING OF NLP PROBLEMS OF CHEMICAL PROCESSES DESCRIBED BY ODEs

Tvrzská de Gouvêa, M. [1] **and Odloak, D.**[2]

[1]*Universidade Presbiteriana Mackenzie, Department of Materials Engineering*
Rua da Consolação 896, 01302-907, São Paulo-SP, Brazil; miriamtg_br@yahoo.com
[2]*Universidade de São Paulo, Department of Chemical Engineering*
odloak@usp.br

Abstract: Both real-time and off-line optimizations are commonly performed in order to enhance productivity. The optimization problem is often posed as a non-linear programming (NLP) problem solved by a SQP algorithm. When processes need to be described by differential equations, difficulties will arise in using SQP algorithms, since Jacobians of constraints described by differential equations will have to be evaluated. In this paper, we show how to derive analytical expressions for both Jacobian and Hessian matrices for the constraints described by ordinary differential equations, without increasing the dimension of the resultant NLP problem to be solved. *Copyright © 2006 IFAC*

Keywords: differential equations, nonlinear programming, process models, optimization problems.

1. INTRODUCTION

Globalization has lead to the necessity of optimally operating the chemical plants. Thus, not only is it necessary to adequately control the chemical processes, but, moreover, optimal operating conditions must be continuously forecast and implemented, which may be achieved by solving a real-time optimization (RTO) problem. As far as continuous processes described by concentrated parameters models are under regard, optimal operating policies can be obtained by solving nonlinear programming (NLP) problems, possessing constraints described solely by algebraic equations. SQP algorithms may effectively solve NLP problems and successful implementations of real time optimization strategies are well known (Zanin et al., 2002, Jakhete et al., 1999, Ellis et al., 1998, Agrawal et al., 1996). When it comes to optimize continuous processes described by distributed models or semi-batch and batch processes, one difficulty arises when the corresponding optimization problem is to be solved by a SQP algorithm. The latter will require that Jacobians or even a Hessian matrix be evaluated for the constraints represented by differential equations. One common approach to override the difficulties arisen from the mathematical description of the process by differential equations is to discretize the latter (Cuthrell and Biegler, 1989), i.e., the continuous state variables are transformed into several discrete variables. This approach leads to an increase in the dimension of the NLP problem to be solved and to the loss of information due to the discretization performed. In this paper, we aim to present a general procedure to analytically model Jacobians and Hessian matrices of the constraints described by ordinary differential equations (ODEs). The resulting model for evaluation of the Jacobians and Hessians is composed by ordinary differential equations that are coupled to the differential equations describing the process model. Thus, evaluation of the Jacobians and Hessians may be obtained with the same numerical precision as the solution of the process model and without any loss of information.

In section 2, we briefly review the available different SQP algorithms emphasizing the need for the evaluation of Jacobians and even Hessians. In section 3 we present guidelines on how to write the optimization problem. In sections 4 and 5, the models for calculating Jacobians and Hessians are presented. In section 6, we apply the proposed procedure to the optimization problem of a batch reactor. The paper is finally concluded in section 7.

2. GENERAL STEPS OF SQP ALGORITHMS

Given a NLP problem as in (1), the SQP algorithms produce a sequence of values as in (2), where the search direction d_k is the solution of the QP problem (3) and α_k is so as to guarantee $f(x_{k+1}) < f(x_k)$. Hence, the general idea of the SQP algorithms can be summarized in algorithm A_o.

$$f(x^*) = \min_{x \in S \cap B_\delta(x^*)} f(x) \qquad (1)$$

where,

$S = \{x \in R^n : h(x) = 0; g(x) \le 0\}$,

$f : R^n \to R; h : R^n \to R^m; g : R^n \to R^p$, $B_\delta(x^*)$ is an open–ball of radius δ with center in x^*.

$$\{x_k\} \to x^* : x_{k+1} = x_k + \alpha_k d_k \qquad (2)$$

$$\min_{d_k \in S_l} \frac{1}{2} d_k^T H_k d_k + \nabla f^T(x_k) d_k \qquad (3)$$

where, H_k is either equal to $\nabla^2 L(x_k, \lambda_k, \mu_k)$ or an approximation to it and $\nabla^2 L(x_k, \lambda_k, \mu_k)$ is the Hessian evaluated at the point x_k, λ_k, μ_k of the Lagrangian function associated with the NLP defined as in (4), S_l is the set of constraints, which is either chosen as in (5) or (6).

$$L(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \mu^T g(x) \qquad (4)$$

where, λ and μ are the Lagrange multipliers of the equality and inequality constraints of the NLP.

$$S_l = \begin{Bmatrix} d_k \in R^n : \nabla h^T(x_k) d_k = -h(x_k); \\ \nabla g^T(x_k) d_k \le -g(x_k) \end{Bmatrix} \qquad (5)$$

$$S_l = \begin{Bmatrix} d_k \in \Delta_k : \nabla h^T(x_k) d_k = -h(x_k); \\ \nabla g^T(x_k) d_k \le -g(x_k) \end{Bmatrix} \qquad (6)$$

where, Δ_k is the trust region where the linear approximation S_l of S is expected to hold well.

Algorithm A_o

```
1. k = 0;  $x_k = x_o$
2. Solve the QP problem to obtain $d_k$
3. Obtain $\alpha_k$ so that $f(x_k + \alpha_k d_k) < f(x_k)$
4. $x_{k+1} = x_k + \alpha_k d_k$ and check the optimality condition on $x_{k+1}$. If it is satisfied stop otherwise $k = k + 1$ and return to 2.
```

Differences in the SQP algorithms are related to whether analytical expressions are provided for the Hessian matrix H_k or if it is estimated from the Jacobian matrix of the constraints and enforced to be positive definite, to whether (5) or (6) are chosen as the constraints of the QP problem and in what manner α_k is calculated (Tvrzská de Gouvêa and Odloak, 1998, Ternet and Biegler, 1998, Lucia et al., 1996, Bartholomew-Biggs and Hernandez, 1995, Schmid and Biegler, 1994). Different implementations affect convergence properties and robustness of the SQP algorithm. Analytical expressions for both the Jacobians and the Hessian matrix may make the SQP algorithms achieve a quadratic convergence rate and by properly managing the nonconvex QP problems obtained with analytical expressions of H_k, robustness of the SQP algorithms may be increased (Tvrzská de Gouvêa and Odloak, 1998). So it may be desired to have available not only analytical expressions for $\nabla h^T(x_k)$ and $\nabla g^T(x_k)$, but also for H_k. As far as the constraints in (1) are solely described by algebraic equations, difficulties in establishing analytical expressions for the Jacobians and Hessian matrix are restricted to the difficulties in obtaining derivatives. When there are constraints described by ODEs, analytical expressions are not readily available and so the common practice (Cuthrell and Biegler, 1989) is to simply discretize the differential equations. By doing so, important process information may be lost and the SQP algorithm will have its convergence rate deteriorated. If discretization is not adequately performed, numerical instabilities may also occur. So it may be advantageous to have analytical expressions for the Jacobian and Hessian matrices of the constraints described by ODEs. In sections 4 and 5 we show how to derive differential equations that analytically describe the Jacobians and Hessian matrix for processes described by ODEs, which enables one to use SQP algorithms that explicitly deal with nonconvex QP subproblems.

3. THE GENERAL OPTIMIZATION PROBLEM OF PROCESSES DESCRIBED BY ODEs

Equation (7) generally describes optimization problems of processes described by ODEs. The economical objective function (f) is modeled by an algebraic equation. It may correspond to the batch time or to the heat consumption or to operational costs or any other desired economical specification. So it will typically depend on the initial conditions of the process (x_o), on the operational time or on the length of the equipment, both of these latter variables denoted by t, on state variables x^e, on the degrees of freedom of the process given by the manipulated variables (u) and on any other process variables z. In the formulation presented in equation (7), x^e corresponds to state values calculated by the SQP algorithm, i.e., these values must be equal to the solutions x of the differential equations that describe the process model given in equation (8). The general analytical solution $x(x_o,u,t)$ of (8) is not known, just its numerical one that must equal to x^e. In the formulation of the NLP problem presented in (7), the equality constraints h were written in an algebraic form and were divided into two groups. By means of the first group of equality constraints h_1, the state variables are to be evaluated. Though h_1 is written as an algebraic equation, x is actually the numerical solution of a system of ODEs. Therefore, for the evaluation of the Jacobian matrix of the constraints h_1, derivatives in terms of x_o, u and t must be evaluated. In constraints h_2, just algebraic equations are considered. Note that these constraints are written in terms of the state variables evaluated by the SQP algorithm (x^e) and not in terms of the state variables evaluated by the solution of the ODEs (x). This is a subtle way to eliminate the dependence on the independent variable t of all variables not explicitly appearing in the differential term of the ODEs. Also note that x^e may contain values for a same physical variable evaluated at different values of t. For example, if bound constraints are to be made on the temperature along a tubular reactor, different values for the state variable temperature must be available for different positions. Inequality constraints are presented in g. The remaining constraints correspond to bound constraints on the decision variables of the NLP problem described in (1). As to the dimension of the NLP problem (1), it is assumed that: $z \in R^{n_z}$; $x^e, x_o \in R^{n_x}$, $u \in R^{n_u}$, $t \in R$, $f: R^{n_z+2n_x+n_u+1} \to R$; $h_1: R^{2n_x+n_u+1} \to R^{n_x}$; $h_2: R^{n_z+2n_x+n_u+1} \to R^{m-n_x}$; $g: R^{n_z+2n_x+n_u+1} \to R^p$.

$$\begin{aligned}
&\min_{z,x^e,x_o,u,t} f(z,x^e,x_o,u,t)\\
&s.t.\quad h_1(x^e,x_o,u,t) = -x^e + x(x_o,u,t) = 0\\
&\qquad h_2(z,x^e,x_o,u,t) = 0\\
&\qquad g(z,x^e,x_o,u,t) \le 0\\
&\qquad z_{\min} \le z \le z_{\max}\\
&\qquad x^e_{\min} \le x^e \le x^e_{\max}\\
&\qquad x^o_{\min} \le x^o \le x^o_{\max}\\
&\qquad u_{\min} \le u \le u_{\max}\\
&\qquad 0 \le t \le t_{\max}
\end{aligned} \tag{7}$$

$$\dot{x} = \bar{h}(x_o,u,t) \tag{8}$$

Since f, h_2 and g are algebraic equations, the evaluation of their derivatives with respect to the decision variables is straightforward and well known. So, no further comments will be made. Equation (7) can be discretized by either finite differences or orthogonal collocation methods and put in an algebraic form as in (9), where now x^e are estimates for the state variables taken at nd discretization points. Thus, instead of n_x state variables, one will now have $n_x \times nd$ state variables and the dimension of the NLP problem (7) will be significantly increased, which will affect the performance of the SQP algorithms. Not only there will be the need to discretize the state variables, but if the manipulated variables are dependent on t, they will also have to be discretized, leading to another increase in the dimension of the SQP. At the same time, loss of information is inevitable. So it is expected that the number of iterations for convergence to an optimal solution will augment as well as the computational cost of each SQP iteration.

$$h_1(x^e,x_o,u,t) = 0 \tag{9}$$

So there is the interest in not performing any discretization and derive analytical expressions for both Jacobain and Hessian matrices of the constraints in the partition h_1. In order to facilitate the derivation of analytical expressions for the Jacobians and Hessians of h_1, it is convenient that some further assumptions be taken on the way the equations are written in (7), which are:

- All manipulated variables are assumed to be independent.
- The initial conditions are independent from the manipulated variables u.

With those assumptions, simple ordinary differential equations may be derived by means of which, the Jacobians and Hessians of the constraints h_1 may be evaluated, as will be shown in the next two sections. The procedure will not increase the size of the NLP, will not result in any loss of process information and allows the manipulated variables to have a continuous or discrete dependence on t. Since analytical expressions are made available by the described procedure, SQP algorithms may be chosen in order to deal with nonconvex QPs and thus better convergence properties may be expected. There are some drawbacks, though. The analytical derivation of the expressions shown in sections 4 and 5 may be complex and the number of differential equations needed to be solved in order to evaluate both the Jacobians and Hessians may be large if either n_x or n_u are too large. Fortunately, chemical processes typically have small number of degrees of freedom and so n_u will be restricted to a small number.

4. EVALUATION OF THE JACOBIAN MATRIX

Since the constraints h_1 as defined in equation (7) do not depend on z and depend linearly on x^e, and taking (8) into account, the Jacobian matrix ∇h_1^T of the constraints h_1 will have the general structure given in equation (10).

$$\left[\begin{array}{c|c|ccc}
 & & \frac{\partial x_1}{\partial x_{o,1}} \cdots \frac{\partial x_1}{\partial x_{o,n_x}} & \frac{\partial x_1}{\partial u_1} \cdots \frac{\partial x_1}{\partial u_{n_u}} & \bar{h}_1(x_o,u,t) \\
0 & -I & \vdots & \ddots & \vdots \\
 & & \frac{\partial x_{n_x}}{\partial x_{o,1}} \cdots \frac{\partial x_{n_x}}{\partial x_{o,n_x}} & \frac{\partial x_{n_x}}{\partial u_1} \cdots \frac{\partial x_{n_x}}{\partial u_{n_u}} & \bar{h}_{n_x}(x_o,u,t)
\end{array}\right] \tag{10}$$

Let $y_{1,i,j} = \frac{\partial x_i}{\partial x_{o,j}}; j = 1 \ldots n_x; i = 1 \ldots n_x$ and $y_{2,i,j} = \frac{\partial x_i}{\partial u_j}$; $j = 1 \ldots n_u; i = 1 \ldots n_x$. So, for evaluating (10) at any point (x_o, u, t), one has to obtain $y_{1,i,j}$ and $y_{2,i,j}$. Since, n_x differential equations in t will be solved in (8), the idea is to augment the number of differential equations in time and by means of the added ODEs, each $y_{1,i,j}$ and $y_{2,i,j}$ are to be evaluated. This may be done by adequately applying the chain rule as is shown in equation (11) for the evaluation of $y_{1,i,j}$.

$$\frac{\partial y_{1,i,j}}{\partial t} = \frac{\partial}{\partial t}\left(\frac{\partial x_i}{\partial x_{o,j}}\right) = \frac{\partial}{\partial x_{o,j}}\left(\frac{\partial x_i}{\partial t}\right) \tag{11}$$

Since, from (8) $\frac{dx_i}{dt} = \bar{h}_i(x_o,u,t)$, equation (11) becomes equation (12).

$$\frac{\partial y_{1,i,j}}{\partial t} = \frac{\partial \bar{h}_i}{\partial x_{o,j}} + \sum_{\substack{k=1\\k\neq j}}^{n_x} \frac{\partial \bar{h}_i}{\partial x_{o,k}}\frac{\partial x_{o,k}}{\partial x_{o,j}} + \sum_{k=1}^{n_x} \frac{\partial \bar{h}_i}{\partial x_k}\frac{\partial x_k}{\partial x_{o,j}} + \sum_{k=1}^{n_u} \frac{\partial \bar{h}_i}{\partial u_k}\frac{\partial u_k}{\partial x_{o,j}} + \frac{\partial \bar{h}_i}{\partial t}\frac{\partial t}{\partial x_{o,j}} \quad (12)$$

Since t is an independent variable, u and x_o were assumed independent one from another as well as the initial conditions are also taken independently, equation (12) can be reduced to (13).

$$\frac{\partial y_{1,i,j}}{\partial t} = \frac{\partial \bar{h}_i}{\partial x_{o,j}} + \sum_{k=1}^{n_x} \frac{\partial \bar{h}_i}{\partial x_k} y_{1,k,j} \quad (13)$$

A similar procedure is now applied to $y_{2,i,j}$, as is shown in equations (14) and (15).

$$\frac{\partial}{\partial t} y_{2,i,j} = \frac{\partial}{\partial t}\left(\frac{\partial x_i}{\partial u_j}\right) = \frac{\partial}{\partial u_j}\left(\frac{\partial x_i}{\partial t}\right) \quad (14)$$

$$\frac{\partial}{\partial t} y_{2,i,j} = \frac{\partial \bar{h}_i}{\partial u_j} + \sum_{k=1}^{n_x} \frac{\partial \bar{h}_i}{\partial x_k}\frac{\partial x_k}{\partial u_j} + \sum_{\substack{k=1\\k\neq j}}^{n_u} \frac{\partial \bar{h}_i}{\partial u_k}\frac{\partial u_k}{\partial u_j} + \sum_{k=1}^{n_x} \frac{\partial \bar{h}_i}{\partial x_{o,k}}\frac{\partial x_{o,k}}{\partial u_j} + \frac{\partial \bar{h}_i}{\partial t}\frac{\partial t}{\partial u_j} \quad (15)$$

Since u_j and u_k were also assumed to be independent one from another, equation (15) can be reduced to (16).

$$\frac{\partial y_{2,i,j}}{\partial t} = \frac{\partial \bar{h}_i}{\partial u_j} + \sum_{k=1}^{n_x} \frac{\partial \bar{h}_i}{\partial x_k} y_{2,k,j} \quad (16)$$

By adding equations (15) and (16) to the set of differential equations (8), one can simultaneously obtain the state variables x and the Jacobian matrix. It is noteworthy to note that $2n_x n_u$ differential equations are needed to evaluate the Jacobian matrix. Since the number of degrees of freedom is usually not very large, the number of differential equations will not be too large.

5. EVALUATION OF THE HESSIAN MATRIX OF THE CONSTRAINTS

Equations (17) to (22) show the general block structure of the Hessian matrix associated with the i^{th} constraint of h_1. Recall that the Hessian matrix $\nabla^2 h_{1,i}$ is symmetric and so only the non-symmetrical elements are shown in (18) to (22). The first $n_x + n_z$ rows and columns of $\nabla^2 h_{1,i}$ will be composed of null vectors since h_1 does not depend on z and depends linearly on x^e.

$$\nabla^2 h_{1,i} = \left[\begin{array}{c|ccc} 0 & & 0 & \\ \hline 0 & B_1 & B_3 & B_4 \\ & B_2^T & B_2 & B_5 \\ & B_4^T & B_5^T & \frac{\partial^2 x_i}{\partial t^2} \end{array}\right] \quad (17)$$

$$B_1 = \begin{bmatrix} \frac{\partial^2 x_i}{\partial x_{o,1}^2} & \cdots & \frac{\partial^2 x_i}{\partial x_{o,n_x}\partial x_{o,1}} \\ & \ddots & \vdots \\ & & \frac{\partial^2 x_i}{\partial x_{o,n_x}^2} \end{bmatrix} \quad (18)$$

$$B_2 = \begin{bmatrix} \frac{\partial^2 x_i}{\partial u_1^2} & \cdots & \frac{\partial^2 x_i}{\partial u_{n_u}\partial u_1} \\ & \ddots & \vdots \\ & & \frac{\partial^2 x_i}{\partial u_{n_u}^2} \end{bmatrix} \quad (19)$$

$$B_3 = \begin{bmatrix} \frac{\partial^2 x_i}{\partial u_1 \partial x_{o,1}} & \cdots & \frac{\partial^2 x_i}{\partial u_{n_u}\partial x_{o,1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 x_i}{\partial u_1 \partial x_{o,n_x}} & \cdots & \frac{\partial^2 x_i}{\partial u_{n_u}\partial x_{o,n_x}} \end{bmatrix} \quad (20)$$

$$B_4^T = \begin{bmatrix} \frac{\partial^2 x_i}{\partial t \partial x_{o,1}} & \cdots & \frac{\partial^2 x_i}{\partial t \partial x_{o,n_x}} \end{bmatrix} \quad (21)$$

$$B_5^T = \begin{bmatrix} \frac{\partial^2 x_i}{\partial t \partial u_1} & \cdots & \frac{\partial^2 x_i}{\partial t \partial u_{n_u}} \end{bmatrix} \quad (22)$$

As performed in section 4, the idea is again to evaluate several elements of the Hessian matrix, by differentiating them in time and appropriately apply the chain rule. The last column of the Hessian matrix (17) may be easily obtained and we will start with its characterization, which is done in equations (23) to (25).

$$\frac{\partial^2 x_i}{\partial t \partial x_{o,j}} = \frac{\partial}{\partial t} y_{1,i,j} = \frac{\partial \bar{h}_i}{\partial x_{o,j}} + \sum_{k=1}^{n_x} \frac{\partial \bar{h}_i}{\partial x_k} y_{1,k,j} \quad (23)$$

$$\frac{\partial^2 x_i}{\partial t \partial u_j} = \frac{\partial}{\partial t} y_{2,i,j} = \frac{\partial \bar{h}_i}{\partial u_j} + \sum_{k=1}^{n_x} \frac{\partial \bar{h}_i}{\partial x_k} y_{2,k,j} \quad (24)$$

$$\frac{\partial^2 x_i}{\partial t^2} = \frac{\partial \bar{h}_i}{\partial t} + \sum_{k=1}^{n_x} \frac{\partial \bar{h}_i}{\partial x_k}\frac{\partial x_k}{\partial t} + \sum_{k=1}^{n_u} \frac{\partial \bar{h}_i}{\partial u_k}\frac{\partial u_k}{\partial t} + \sum_{i=1}^{n_x} \frac{\partial \bar{h}_i}{\partial x_{o,k}}\frac{\partial x_{o,k}}{\partial t} \quad (25)$$

Since, the initial conditions are constants and by taking (8) into account, equation (25) is reduced to (26).

$$\frac{\partial^2 x_i}{\partial t^2} = \frac{\partial \bar{h}_i}{\partial t} + \sum_{k=1}^{n_x} \frac{\partial \bar{h}_i}{\partial x_k}\bar{h}_k(x_o,u,t) + \sum_{k=1}^{n_u} \frac{\partial \bar{h}_i}{\partial u_k}\frac{\partial u_k}{\partial t} \quad (26)$$

Now, we will show how to evaluate each term in blocks B_1 to B_3. For that purpose, let for each

$i=1,\ldots,n_x$: $\quad y_{B_1,i,k,j}=\dfrac{\partial^2 x_i}{\partial x_{o,k}\partial x_{o,j}}$, $\quad k=1,\ldots,n_x$, $j=1,\ldots,n_x$, $\quad y_{B_2,i,k,j}=\dfrac{\partial^2 x_i}{\partial u_k\partial x_{o,j}}$, $\quad k=1,\ldots,n_u$, $j=1,\ldots,n_x$ and $y_{B_3,i,k,j}=\dfrac{\partial^2 x_i}{\partial u_k\partial u_j}$, $\quad k=1,\ldots,n_u$, $j=1,\ldots,n_u$. Each term of matrix B_1 can be evaluated by integrating $y_{B_1,i,k,j}$ in time accordingly to equation (29). Similarly, each term of matrices B_2 and B_3 are obtained by integrating $y_{B_2,i,k,j}$ and $y_{B_3,i,k,j}$ in time accordingly to equations (32) and (35).

Equation (29) is obtained by applying the chain rule to the derivative in time of $y_{B_1,i,k,j}$ as shown in (27) and by performing further simplifications as described next. Similar procedures are applied to $y_{B_2,i,k,j}$ and $y_{B_3,i,k,j}$, which is shown in the development that follows.

$$\begin{aligned}\frac{\partial}{\partial t}y_{B_1,i,k,j}&=\frac{\partial}{\partial t}\left(\frac{\partial}{\partial x_{o,k}}\left(\frac{\partial x_i}{\partial x_{o,j}}\right)\right)=\\&=\frac{\partial}{\partial x_{o,k}}\left(\frac{\partial}{\partial t}\left(\frac{\partial x_i}{\partial x_{o,j}}\right)\right)=\frac{\partial}{\partial x_{o,k}}\left(\frac{\partial}{\partial t}y_{1,i,j}\right)\end{aligned}\tag{27}$$

Thus, differentiating equation (13) in respect to $x_{o,k}$, one obtains:

$$\frac{\partial y_{B_1,i,k,j}}{\partial t}=\frac{\partial^2\overline{h}_i}{\partial x_{o,k}\partial x_{o,j}}+\sum_{p=1}^{n_x}\left(\frac{\partial^2\overline{h}_i}{\partial x_{o,k}\partial x_p}y_{1,p,j}+\frac{\partial\overline{h}_i}{\partial x_p}\frac{\partial y_{1,p,j}}{\partial x_{o,k}}\right)\tag{28}$$

Hence, $y_{B_1,i,k,j}$ is obtained by solving (29).

$$\frac{\partial y_{B_1,i,k,j}}{\partial t}=\frac{\partial^2\overline{h}_i}{\partial x_{o,k}\partial x_{o,j}}+\sum_{p=1}^{n_x}\left(\frac{\partial^2\overline{h}_i}{\partial x_{o,k}\partial x_p}y_{1,p,j}+\frac{\partial\overline{h}_i}{\partial x_p}y_{B_1,p,k,j}\right)\tag{29}$$

In a similar way we perform with $y_{B_2,i,k,j}$, as shown in the equations that follow.

$$\begin{aligned}\frac{\partial}{\partial t}y_{B_2,i,k,j}&=\frac{\partial}{\partial t}\left(\frac{\partial}{\partial u_k}\left(\frac{\partial x_i}{\partial x_{o,j}}\right)\right)=\\&=\frac{\partial}{\partial u_k}\left(\frac{\partial}{\partial t}\left(\frac{\partial x_i}{\partial x_{o,j}}\right)\right)=\frac{\partial}{\partial u_k}\left(\frac{\partial}{\partial t}y_{1,i,j}\right)\end{aligned}\tag{30}$$

$$\frac{\partial}{\partial t}y_{B_2,i,k,j}=\frac{\partial^2\overline{h}_i}{\partial u_k\partial x_{o,j}}+\sum_{p=1}^{n_x}\left(\frac{\partial^2\overline{h}_i}{\partial u_k\partial x_p}y_{1,p,j}+\frac{\partial\overline{h}_i}{\partial x_p}\frac{\partial y_{1,p,j}}{\partial u_k}\right)\tag{31}$$

$$\frac{\partial}{\partial t}y_{B_2,i,k,j}=\frac{\partial^2\overline{h}_i}{\partial u_k\partial x_{o,j}}+\sum_{p=1}^{n_x}\left(\frac{\partial^2\overline{h}_i}{\partial u_k\partial x_p}y_{1,p,j}+\frac{\partial\overline{h}_i}{\partial x_p}y_{4,p,k,j}\right)\tag{32}$$

Similarly we perform with $y_{B_3,i,k,j}$, as presented next.

$$\begin{aligned}\frac{\partial y_{B_3,i,k,j}}{\partial t}&=\frac{\partial}{\partial t}\left(\frac{\partial}{\partial u_k}\left(\frac{\partial x_i}{\partial u_j}\right)\right)=\\&=\frac{\partial}{\partial u_k}\left(\frac{\partial}{\partial t}\left(\frac{\partial x_i}{\partial u_j}\right)\right)=\frac{\partial}{\partial u_k}\left(\frac{\partial}{\partial t}y_{2,i,j}\right)\end{aligned}\tag{33}$$

$$\frac{\partial y_{B_3,i,k,j}}{\partial t}=\frac{\partial^2\overline{h}_i}{\partial u_k\partial u_j}+\sum_{p=1}^{n_x}\left(\frac{\partial^2\overline{h}_i}{\partial u_k\partial x_p}y_{2,i,j}+\frac{\partial\overline{h}_i}{\partial x_p}y_{3,p,k,j}\right)\tag{34}$$

By adding equations (29), (32) and (34) to the set of differential equations formed by (8), (15) and (16), one can simultaneously obtain the state variables *x*, the Jacobian and Hessian matrices of the constraints. In order to calculate the Hessian matrices for each one of the n_x constraints, one will need to solve $\dfrac{n_x(n_x+3)+n_u(n_u+3)}{2}+n_u n_x+1$ differential equations, which indeed may become a large number if the number of state variables is large. It is also noteworthy to say that Hessian matrices related to real processes typically have a sparse structure and so several terms will be actually zero, which will also reduce the number of differential equations that will be solved.

6. APPLICATION TO A BATCH REACTOR

Because of the lack of space, in this section just a brief outline of the application of the proposed formulation to an academic simple problem will be presented. A thorough description of the application of the procedure to a problem of industrial interest will be presented elsewhere (Souza et al., 2006). Equation (35) corresponds to the optimization problem of minimizing the batch time of a reactor (t_f), where a first order reaction is taking place and the concentration of specimen A must be kept below a limit value. The optimal temperature profile ($T(t)$) and the initial concentration ($c_{A,o}$) are to be evaluated.

$$\begin{aligned}&\min_{c_{A,o},c_A^E,T(t),t_f}\ t_f\\&s.t.\ -c_A^E+c_A(t_f)=0\\&\quad 0\le c_A^E\le c_{\max}\\&\quad 0\le T(t)\le T^{\max}\\&\quad t_f\ge 0\end{aligned}\tag{35}$$

where, $c_A(t_f)$ is the concentration of A at the end of the batch and is obtained by solving (36).

$$\frac{dc_A}{dt}=-kc_A\ ;k=k_o e^{-\frac{E_R}{T(t)}}\tag{36}$$

For any instant of time *t*, the Jacobian and the Hessian matrices of the equality constraint are given in equations (37) and (38) and the values for y_1 to y_5 are obtained for any instant *t* from the integration of equations (39) to (43). As for the initial conditions, from the definitions presented in sections 4 and 5, it follows that y_1=1 and y_2 to y_5 equal to 0. So one will have to integrate equations (36) together with equations (39) to (43) to obtain the Jacobian and Hessian matrices as in (37) and (38).

$$\nabla h^T = \begin{bmatrix} -1 & y_1 & y_2 & -kc_A \end{bmatrix} \tag{37}$$

$$\nabla^2 h = \left[\begin{array}{c|ccc} 0 & & 0 & \\ \hline & y_5 & y_4 & -ky_1 \\ 0 & y_4 & y_3 & -kc_A \dfrac{E_R}{T^2} - ky_2 \\ & -ky_1 & -kc_A \dfrac{E_R}{T^2} - ky_2 & k^2 c_A \end{array}\right] \tag{38}$$

$$\frac{dy_1}{dt} = -ky_1 \tag{39}$$

$$\frac{dy_2}{dt} = -kc_A \frac{E_R}{T^2} - ky_2 \tag{40}$$

$$\frac{dy_3}{dt} = -kc_A \left[\left(\frac{E_R}{T^2} \right)^2 - 2\frac{E_R}{T^3} \right] - 2k\frac{E_R}{T^2} y_2 - ky_3 \tag{41}$$

$$\frac{dy_4}{dt} = -y_1 k \frac{E_R}{T^2} - ky_4 \tag{42}$$

$$\frac{dy_5}{dt} = -ky_5 \tag{43}$$

The above model describes an isothermal operation of the reactor, for which equation (36) has a trivial solution given by $c_A = c_{A,o} e^{-kt}$ and hence equations (37) and (38) must equal to equations (44) and (45). Table 1 gives the comparison of the Jacobian and Hessian matrices evaluated by equations (37) and (38) and by equations (44) and (45) with $c_{A,o}$=1000 mol/m^3, T=523 K, k_o=3370 s^{-1}, E_R=7000 K^{-1} and t_f=360 s.

$$\nabla h^T = \begin{bmatrix} -1 & e^{-kt} & -c_A tk \dfrac{E}{T^2} & -kc_A \end{bmatrix} \tag{44}$$

$$\nabla^2 h = \left[\begin{array}{c|ccc} 0 & & 0 & \\ \hline & 0 & -kt\dfrac{E}{T^2} e^{-kt} & -ke^{-kt} \\ 0 & \begin{matrix} = el. \\ (2,3) \end{matrix} & c_A kt \dfrac{E}{T^2} \left(\dfrac{2}{T} - \dfrac{E}{T^2} + kt\dfrac{E}{T^2} \right) & \begin{matrix} = el. \\ (4,3) \end{matrix} \\ & \begin{matrix} = el. \\ (2,4) \end{matrix} & c_A k \dfrac{E}{T^2} (kt - 1) & k^2 c_A \end{array}\right] \tag{45}$$

Table 1: Comparison of the elements of the Jacobian and Hessian matrices

element	matrix	evaluated by (37) or (38)	evaluated by (44) or (45)	error (%)
(1,2)	∇h^T	0.15470	0.15474	0.03
(1,3)	∇h^T	-7.390	-7.389	0.01
(2,2)	$\nabla^2 h$	0	0	0
(2,3)	$\nabla^2 h$	-0.0073896	-0.0073895	0.001
(2,4)	$\nabla^2 h$	-8.0209e-4	-8.0208e-4	0.001
(3,3)	$\nabla^2 h$	0.19208	0.19203	0.03
(3,4)	$\nabla^2 h$	0.01779	0.01778	0.06

7. CONCLUSIONS

A general procedure was shown on how to develop analytical expressions for the evaluation of Jacobian and Hessian matrices for constraints described by ODEs. The developed expressions are also composed of ODEs and are obtained simultaneously with the integration of the process model.

ACKNOWLEDGMENT

Financial support from MACKPESQUISA under grant 1176, from FAPESP under grant 97/07963-4 and from CNPq under grant 303304/2004-9 is gratefully acknowledged.

REFERENCES

Agrawal, S.S., Beach, W., Rendon, G. T., Olvera, R. O. (1996) Implementation of advanced on-line control, optimization and planning systems in Mexican refineries. In: NPRA Computer Conference. Atlanta, USA, November, 11-13

Bartholomew_Biggs, M.C. and Hernandez, F.G. (1995) Using the KKT matrix in an augmented Lagrangian SQP method for sparse constrained optimization. *J.O.T.A.*, **85**(1), 201-220

Cuthrell, J.E. and Biegler, L.T. (1989) Simultaneous optimization and solution methods for batch reactor control profiles. *Computers Chem. Engineering*, **13**(1-2), 49-62

Ellis, R.C., Xuan, L.; Riggs, J.B. (1998) Modeling and optimization of a model IV fluid catalytic cracking unit. *AIChE Journal*, **44**, 2068-2079

Lucia, A., Xu, J., Layn, K.M. (1996) Nonconvex process optimization. *Computers Chem. Engng.*, **20**(2), 1375-1398.

Jakhete, R.; Rager, W.; Hoffman, D.W. (1999) Online implementation of composite LP optimizers FCCU/GPU complex. *Hydrocarbon processing*. **78**(2), 69

Schmid, C. and Biegler, L.T. (1994) Quadratic programming methods for reduced Hessian SQP. *Computers Chem. Engng.*, **18**(9), 817-832

Souza, G. F, Gonçalves, A. P. C., Odloak, D., Tvrzská de Gouvêa, M. Optimization of the operation of a reactor for the production of ethylene oxide/ working paper/ 2006

Ternet, D.J. and Biegler, L.T. (1998) Recent improvements to a multiplier-free reduced Hessian successive quadratic programming algorithm. *Computers Chem. Engng.*, **22**(7-8), 963-978

Tvrzská de Gouvêa, M. and Odloak, D. (1998) A new treatment of inconsistent quadratic programs in a SQP-based algorithm. *Computers Chem. Engng.*, **22**(11), 1623-1651

Zanin, A.C., Tvrzská de Gouvêa, M., Odloak, D. (2002) Integrating real-time optimization into the model predictive controller of the FCC system. *Control Engineering Practice*, **10**, 819-831