Proceedings of the 18th World Congress
The International Federation of Automatic Control
Milano (Italy) August 28 - September 2, 2011

Adjoint-based distributed multiple shooting for large-scale systems

Carlo Savorgnan [*] Attila Kozma [*] Joel Andersson [*]
Moritz Diehl [*]

[*] *Electrical Engineering Department (ESAT-SCD) and Optimization in Engineering Center (OPTEC), K.U.Leuven, Kasteelpark Arenberg 10, bus 2446, 3001 Leuven-Heverlee, Belgium (e-mail: {carlo.savorgnan, attila.kozma, joel.andersson, moritz.diehl}@esat.kuleuven.be).*

Abstract: Distributed multiple shooting is a modification of the standard multiple shooting method which takes into account the structure of certain large-scale systems in order to obtain a better controller design flexibility and high parallelizability. The aim of this paper is to extend the framework where distributed multiple shooting can be deployed and to propose a new solution method based on adjoint-based sequential quadratic programming. A numerical experiment shows that this can lead to considerable savings in computational time for the sensitivity generation.

Keywords: Large-scale systems, optimal control, nonlinear systems, numerical methods, optimization.

1. INTRODUCTION

The control of networked systems has captured a lot of attention in the scientific literature in the past decades (see the books (Mesarovic et al., 1970; Findeisen et al., 1980)). More recently, many results have been devised in the field of hierarchical and distributed model predictive control ((Camponogara et al., 2002; Magni and Scattolini, 2006; Venkat, 2006; Richards and How, 2007; Rawlings and Stewart, 2008; Scattolini, 2009)). An element that is shared by most of these works is that they make use of local controllers. The main motivations behind this choice are that the large optimization problems to be solved in order to find a controller are hard to solve in a centralized way and that local controllers are easier to maintain. In (Laird and Biegler, 2008; Zhu et al., 2009) it has been shown that by exploiting the problem structure large optimization problems can be solved efficiently by modified standard numerical methods.

In (Savorgnan et al., 2011) distributed multiple shooting (DMS) has been introduced for a class of large systems composed by several interconnected subsystems. This method is a generalization of standard multiple shooting (MS) (Bock and Plitt, 1984). MS divides the control horizon into intervals on which the inputs are parametrized and simulations are performed. This leads to a finite dimensional nonlinear programming problem (NLP) which is usually solved employing sequential quadratic programming (SQP). A careful profiling of a software implementation of MS shows that a large amount of the computational time is spent in function evaluations and generation of sensitivities. DMS tries to overcome this problem by representing the quantities that couple the subsystem dynamics in a finite functional basis. This discretization allows to obtain an algorithm where function evaluation and generation of sensitivities are highly parallelizable. The main drawback of DMS is that one needs to choose carefully the dimension of the basis used to represent the functional space. A large number of basis functions leads to a high accuracy of the solution while a small number reduces the size of the NLP.

Contribution. In this paper we extend the DMS framework to deal with problems with coupling cost and constraints. Furthermore we propose a numerical method to overcome the main shortcoming of DMS. We show that using an adjoint-based SQP method (see (Griewank and Walther, 2002; Bock et al., 2007; Diehl et al., 2010)) one can use a large number of basis functions while keeping the computational burden per SQP iteration low.

The paper is organized as follows. In Section 2 the problem class considered is presented. Section 3 illustrates distributed multiple shooting, while Section 4 explains how adjoint-based SQP methods can be applied to DMS. Section 5 show some numerical experiments to validate the numerical method proposed.

We use the following notation. $\mathbb{R}$ represents the set of real numbers. Superscript ^{i} indicates the subsystem the variable or function belongs to. When v is a vector in $\mathbb{R}^n$, v_k indicates the k-th component of v. The superscript ^{T} indicates transposition. When $f(\cdot)$ is a function depending on x, f_x represents the Jacobian. g_{xx} represents the Hessian of the scalar function $g(x)$.

2. PROBLEM FORMULATION

In this paper we consider systems which can be decomposed into M coupled subsystems whose behavior is characterized by the following equations

$$\begin{cases} \dot{x}^i(t) = f^i(x^i(t), u^i(t), z^i(t)) \\ y^i(t) = g^i(x^i(t), u^i(t), z^i(t)) \end{cases} \quad i = 1, \ldots, M \qquad (1)$$

The vectors $x^i(t) \in \mathbb{R}^{n_x^i}$, $u^i(t) \in \mathbb{R}^{n_u^i}$, $z^i(t) \in \mathbb{R}^{n_z^i}$ and $y^i(t) \in \mathbb{R}^{n_y^i}$ represent the state, the control input, the coupling input and the output of subsystem i, respectively. We discriminate between control and coupling inputs because they have a different role in our setting. The former can be used to control the system, while the latter takes into account dynamic couplings between the subsystems.

We assume the couplings between the subsystems can be expressed in the following form

$$z^i(t) = \sum_{j=1}^{M} A_{ij} y^j(t) \tag{2}$$

where the matrices A_{ij} are used to express the connections. We assume that $A_{jj} = 0$ and that the couplings between the systems are sparse. This results in many zero matrices A_{ij} and the presence of many zeros in the non-zero matrices A_{ij}. An example of an interconnected system which is included in our framework is depicted in Figure 1.

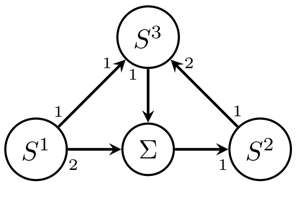

Fig. 1. A system composed by $M = 3$ subsystems. The coupling between the subsystems are given by $z_1^3(t) = y_1^1(t)$, $z_2^3(t) = y_1^2(t)$ and $z_1^2(t) = y_2^1(t) + y_1^3(t)$.

In this paper we consider the solution of optimal control problems of the form

$$\begin{aligned}
\min_{\substack{x,u,z,\\ y,e}} \quad & \int_0^T \ell(e(t))dt + \sum_{i=1}^{M} \int_0^T \ell^i(x^i(t), u^i(t), z^i(t))dt \\
\text{s.t.} \quad & \dot{x}^i(t) = f^i(x^i(t), u^i(t), z^i(t)) \\
& y^i(t) = g^i(x^i(t), u^i(t), z^i(t)) \\
& x^i(0) = \bar{x}_0^i \\
& z^i(t) = \sum_{j=1}^{M} A_{ij} y^j(t) \\
& e(t) = r(t) + \sum_{i=1}^{M} B^i y^i(t) \\
& p^i(x^i(t), u^i(t)) \geq 0, \quad q(e(t)) \geq 0 \quad t \in [0, T]
\end{aligned} \tag{3}$$

where the terminal time T and the initial condition $\bar{x}_0^i$ for the subsystems are fixed. The vector $e(t) \in \mathbb{R}^{n_e}$ is used as an auxiliary variable to formulate a coupling objective and coupling constraints which were not considered in (Savorgnan et al., 2011). The vector $r(t) \in \mathbb{R}^{n_e}$ is a reference signal. The functions p^i and q define local and coupling constraints.

We assume that (3) is well defined and has an optimal solution.

Remark 1. Although the functions ℓ, f, g, p, and q do not depend explicitly on time, the technique we present can be trivially extended to this case by introducing dummy states.

3. THE DISTRIBUTED MULTIPLE SHOOTING METHOD

In this section we show how distributed multiple shooting (DMS) can be adapted to solve Problem (3). The numerical solution of this infinite dimensional problem is addressed in two steps: in the first steps the problem is discretized in order to obtain a finite dimensional nonlinear programming problem (NLP) that is then solved using sequential quadratic programming (SQP).

3.1 Control input discretization

Define the time points

$$0 = t_0 < t_1 < \cdots < t_N = T. \tag{4}$$

which divide the interval $[0, T]$ into N sub-intervals. The inputs are discretized using a finite parametrization on the time sub-intervals. For simplicity, in this paper we will use a piece-wise constant parametrization:

$$u^i(t) = u_n^i \quad \forall t \in [t_n, t_{n+1}), \quad n = 0, \ldots, N-1 \tag{5}$$

and

$$u^i(t_N) = u_N^i. \tag{6}$$

3.2 Coupling discretization

To be able to integrate the dynamics of the subsystems independently, the vectors $z^i(t)$, $e(t)$, $r(t)$ and $y^i(t)$ are represented as a linear combination of a function basis. Although other choices are possible, we will restrict our attention to a basis composed by normalized Legendre polynomials $\gamma_q(t)$, which are defined by the relation

$$\int_{-1}^{1} \gamma_q(t)\gamma_m(t)dt = \begin{cases} 0 & \text{if } q \neq m \\ 1 & \text{otherwise} \end{cases}. \tag{7}$$

Define the vector of normalized Legendre polynomials up to degree S

$$\Gamma(t) = [\gamma_0(t)\ \gamma_1(t)\ \gamma_2(t)\ \ldots\ \gamma_S(t)]^T. \tag{8}$$

For $t \in [t_n, t_{n+1})$ we introduce the following approximation

$$z_p^i(t) = \Gamma_n(t)^T \mathbf{z}_{n,p}^i \tag{9}$$

where $\mathbf{z}_{n,p}^i \in \mathbb{R}^{S+1}$ is a coefficient vector and

$$\Gamma_n(t) = \Gamma\left(2\frac{t - t_n}{t_{n+1} - t_n} - 1\right). \tag{10}$$

Notice that in order to avoid using new letters in our notation, we use the a boldface font $(\mathbf{z}_{n,p}^i)$ to represent the coefficient vector corresponding to time varying quantities $(z^i(t))$. We will use this convention in the remainder of the paper.

For notational convenience we define the matrix

$$\mathbf{z}_n^i = \left[\mathbf{z}_{n,1}^i\ \ldots\ \mathbf{z}_{n,n_z^i}^i\right] \tag{11}$$

such that $z^i(t) = \left(\mathbf{z}_n^i\right)^T \Gamma_n(t)$. Using the same approximation used for $z^i(t)$, we represent $e(t)$ and $r(t)$ as a linear combination of the basis $\Gamma_n(t)$. Therefore, for $t \in [t_n, t_{n+1})$

$$e(t) = \mathbf{e}_n^T \Gamma_n(t) \quad \text{and} \quad r(t) = \mathbf{r}_n^T \Gamma_n(t). \tag{12}$$

To find the coefficient matrix $\mathbf{y}_n^i$ used to represent $y^i(t)$, we have to solve the following unconstrained quadratic optimization problem

$$\mathbf{y}_{n,q}^i = \arg\min_{\mathbf{y} \in \mathbb{R}^{S+1}} \int_{t_n}^{t_{n+1}} \left(\Gamma_n(t)^T \mathbf{y} - y_q^i(t)\right)^2 dt. \tag{13}$$

Due to the orthogonality of $\Gamma_n(t)$, (13) can be solved performing the following integration

$$\mathbf{y}_n^i = \frac{2}{t_{n+1} - t_n} \int_{t_n}^{t_{n+1}} \Gamma_n(t) \left(y^i(t)\right)^T dt. \tag{14}$$

3.3 State discretization by distributed multiple shooting

Denote by x_n^i the value of $x^i(t_n)$ [1]. Given x_n^i, u_n^i, $\mathbf{z}_n^i$ we can simulate the i-th subsystem on the n-th interval. Using the corresponding solution $x_n^i(t)$, $t \in [t_n, t_{n+1}]$, define the functions F_n^i, L_n^i and G_n^i as

$$F_n^i(x_n^i, u_n^i, \mathbf{z}_n^i) := x_n^i\left(t_{n+1}\right), \tag{15}$$

$$L_n^i(x_n^i, u_n^i, \mathbf{z}_n^i) := \int_{t_n}^{t_{n+1}} \ell^i(x_n^i(t), u_n^i(t), \Gamma(t)^T \mathbf{z}_n^i) dt \tag{16}$$

and

$$G_n^i(x_n^i, u_n^i, \mathbf{z}_n^i) := \frac{2}{t_{n+1} - t_n} \int_{t_n}^{t_{n+1}} \Gamma_n(t) \left(y_n^i(t)\right)^T dt. \tag{17}$$

It should be understood that F_n^i, L_n^i and G_n^i are defined up to the approximation error introduced by the coupling input discretization.

3.4 NLP formulation

Distributed multiple shooting solves the following NLP

$$\begin{aligned} \min_{\substack{u_n^i, x_n^i, \mathbf{z}_n^i,\\ \mathbf{y}_n^i, \mathbf{e}_n}} \quad & \sum_{n=0}^{N-1} \left(L_n(\mathbf{e}_n) + \sum_{i=1}^{M} L_n^i(x_n^i, u_n^i, \mathbf{z}_n^i) \right) \\ \text{s.t.} \quad & x_{n+1}^i = F_n^i(x_n^i, u_n^i, \mathbf{z}_n^i) \qquad n = 0, \ldots, N-1 \\ & \mathbf{y}_n^i = G_n^i(x_n^i, u_n^i, \mathbf{z}_n^i) \qquad n = 0, \ldots, N-1 \\ & x_0^i = \bar{x}_0^i \\ & \mathbf{z}_n^i = \sum_{j=1}^{M} A_{ij} \mathbf{y}_n^j \\ & \mathbf{e}_n = \mathbf{r}_n + \sum_{j=1}^{M} B_{ij} \mathbf{y}_n^j \\ & p^i(x_n^i, u_n^i) \geq 0, \qquad Q_n(\mathbf{e}_n) \geq 0 \end{aligned} \tag{18}$$

where the functions Q_n and L_n are defined as

$$Q_n(\mathbf{e}_n) = q(\mathbf{e}_n^T \Gamma_n(t_n)).$$

and

$$L_n(\mathbf{e}_n) = \int_{t_n}^{t_{n+1}} \ell(\mathbf{e}_n^T \Gamma_n(t)) dt.$$

As in standard multiple shooting, Problem (18) can be solved using sequential quadratic programming. This method iteratively solves a local quadratic approximation of the the original NLP. To employ SQP one needs to evaluate several times the functions which define the system and compute first and possibly second order sensitivities. Since they require the integration of ordinary differential equations, the evaluation of the functions L_n^i, F_n^i and G_n^i together with the generation of the corresponding sensitivities can be very time consuming (see (Bauer et al., 2000; Petzold et al., 2006; Albersmeyer and Bock, 2008) for a description of the techniques which can be used to generate sensitivities).

When solving Problem (18) by standard SQP, one has to decide the value of S, i.e. the maximum degree of the basis $\Gamma(t)$. If S is high, the representation of $z(t)$, $y(t)$, $e(t)$ and $r(t)$, and consequently the solution of the optimal control problem, will be more accurate. However, increasing S, also increases the number of decision variables and therefore the number of sensitivities evaluated at every SQP iteration, which could make the computational time considerably larger.

4. THE ADJOINT SCHEME FOR THE NLP SOLUTION

In this section we propose a method which allows to use a large value of S while keeping the computational load due to sensitivity generation low. Therefore, the method is able to find an accurate optimal solution to the control problem. The only drawback, as we will see later, is that the local convergence to a solution is only linear. This method is based on the following assumption.

Assumption 1. The high order components of $z^i(t)$ have a small influence on the system dynamics and output.

This assumption is reasonable when the output can be well approximated by a polynomial decomposition of order smaller than S.

Divide the vector Γ in two parts

$$\tilde{\Gamma}(t) = [\gamma_0(t) \ \ldots \ \gamma_T(t)]^T \tag{19}$$

$$\hat{\Gamma}(t) = [\gamma_{T+1}(t) \ \ldots \ \gamma_S(t)]^T \tag{20}$$

and define the coefficient matrices $\tilde{\mathbf{z}}_n$ and $\hat{\mathbf{z}}_n$ such that

$$\left(\mathbf{z}_n\right)^T \Gamma_n(t) = \left(\tilde{\mathbf{z}}_n\right)^T \tilde{\Gamma}_n(t) + \left(\hat{\mathbf{z}}_n\right)^T \hat{\Gamma}_n(t). \tag{21}$$

When Assumption 1 is satisfied, the values of F_n^i and G_n^i are weakly influenced by $\hat{\mathbf{z}}_n^i$. Equivalently, Assumption 1 implies that

$$\frac{\partial F_n^i(x_n^i, u_n^i, \mathbf{z}_n^i)}{\partial \hat{\mathbf{z}}_n^i} \approx 0 \quad \text{and} \quad \frac{\partial G_n^i(x_n^i, u_n^i, \mathbf{z}_n^i)}{\partial \hat{\mathbf{z}}_n^i} \approx 0. \tag{22}$$

In order to reduce the computation related to sensitivity generation, we propose to neglect most of the derivatives with respect to $\hat{\mathbf{z}}_n^i$ since they are anyway close to zero. Instead we will only evaluate one exact adjoint gradient of the Lagrangian. The algorithm we propose is based on adjoint-based SQP ((Bock et al., 2007; Diehl et al., 2010)).

To avoid an overcomplicated notation we will continue our explanation using only the vectors $\hat{z}$ and x which are defined as follows: the former contains all the variables in $\hat{\mathbf{z}}_n^i$ for all i and n, while the latter contains all the other decision variables appearing in Problem (18) (i.e. u_n^i, x_n^i, $\tilde{\mathbf{z}}_n^i$, $\mathbf{y}_n^i$, $\mathbf{e}_n$).

Problem (18) can be rearranged in the following form

$$\begin{aligned} \min_{x, \hat{z}} \quad & l(x, \hat{z}) \\ & \hat{z} - Ax = 0 \\ & g(x, \hat{z}) = 0 \\ & p(x) \geq 0 \end{aligned} \tag{23}$$

where l can be derived from the functions L_n^i and the matrix A is defined using the coefficients in A_{ij}. The

[1] To keep the notation and the exposition as simple as possible we use the same intervals for the discretization of the states and the inputs.

functions g and p are used to express all the remaining equality and inequality constraints. Assumption 1 implies that the Jacobian of g w.r.t. $\hat{z}$ is close to zero, i.e. $g_{\hat{z}} \approx 0$.

To introduce adjoint-based SQP we first consider a simplified setting where the inequality constraint $p(x) \geq 0$ is neglected. Associate to the first equality constraint the vector of multipliers μ and to the second constraint the vector λ. The Lagrangian and KKT system for the simplified problem read

$$\mathcal{L}(x, \hat{z}, \lambda, \mu) = l(x, \hat{z}) - \mu^T(\hat{z} - Ax) - \lambda^T g(x, \hat{z}) \tag{24}$$

and

$$\begin{bmatrix} \mathcal{L}_x^T(x, \hat{z}, \mu, \lambda) \\ \mathcal{L}_{\hat{z}}^T(x, \hat{z}, \mu, \lambda) \\ \hat{z} - Ax \\ g(x, \hat{z}) \end{bmatrix} = 0, \tag{25}$$

respectively. The system of equalities (25) can be solved by the Newton method defined by the iteration

$$\begin{bmatrix} \mathcal{L}_{xx} & \mathcal{L}_{x\hat{z}} & A^T & -g_x^T \\ \mathcal{L}_{\hat{z}x} & \mathcal{L}_{\hat{z}\hat{z}} & -I & -g_{\hat{z}}^T \\ -A & I & 0 & 0 \\ g_x & g_{\hat{z}} & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{k+1} - x_k \\ \hat{z}_{k+1} - \hat{z}_k \\ \mu_{k+1} - \mu_k \\ \lambda_{k+1} - \lambda_k \end{bmatrix} = \begin{bmatrix} -\mathcal{L}_x \\ -\mathcal{L}_{\hat{z}} \\ -\hat{z}_k + Ax_k \\ -g(x_k, \hat{z}_k) \end{bmatrix} \tag{26}$$

where the Jacobians and Hessians are evaluated in $(x_k, \hat{z}_k, \mu_k, \lambda_k)$.

Newton's method is known to have local quadratic convergence to a solution when the so called KKT matrix on the l.h.s. and the vector on the r.h.s. of (26) are evaluated exactly. When the computation of the matrix on the l.h.s. is too expensive, we can use an approximation. If the approximation is good enough the iteration will still converge to an optimal solution, but with a linear rate (Deuflhard, 2006).

To solve Problem (23) (and therefore (18)) we propose to use the following approximation of the KKT matrix where $g_{\hat{z}}$ is set to zero, i.e. many derivatives do not need to be computed

$$\begin{bmatrix} \mathcal{L}_{xx} & \mathcal{L}_{x\hat{z}} & A^T & -g_x^T \\ \mathcal{L}_{\hat{z}x} & \mathcal{L}_{\hat{z}\hat{z}} & -I & -g_{\hat{z}}^T \\ -A & I & 0 & 0 \\ g_x & g_{\hat{z}} & 0 & 0 \end{bmatrix} \approx \begin{bmatrix} L_{xx} & L_{x\hat{z}} & A^T & -g_x^T \\ L_{\hat{z}x} & L_{\hat{z}\hat{z}} & -I & 0 \\ -A & I & 0 & 0 \\ g_x & 0 & 0 & 0 \end{bmatrix}. \tag{27}$$

The matrices L are a suitably chosen approximation of the Hessian term (depending on the problem at hand the approximation can be obtained with the BFGS method, Gauss-Newton, or other techniques).

To obtain convergence to an exact solution, the r.h.s. of equation (26) must be evaluated exactly. This requires the evaluation of

$$\mathcal{L}_{\hat{z}} = l_{\hat{z}} - I\mu - g_{\hat{z}}^T \lambda. \tag{28}$$

Notice that to evaluate the term $g_{\hat{z}}^T \lambda$ we don't need to evaluate the full Jacobian $g_{\hat{z}}$. Instead $g_{\hat{z}}^T \lambda$ can be cheaply evaluated using adjoint derivatives. The usage of adjoint derivatives gives the name to the method.

Define the following quantities

$$\begin{aligned} \delta x_k &= x_{k+1} - x_k, & \delta \hat{z}_k &= \hat{z}_{k+1} - \hat{z}_k, \\ \delta \mu_k &= \mu_{k+1} - \mu_k, & \delta \hat{\lambda}_k &= \hat{\lambda}_{k+1} - \hat{\lambda}_k. \end{aligned} \tag{29}$$

The solution of the iterations of (26) with the approximation (27) can be also interpreted as the solution of the following quadratic problem

$$\begin{aligned} \min_{\delta x_k, \delta \hat{z}_k} \quad & \frac{1}{2} \begin{bmatrix} \delta x_k \\ \delta \hat{z}_k \end{bmatrix}^T \begin{bmatrix} L_{xx} & L_{x\hat{z}} \\ L_{\hat{z}x} & L_{\hat{z}\hat{z}} \end{bmatrix} \begin{bmatrix} \delta x_k \\ \delta \hat{z}_k \end{bmatrix} + [\mathcal{L}_x \mathcal{L}_{\hat{z}}] \begin{bmatrix} \delta x_k \\ \delta \hat{z}_k \end{bmatrix} \\ & \delta \hat{z}_k + \hat{z}_k - A(\delta x_k + x_k) = 0 \\ & g_x^T \delta x_k + g(x_k, \hat{z}_k) = 0 \end{aligned} \tag{30}$$

where the values of $\delta\mu_k$ and $\delta\lambda_k$ are the Lagrange multipliers corresponding to the first and second constraint, respectively. This fact can be used when we want to introduce the inequality constraint $p(x) \geq 0$. In this case the modified QP problem we have to solve is just

$$\begin{aligned} \min_{\delta x_k, \delta \hat{z}_k} \quad & \frac{1}{2} \begin{bmatrix} \delta x_k \\ \delta \hat{z}_k \end{bmatrix}^T \begin{bmatrix} L_{xx} & L_{x\hat{z}} \\ L_{\hat{z}x} & L_{\hat{z}\hat{z}} \end{bmatrix} \begin{bmatrix} \delta x_k \\ \delta \hat{z}_k \end{bmatrix} + [\mathcal{L}_x \ \mathcal{L}_{\hat{z}}] \begin{bmatrix} \delta x \\ \delta \hat{z} \end{bmatrix} \\ & \delta \hat{z}_k + \hat{z}_k - A(\delta x_k + x_k) = 0 \\ & g_x^T \delta x_k + g(x_k, \hat{z}_k) = 0 \\ & p_x^T \delta x_k + p(x_k) \geq 0 \end{aligned} . \tag{31}$$

It can be shown that when the optimal active set has been found during the SQP iteration the method converges linearly to the solution (Diehl et al., 2010).

Remark 2. The size of Problem (31) can be reduced by eliminating the variable $\delta\hat{z}$ exploiting the equation

$$\delta \hat{z}_k = -\hat{z}_k + A(\delta x_k + x_k). \tag{32}$$

5. NUMERICAL EXPERIMENTS

In this section we show some timing tests for sensitivity generation applied to the benchmark control problem which is described in detail in (Savorgnan and Diehl, 2010). The system under consideration is a hydro power plant composed by a river and 3 lakes. Figure 2 gives an overview of the system. The river is divided into 6 reaches

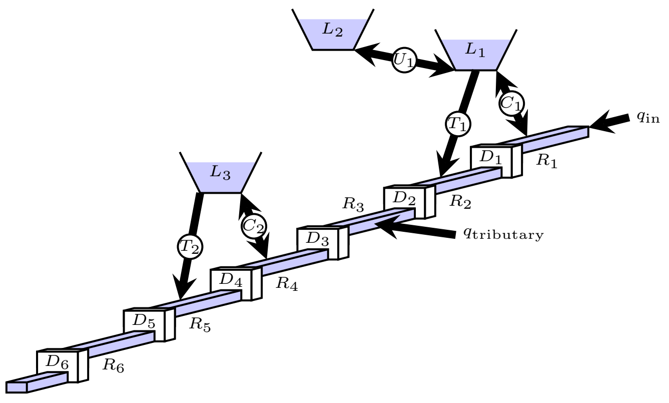

Fig. 2. Overview of the hydro power plant.

which terminate with dams equipped with turbines for power production. We regard every reach together with the following dam as a separate subsystem ($R_1 + D_1$, $R_2 + D_2$, $R_3 + D_3$, $R_4 + D_4$, $R_5 + D_5$, $R_6 + D_6$). Lakes L_1 and L_2 are connected by a duct (U_1) and lake L_1 is connected to the river by duct with a turbine (T_1) and a duct with a turbine and a pump (C_1). Lakes L_1 and L_2 form together with U_1, C_1 and T_1 another subsystem. Lake L_3 is connected to the river by a duct with a turbine (T_2) and a duct with a turbine and a pump (C_2). L_3, C_2 and T_2 compose the last subsytem. The hydro power plant is composed of 8 subsystems, and it has in total 249 states, 14 inputs, 18 coupling inputs and 18 coupling outputs.

All the tests have been performed using the CVODES integrator by (Serban and Hindmarsh, 2005) from the SUNDIALS suite (Hindmarsh et al., 2005).

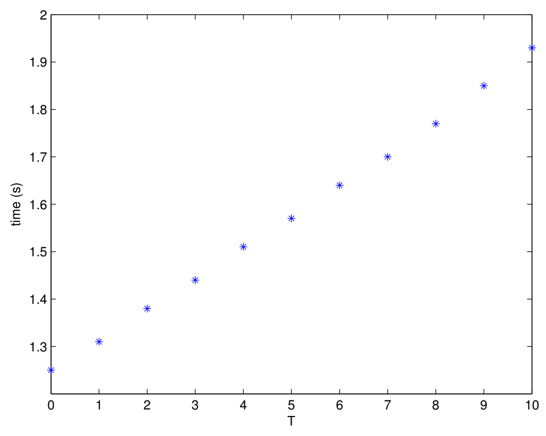

Fig. 3. Time required to integrate and linearize a subsystem dynamics for a time interval of 30 minutes using $S = 10$.

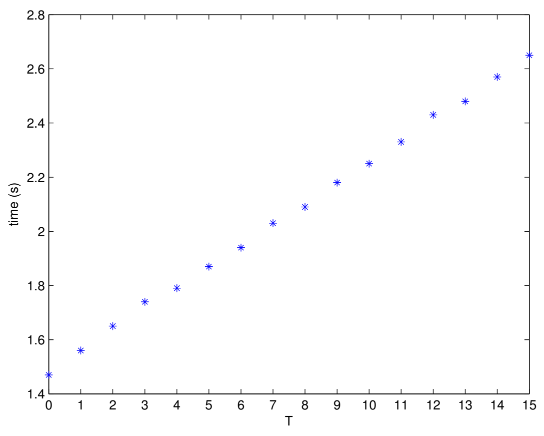

Fig. 4. Time required to integrate and linearize a subsystem dynamics for an time interval of 30 minutes using $S = 15$.

5.1 Generation of sensitivities for standard MS

To have a reference timing to compare our method we implemented a full model for the hydro power plant (i.e. not divided into subsystems) and computed the sensitivities needed to linearize the system dynamics containing 249 states.

The control specifications are such that the control horizon is 24 hours and the control inputs can be updated every 30 minutes. The time needed to integrate the dynamics and generate the sensitivities corresponding to one time interval of 30 minutes is 33.96 seconds [2] . This corresponds to a total sensitivity computation time of $33.96 \times 48 = 1630.1$ seconds.

5.2 Generation of sensitivities for DMS

To have an upper bound on the time needed for the linearization of F_n^i and G_n^i we performed some tests considering only the subsystem composed by the reach R_2 and the dam D_2 because it is the system with the highest number of states (41), coupling inputs (3) and ouputs (3). To implement efficiently the integration required in (17) we used error controlled quadrature states as provided by CVODES. In Figures 3 and 4 we can observe the time required to integrate and linearize the dynamics of this subsystem for an interval of 30 minutes. The first figure shows the time spent when $S = 10$ and the number of significant coefficients T varies from 0 to 10. The second figure shows the same kind of experiment with $S = 15$ and T varying from 0 to 15. The number of quadrature states needed is 33 for $S = 10$ and 48 for $S = 15$. From the figure we can observe two important facts:

- Considerable time savings can be achieved chosing $T < S$ to apply the adjoint based SQP method.
- The maximum time needed for integrating and linearizing the dynamics for one subsystem (1.92 s for $S = 10$ and 2.69 s for $S = 15$) is much lower than the time required in the standard MS divided by the number of subsystems $33.96/8 = 4.25$. This means that using DMS can give advantages w.r.t. using MS also without a parallel implementation. E.g., choosing $S = 15$ and $T = 5$ the total sensitivity computation time would be lower than $1.84 \times 8 \times 48 = 706.56$ seconds and therefore less than half of the time required by MS. Furthermore this computation can be parallelized into $8 \times 48 = 384$ task while the computation required by MS can be parallelized only into 48 tasks.

[2] The sensitivities have been generated in forward mode. An iterative linear solver has been used.

6. CONCLUSIONS

In this paper we presented an extended version of the distributed multiple shooting method introduced in (Savorgnan et al., 2011) which uses an adjoint based SQP framework. Adjoint-based distributed multiple shooting shows many benefits which are important when controlling large-scale systems:

- The algorithm is scalable because the subsystem dynamics can be integrated separatly. This makes it possible to use different integrators for the subsystems, which can be particularly useful when this method is applied to multiphysics systems.
- When integrators with step size control are used, the step is adapted for each subsystem individually, avoiding unnecessarily short steps for the other subsystems.
- Distributed multiple shooting considers each subsystem's dynamics independently. Due to this fact it is easier to validate and update the subsystem model independently, so that the maintenance effort can be carried out by different people.
- For a numerical example we have shown that considerable savings can be achieved for the generation of sensitivities.

Future work will include testing of the algorithm for systems with several thousands of states.

ACKNOWLEDGEMENTS

Research supported by Research Council KUL: GOA/11 /05 Ambiorics, GOA/10/09 MaNet, CoE EF/05/006 Optimization in Engineering (OPTEC), IOF-SCORES4CHEM;

FWO: G0226.06, G0321.06, G.0302.07, G.0320.08 (convex MPC), G.0558.08 (Robust MHE), G.0557.08, G.0588.09. Research communities (WOG: ICCoS, ANMMM, MLDM); G.0377.09 (Mechatronics MPC) IWT: PhD Grants, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, O&O-Dsquare Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007-2011); EU: ERNSI; FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCON (ICT-248940), SADCO Contract Research: AMINAL Other: Helmholtz: viCERP ACCM Bauknecht Hoerbiger.

REFERENCES

Albersmeyer, J. and Bock, H. (2008). Sensitivity Generation in an Adaptive BDF-Method. In H. Bock, E. Kostina, X. Phu, and R. Rannacher (eds.), *Modeling, Simulation and Optimization of Complex Processes: Proceedings of the International Conference on High Performance Scientific Computing, March 6-10, 2006, Hanoi, Vietnam*, 15–24. Springer.

Bauer, I., Bock, H., Körkel, S., and Schlöder, J. (2000). Numerical methods for optimum experimental design in DAE systems. *J. Comput. Appl. Math.*, 120(1-2), 1–15.

Bock, H., Diehl, M., Kostina, E., and Schlöder, J. (2007). Constrained Optimal Feedback Control of Systems Governed by Large Differential Algebraic Equations. In L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders (eds.), *Real-Time and Online PDE-Constrained Optimization*, 3 – 22. SIAM.

Bock, H. and Plitt, K. (1984). A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress Budapest*, 243–247. Pergamon Press.

Camponogara, E., Jia, D., Krogh, B., and Talukdar, S. (2002). Distributed model predictive control. *Control Systems Magazine*, 22(1), 44–52.

Deuflhard, P. (2006). *Newton Methods for Nonlinear Problems – Affine Invariance and Adaptative Algorithms*, volume 35 of *Springer Series in Computational Mathematics*. Springer, second edition.

Diehl, M., Walther, A., Bock, H.G., and Kostina, E. (2010). An adjoint-based SQP algorithm with quasi-Newton Jacobian updates for inequality constrained optimization. *Optimization Methods and Software*, 25, 531–552.

Findeisen, W., Bailey, F.N., Malinowski, K., Tatjewski, P., and Wozniak, A. (1980). *Control and Coordination in Hierarchical Systems*. Wiley.

Griewank, A. and Walther, A. (2002). On Constrained Optimization by Adjoint based quasi-Newton Methods. *Optimization Methods and Software*, 17, 869 – 889.

Hindmarsh, A., Brown, P., Grant, K., Lee, S., Serban, R., Shumaker, D., and Woodward, C. (2005). SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software*, 31, 363–396.

Laird, C.D. and Biegler, L.T. (2008). Large-scale nonlinear programming for multi-scenario optimization. In H.G. Bock, E. Kostina, H.X. Phu, and R. Ranache (eds.), *Modeling, Simulation and Optimization of Complex Processes*. Springer.

Magni, L. and Scattolini, R. (2006). Stabilizing decentralized model predictive control of nonlinear systems. *Automatica*, 42, 1231–1236.

Mesarovic, M., Macko, D., and Takahara, Y. (1970). *Theory of Hierarchical, Multilevel Systems*. Academic Press.

Petzold, L., Li, S., Cao, Y., and Serban, R. (2006). Sensitivity analysis of differential-algebraic equations and partial differential equations. *Computers and Chemical Engineering*, 30, 1553–1559.

Rawlings, J. and Stewart, B. (2008). Coordinating multiple optimization-based controllers: New opportunities and challenges. *Journal of Process Control*, 18, 839–845.

Richards, A. and How, J. (2007). Robust distributed model predictive control. *International Journal of Control*, 80(9), 1517–1531.

Savorgnan, C. and Diehl, M. (2010). Control benchmark of a hydro power plant. URL `http://homes.esat.kuleuven.be/~mdiehl/`.

Savorgnan, C., Romani, C., Kozma, A., and Diehl, M. (2011). Multiple shooting for distributed systems with applications in hydro electricity production. *Journal of Process Control*. doi:10.1016/j.jprocont.2011.01.011.

Scattolini, R. (2009). Architectures fior distributed and hierarchical model predictive control. *Journal of Process Control*, 19, 723–731.

Serban, R. and Hindmarsh, A. (2005). CVODES: the sensitivity-enabled ODE solver in SUNDIALS. In *Proceedings of IDETC/CIE 2005*.

Venkat, A. (2006). *Distributed Model Predictive Control: Theory and Applications*. Ph.D. thesis, University of Wisconsin-Madison.

Zhu, Y., Word, D., Siirola, J., and Laird, C. (2009). Exploiting modern computing architectures for efficient large-scale nonlinear programming. In *10th International Symposium on Process Systems Engineering: Part A, Computer Aided Chemical Engineering*, volume 27, 783–788.