

Abstract

The aim of this paper is to introduce a new method for the solution of optimal control problems for which the system is composed by many subsystems whose dynamics are coupled through input-output connections. The proposed approach can be regarded as a generalization of the direct multiple shooting method and exploits the structure of the problem to achieve a highly parallelizable algorithm. To demonstrate its effectiveness, the new method is applied to the control of a hydro power plant composed of several connected reaches.

Distributed direct multiple shooting with applications in hydro electricity production

C. Savorgnan, C. Romani, A. Kozma, M. Diehl *

Preprint submitted on 21/07/2010

1 Introduction

Manufacturing systems, process plants and networked systems are often composed by several subsystems. The control of these systems can be very challenging and this is the reason why a lot of research has been carried out in this field. Several results had been achieved in the past decades [16, 9] and a revived interest can be seen in recent years, in particular in the field of hierarchical and distributed model predictive control [5, 14, 26, 21, 20, 23] and control of multi agent systems [4, 25]. Most of these works assume that the deployment of local controllers is necessary to control this kind of large scale systems. However, the adoption of local controllers which are coordinated to achieve some global properties comes at a price. These methods often can only guarantee suboptimal performance and they often show extremely slow convergence.

When the use of local controllers is not imposed by design constraints, a centralized controller can be still a viable alternative. A straight implementation of standard control methods could not be applicable due to computational complexity and poor maintainability. To solve this problem it is necessary to exploit the problem structure to achieve a numerically efficient method [13, 27]. When designing a method for large scale systems it is important to keep in mind the architecture of the available computers and the possibility to keep model maintenance local. After decades of increasing clock speeds of the processors, last years have shown a new trend where the clock speeds are slowly changing but the number of processors available on a single workstation is increasing rapidly. This fact increases the importance to develop parallelizable algorithms.

An effective method to solve nonlinear optimal control problems with continuous time dynamics is the direct multiple shooting method [3]. This method divides the control horizon into intervals on which the control inputs are parametri-

*C. Savorgnan, A. Kozma and M. Diehl are with the Electrical Engineering Department (ESAT-SCD) and Optimization in Engineering Center (OPTEC), K.U. Leuven, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium.

C. Romani is with Dipartimento di Elettronica e Informazione, Politecnico di Milano, Via Ponzio 34/5, 20133 Milano, Italy

zed. A finite dimensional nonlinear programming problem is obtained by including in the formulation the value of the states only at the start of each time interval. Integrators are used to take care of the continuous-time dynamics between these points. The optimization problem can then be solved using nonlinear programming methods as sequential quadratic programming (SQP). When applied to multiple shooting, SQP requires the calculation of many derivatives. In practical applications, this constitutes a large amount of the computation time. **Contribution.** In this paper we introduce an extension of multiple shooting for optimal control problems which consider several subsystems coupled through input-output connections. The structure is exploited to distribute the computational load required by the integration of the subsystems. This is achieved by representing the coupling variables as a linear combination of basis functions. The new method is called Distributed Multiple Shooting (DMS) and shows the following advantages:

- the algorithm is highly parallelizable;
- different integrators can be used for the subsystems - this is a very useful property when dealing with multiphysics systems;
- by integrating the subsystem dynamics independently the step size control of the integrators can also be performed independently achieving substantial savings in computation time;
- the models of the subsystems can be modified and updated independently, giving more maintainability and design flexibility.

Simulation methods which are related to DMS can be found in the scientific literature which studies the spatial decomposition of partial differential equations (e.g., see the books [10, 19]). A method which is particularly related to the one we propose can be found in [7, 11] where a Newton method is used to improve the convergence.

Paper outline. In Section 2 the problem class considered is presented. Section 3 illustrates distributed multiple shooting, while in Section 4 the new method is applied to the control of hydro power plant. Section 5 contains a discussion on the proposed method and conclusions.

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 . Calligraphic capital letters represent sets.

2 Problem formulation

We consider continuous-time systems which can be decomposed into M coupled subsystems described by the following equations

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

The vector $x^{(i)}(t) \in \mathbb{R}^{n_x^{(i)}}$ represents the state of the subsystem i at time t . The vectors $u^{(i)}(t) \in \mathbb{R}^{n_u^{(i)}}$ and $z^{(i)}(t) \in \mathbb{R}^{n_z^{(i)}}$ both represent inputs of subsystem i : the former can be used to control the system while the latter is used to take into account the dynamic couplings between the subsystems. The vector $y^{(i)}(t) \in \mathbb{R}^{n_y^{(i)}}$ represents the output of subsystem i .

We assume that the subsystems are coupled through input-output connections. We use the equality

$$\left(z^{(i)}(t)\right)_p = \left(y^{(j)}(t)\right)_q \quad (2)$$

to indicate that the p -th coupling input of subsystem i is connected to the q -th output of subsystem j . To simplify the notation in the following sections, we define \mathcal{C} as the set containing all the quadruplets (i, p, j, q) representing the input-output connections of the form (2). E.g., consider the system in Figure 1. The set \mathcal{C} corresponding to this system contains the quadruplets $(1, 1, 3, 1)$, $(2, 1, 1, 2)$, $(3, 1, 2, 1)$ and $(3, 2, 1, 1)$.

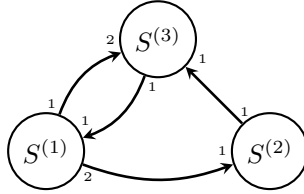


Figure 1: A system composed by $M = 3$ subsystems. The arrows represent the couplings while the numbers at the ends of the arrows indicate the output and coupling input components.

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

$$\begin{aligned} \min_{x,u,z,y} \quad & \sum_{i=1}^M \int_0^T \ell^{(i)}(t, x^{(i)}(t), u^{(i)}(t), z^{(i)}(t)) dt \\ & \dot{x}^{(i)}(t) = f^{(i)}(t, x^{(i)}(t), u^{(i)}(t), z^{(i)}(t)), \quad i = 1, \dots, M \\ & y^{(i)}(t) = g^{(i)}(t, x^{(i)}(t), u^{(i)}(t), z^{(i)}(t)), \quad i = 1, \dots, M \\ & x^{(i)}(t) \in \mathcal{X}^{(i)}, \quad u^{(i)}(t) \in \mathcal{U}^{(i)}, \quad i = 1, \dots, M \\ & x^{(i)}(0) = x_0^{(i)}, \quad i = 1, \dots, M \\ & \left(z^{(i)}(t)\right)_p = \left(y^{(j)}(t)\right)_q, \quad \forall (i, p, j, q) \in \mathcal{C} \end{aligned} \quad (3)$$

where the terminal time T and the initial condition $x_0^{(i)}$ for the subsystems are fixed. The sets $\mathcal{X}^{(i)}$ and $\mathcal{U}^{(i)}$ represent the constraints on the state and control input vectors, respectively.

In the following we assume that the functions $g^{(i)}$ are continuous and that the optimal control problem (3) is well defined and that it has a solution.

Remark 1. Notice that problem (3) consists of M independent optimal control problems which are coupled only by the last constraint.

3 The distributed multiple shooting method

Following the same approach used in standard multiple shooting, the method we propose addresses the infinite dimensional problem (3) in two steps: the problem is first discretized and then the finite dimensional nonlinear programming problem (NLP) is solved using a suitable optimization method. We refer to the new method as Distributed Multiple Shooting (DMS).

3.1 Control input discretization

The discretization of the control inputs is obtained - as in most direct methods - by dividing the interval $[0, T]$ into N sub-intervals on which a finite parametrization for the input $u(t)$ is used. Define the time points

$$0 = t_0 < t_1 < \dots < t_N = T. \quad (4)$$

Using a piece-wise constant input parametrization we obtain

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

and

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

3.2 Coupling discretization

To achieve a parallelization across subsystems we propose to use a finite dimensional representation of $z^{(i)}(t)$ and $y^{(i)}(t)$. Define the vector of basis functions for an interval $[t_a, t_b]$

$$\Gamma(t) = [\gamma_0(t) \ \gamma_1(t) \ \gamma_2(t) \ \dots \ \gamma_S(t)]^T. \quad (7)$$

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

$$\left(z^{(i)}(t) \right)_p = \Gamma_n(t)^T a_{n,p}^{(i)} \quad (8)$$

where $a_{n,p}^{(i)} \in \mathbb{R}^{S+1}$ is a coefficient vector and

$$\Gamma_n(t) = \Gamma \left(t_a + \frac{t - t_n}{t_{n+1} - t_n} (t_b - t_a) \right). \quad (9)$$

Define the vector

$$b_{n,q}^{(i)} = \arg \min_{b \in \mathbb{R}^{S+1}} \int_{t_n}^{t_{n+1}} \left(\Gamma_n(t)^T b - \left(y^{(i)}(t) \right)_q \right)^2 \omega(t) dt \quad (10)$$

where $\omega(t) > 0$ is a weight function. In DMS the input-output couplings are represented by the constraint

$$a_{n,p}^{(i)} = b_{n,q}^{(j)} \quad \forall (i, p, j, q) \in \mathcal{C}. \quad (11)$$

For notational convenience we define the matrices

$$a_n^{(i)} = \begin{bmatrix} a_{n,1}^{(i)} & \dots & a_{n,n_z}^{(i)} \end{bmatrix} \quad \text{and} \quad b_n^{(i)} = \begin{bmatrix} b_{n,1}^{(i)} & \dots & b_{n,n_y}^{(i)} \end{bmatrix}. \quad (12)$$

Later in this section we will discuss the choice of $\Gamma(t)$ and how to compute the vectors $b_n^{(i)}$ from the simulation output $y^{(i)}(t)$. In the remainder of the paper we will rely on the assumption that a good approximation of $y^{(i)}$ and $z^{(i)}$ can be obtained with a relatively small number of basis functions.

3.3 State discretization

To simplify the exposition we use the same intervals for the discretization of the input and the state. However, in special cases, we may want to use a different discretization for the state than the one used for the input. Note that in this case we shall use the finest grid for the coupling constraints. Denote by $s_n^{(i)}$ the value of $x^{(i)}(t_n)$. Given $s_n^{(i)}, u_n^{(i)}, a_n^{(i)}$ 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 $F_n^{(i)}(\cdot), L_n^{(i)}(\cdot)$ and $G_n^{(i)}(\cdot)$ as

$$F_n^{(i)}(s_n^{(i)}, u_n^{(i)}, a_n^{(i)}) := x_n^{(i)}(t_{n+1}), \quad (13)$$

$$L_n^{(i)}(s_n^{(i)}, u_n^{(i)}, a_n^{(i)}) := \int_{t_n}^{t_{n+1}} \ell^{(i)}(t, x_n^{(i)}(t), u_n^{(i)}(t), \Gamma(t)^T a_n^{(i)}) dt \quad (14)$$

and

$$G_n^{(i)}(s_n^{(i)}, u_n^{(i)}, a_n^{(i)}) := b_n^{(i)}. \quad (15)$$

It should be understood that $F_n^{(i)}, L_n^{(i)}$ and $G_n^{(i)}(s_n^{(i)}, u_n^{(i)}, a_n^{(i)})$ are defined up to the approximation error introduced by the coupling input discretization.

3.4 NLP formulation

DMS solves the following NLP

$$\begin{aligned} \min_{u_n^{(i)}, s_n^{(i)}, a_n^{(i)}, b_n^{(i)}} \quad & \sum_{n=0}^{N-1} \sum_{i=1}^M L_n^{(i)}(s_n^{(i)}, u_n^{(i)}, a_n^{(i)}) \\ & s_{n+1}^{(i)} = F_n^{(i)}(s_n^{(i)}, u_n^{(i)}, a_n^{(i)}) \quad n = 0, \dots, N-1 \\ & b_n^{(i)} = G_n^{(i)}(s_n^{(i)}, u_n^{(i)}, a_n^{(i)}) \quad n = 0, \dots, N-1 \\ & s_0^{(i)} = x_0^{(i)}, \quad s_n^{(i)} \in \mathcal{X}^{(i)} \quad n = 1, \dots, N \\ & u_n^{(i)} \in \mathcal{U}^{(i)} \quad n = 0, \dots, N \\ & a_{n,p}^{(i)} = b_{n,q}^{(j)} \quad n = 0, \dots, N-1 \quad \forall (i, p, j, q) \in \mathcal{C} \end{aligned} \quad (16)$$

As in standard multiple shooting, Problem (16) can be solved using sequential quadratic programming (SQP). This method is based on the iterative solution

of a local quadratic approximation of the original NLP. The deployment of SQP requires the computation of the Jacobians of $L_n^{(i)}$, $F_n^{(i)}$ and $G_n^{(i)}$ (denoted as $J_{L_n^{(i)}}$, $J_{F_n^{(i)}}$ and $J_{G_n^{(i)}}$) and, depending on the implementation, the Hessian of the Lagrangian or an approximation (in our example in Section 4 we used a Gauss-Newton approximation).

The reason why it is desirable to have a low order parametrization to represent $y^{(i)}$ and $z^{(i)}$ is the following: applying the SQP method for the solution of (16) requires the computation of derivatives with respect to all elements of $a_n^{(i)}$. Therefore, if a large number of parameters is used, the benefits introduced by DMS which we shall illustrate can vanish due to the computational overhead introduced.

When compared to standard single or multiple shooting, DMS shows the following benefits:

- The evaluation of $L_n^{(i)}$, $F_n^{(i)}$ and $G_n^{(i)}$ together with their directional derivative information can be carried out separately for every subsystem and every time interval. Therefore, this part of the computation, which often constitutes the largest amount of the computational time, can be parallelized in $N \times M$ completely independent tasks. In standard MS there are only N independent tasks corresponding to the time intervals.
- The evaluation of $L_n^{(i)}$, $F_n^{(i)}$ and $G_n^{(i)}$ is performed using integrators. The fact the computations are independent for the different subsystem allows the use of different integrators for the subsystems. This can be particularly useful when DMS is applied to multiphysics systems. Furthermore, when integrators with step size control are used, the step is adapted for each subsystem individually, avoiding unnecessarily short steps for the other subsystems.
- Since DMS considers the subsystem dynamics independently, it is easier to validate and update the subsystem models independently, and the model maintenance effort can be easily performed by different people.

3.5 Choice of basis functions and generation of derivatives

As we already mentioned, the evaluation of $F_n^{(i)}$ and $L_n^{(i)}$ can be simply carried out using an integrator. The value of the first function can be achieved by integrating the i -th subsystem dynamics, while the value of the second function can be obtained adding an extra state to the dynamics or using a quadrature formula. The Jacobian and, possibly, the Hessian of this functions can be calculated by using automatic or numerical differentiation, or integrating the sensitivity equations [2, 18, 1]. Several integrators provide this functionality.

When choosing the basis function vector $\Gamma(t)$ it is desirable that computational and implementation overhead for the evaluation of $G_n^{(i)}$ and its Jacobian are not too big. This feature can be obtained by using normalized orthogonal

polynomials, i.e. polynomials satisfying¹

$$\int_{t_a}^{t_b} \gamma_n(t)\gamma_m(t)\omega(t)dt = \begin{cases} 0 & \text{if } n \neq m \\ 1 & \text{otherwise} \end{cases} \quad (17)$$

where $\omega(t) \geq 0$ is a weight function. We will illustrate how $G_n^{(i)}$ and its Jacobian can be evaluated when we consider Legendre and Chebyshev polynomials.

3.5.1 Legendre polynomials

Legendre polynomials are obtained by setting $t_a = -1$, $t_b = 1$ and $\omega(t) = 1$. Thanks to the normalization in (17)

$$G_n^{(i)}(s_n^{(i)}, u_n^{(i)}, a_n^{(i)}) = \frac{2}{t_{n+1} - t_n} \int_{t_n}^{t_{n+1}} \Gamma_n(t) \left(y_n^{(i)}(t) \right)^T dt \quad (18)$$

Notice that the integrand in the previous equation is a matrix and therefore the integration is element-wise. The evaluation of $G_n^{(i)}$ can be evaluated with the same method used for $L_n^{(i)}$, i.e. adding extra states to the system dynamics or using a quadrature formula. The Jacobian of $G_n^{(i)}$ can then be achieved using the same differentiation technique used for the Jacobian of $F_n^{(i)}$ and $L_n^{(i)}$.

3.5.2 Chebyshev polynomials

Chebyshev polynomials are obtained by setting $t_a = -1$, $t_b = 1$ and $\omega(t) = \frac{1}{\sqrt{1-t^2}}$. Due to the fact that $\omega(t) \rightarrow \infty$ as $t \rightarrow \pm 1$, using (18) for Chebyshev polynomials is not practical since the integrand tends to infinity as $t \rightarrow \pm 1$. Exploiting the fact that Chebyshev polynomials satisfy a discrete orthogonality condition [15], $G_n^{(i)}$ can be approximately evaluated with the following formula

$$G_n^{(i)}(s_n^{(i)}, u_n^{(i)}, a_n^{(i)}) \approx \frac{\pi}{Q+1} \sum_{w=1}^{Q+1} \Gamma_n(t_w) \left(y^{(i)}(t_w) \right)^T \quad (19)$$

where $Q \geq S$ and t_w are the zeros of $\gamma_{Q+1}(t)$

$$t_w = \cos \left(\frac{(w - \frac{1}{2})\pi}{Q+1} \right) \quad (20)$$

The accuracy of the approximation (19) improves for increasing values of Q .

The values of $y^{(i)}(t_w)$ used in (19) can be obtained directly from the integrator used to evaluate $F_n^{(i)}$ and $L_n^{(i)}$. To compute the Jacobian $G_n^{(i)}$ we can apply the chain rule to (19). The evaluation will require the computation of the Jacobian of $x^{(i)}(t)$ for every value of t_w which can also be obtained from the integrator.

¹Usually, the definition of orthogonal polynomials requires that $\int_{t_a}^{t_b} \gamma_n^2(t)\omega(t)dt \neq 0$ and therefore it determines every polynomial $\gamma_n(t)$ up to a constant which is specified using some standardization, e.g. $\gamma_n(t_b) = 1$.

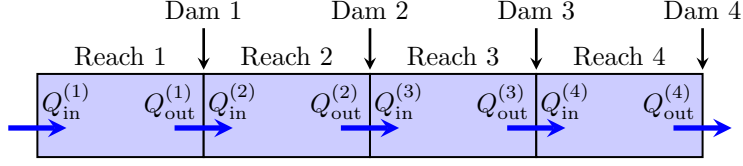


Figure 2: A schematic representation of the hydro power plant. $Q_{\text{in}}^{(i)}$ and $Q_{\text{out}}^{(i)}$ represent the input and output water discharges of the reach i , respectively.

4 Example: control of a hydro power plant

To demonstrate the effectiveness of DMS we deployed this method combined with model predictive control (MPC) for the regulation of a hydro power plant composed by four dams located along the same river. The dams divide the river into four reaches which we regard as subsystems. The subsystems are coupled since the output discharge of one reach constitutes the input discharge of the next reach (see Figure (2)). There are three main specifications for the control which we list in order of importance:

1. the water level must be kept between some prescribed values;
2. the total power generated by the turbines should follow a reference profile which is provided 24 hours in advance;
3. if the first two specifications can be met, then the level of the water should be kept as constant as possible.

The application of linear MPC to the control of hydro power plants can be found in [24, 22]. Here, we use a nonlinear MPC scheme which is necessitated in particular by the nonlinear functions describing the power output.

4.1 The system model

A schematic representation of the system is given in Figure 2. The part of the river considered consist of an upper reach which precedes the first dam and the three reaches delimited by the dams. To model the water flow we use the one dimensional Saint-Venant partial differential equation (also known as shallow water equation)

$$\begin{cases} \frac{\partial Q(z, t)}{\partial z} + \frac{\partial S(z, t)}{\partial t} = 0 \\ \frac{1}{g} \frac{\partial}{\partial t} \left(\frac{Q(z, t)}{S(z, t)} \right) + \frac{1}{2g} \frac{\partial}{\partial z} \left(\frac{Q^2(t, z)}{S^2(t, z)} \right) + \frac{\partial H(t, z)}{\partial z} + I_f(z, t) - I_0(z) = 0 \end{cases} \quad (21)$$

where t is the time variable, z is the space variable, $Q(z, t)$ is the water discharge, $S(z, t)$ is the wetted cross section area, $H(z, t)$ is the water level, g is

the gravitational acceleration, $I_f(z, t)$ is the friction slope, and $I_0(z)$ is the slope of the river bed.

The following assumptions are made regarding the reach geometry: the cross section of the river is rectangular, the river width (w) and the river slope are constant along every reach. These assumptions allow us to express the wetted cross section area as $S(z, t) = wH(z, t)$, so that we can express the Saint Venant equation only in terms of Q and H .

I_f can be found using the Strickler formula

$$I_f(z, t) = \frac{U^2(z, t)}{k_{\text{str}}^2 R^{4/3}(z, t)} \quad U(z, t) = \frac{Q(z, t)}{S(z, t)} \quad (22)$$

where $R(z, t) = \frac{S(z, t)}{P(z, t)}$ is the hydraulic radius and $P(z, t) = w + 2H(z, t)$ is the wetted perimeter. The constant k_{str} is the Gauckler-Manning-Strickler coefficient. The Saint-Venant equation has been discretized using finite differences to obtain an ODE. The state of the reach is composed by the values of the water discharges and levels at the discretization points.

The output water discharge of reach i ($Q_{\text{out}}^{(i)}$) is given by the sum of two components:

- The water discharge $Q_t^{(i)}$ which flows through the turbines and generates electricity. This is the control input of the i -th subsystem which can be modified every 20 minutes.
- The water discharge $Q_b^{(i)}$ which flows through a bypass which is installed in every dam. It is modeled using Torricelli's law

$$Q_b^{(i)}(t) = k_b^{(i)} S_b^{(i)} \sqrt{2gH_b^{(i)}(t)} \quad (23)$$

where $S_b^{(i)}$ is the bypass opening surface (which is kept constant during normal operation), $H_b^{(i)}(t)$ is the water level w.r.t. the bypass opening, and $k_b^{(i)}$ is a constant to take into account that geometry of the opening. It is assumed that the water level at the beginning of reach $i + 1$ is lower than the bypass on the i -th dam so that the water can flow through the bypass only in one direction.

The input-output connection considered is such that reach i influences reach $i + 1$ but not vice versa. The power produced by the subsystem i is given by

$$P^{(i)}(t) = k_t^{(i)} Q_t^{(i)}(t) H_t^{(i)}(t) \quad (24)$$

where $H_t^{(i)}(t)$ is the turbine head and $k_t^{(i)}$ is a constant which characterizes the turbine operation.

4.2 Optimal control problem formulation

To satisfy the control specification, the selected optimal control problem formulation uses hard constraints to impose bounds on the water levels, a \mathcal{L}_1 penalty to reduce the power tracking error and a \mathcal{L}_2 penalty to reduce the deviation of the current value of the water level w.r.t. a reference level.

Denote by $H^{(i)}(t)$ the vector containing the water levels corresponding to the discretization points of the Saint-Venant PDE for the reach i and by $H_r^{(i)}$ the vector of reference levels. The state and input constraints, the subsystems dynamics and the couplings fit the framework we presented in the previous sections. The \mathcal{L}_1 and \mathcal{L}_2 penalties are added to the cost

$$\int_0^T \gamma \|P_r(t) - \sum_{i=1}^4 P^{(i)}(t)\|_1 dt + \sum_{i=1}^M \int_0^T \|H^{(i)}(t) - H_r^{(i)}\|_2^2 dt. \quad (25)$$

where $P_r(t)$ denotes the power reference profile and $\gamma = 50$. The first term in the cost does not fit the framework we proposed. However, we can reformulate the cost using the slack variable $\epsilon(t)$

$$\int_0^T \gamma \epsilon(t) dt + \sum_{i=1}^M \int_0^T \|H^{(i)}(t) - H_r^{(i)}\|_2^2 dt \quad (26)$$

and imposing the constraint

$$-\epsilon(t) \leq P_r(t) - \sum_{i=1}^4 P^{(i)}(t) \leq \epsilon(t). \quad (27)$$

In the discretization of the OCP the constraint (27) is imposed only for $t = t_0, \dots, t_{N-1}$. Using the notation $\epsilon_n = \epsilon(t_n)$, the discretized version of the constraint (27) reads

$$-\epsilon_n \leq P_r(t_n) - \sum_{i=1}^4 P^{(i)}(t_n) \leq \epsilon_n. \quad (28)$$

while the first term in the cost (25) becomes $\gamma \sum_{n=0}^{N-1} (t_{n+1} - t_n) \epsilon_n$. Therefore, the formulation of the \mathcal{L}_1 penalty does not preclude the possibility of integrating the subsystem dynamics separately and the framework presented in the previous sections can be easily extended to deal with the selected formulation.

4.3 Simulation results

The method proposed has been implemented using MATLAB[®] and the BDF integrator from the ACADO Toolkit [12]. An SQP solver has been implemented using as QP solvers the codes qpOASES [8] and CPLEX[®] alternatively. The results have been compared to the ones obtained implementing single shooting (SS) and MS. For all the three implementation the same software has been used.

In the simulation we considered identical reaches with the following parameters (the subsystem index is omitted): $w = 100$ m, $I_0 = 0.0033$, $k_t = 8$ kJm⁻⁴, $k_w = 0.6$, $S_w = 18.26$ m², $k_{str} = 30$ m^{1/3}s⁻¹ (stony earth channel). Every reach is 4 km long and the turbines and bypasses are located at the bottom of the reach so that $H_t^{(i)}(t) = H_b^{(i)}(t) = H^{(i)}(z_{reach}, t)$. The turbine discharges are bounded in the interval $[50, 150]$.

The Saint Venant PDE has been discretized using 5 cells for every reach. The reference level vectors $H_r^{(i)}$ correspond to the steady state condition of the system which can be found simulating the system setting $Q_{in}^{(1)} = 300$ and $Q_t^{(i)} = 100$. The water levels are constrained to take values in intervals of ± 1 m w.r.t. $H_r^{(i)}$. The reference power profile used in the simulations is shown in Figure 3 (it is assumed that the same profile repeats every day). The initial condition of the system corresponds to the steady state for which we computed $H_r^{(i)}$.

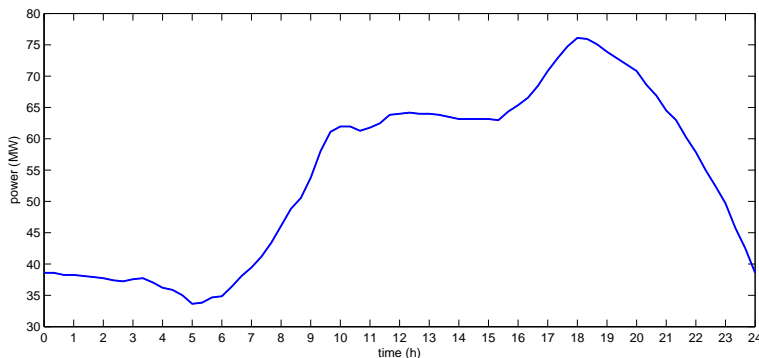


Figure 3: Reference power profile used in the simulation.

We performed a MPC simulation for all the shooting methods considered. To reduce the computation load we used real-time iteration [6] which uses only one SQP iteration for every MPC iteration. Figure 4 shows the results of the simulation using DMS with Legendre polynomials up to degree 2. All the methods implemented achieved a perfect power tracking performance.

In the rest of this subsection we report on the timing of the two main parts of the computational load. All the simulations have been carried out on a computer with a Intel[®] Core2 Quad CPU 2.66GHz processor and 4 GB of memory running GNU/Linux. We discuss the average timings during the simulations.

4.3.1 QP solution

We implemented two variants of the QP solution within the SQP method, both of them performing a full step at every iteration. In the first variant we solved the full QP arising from the linearization of the NLP. In our formulation, the

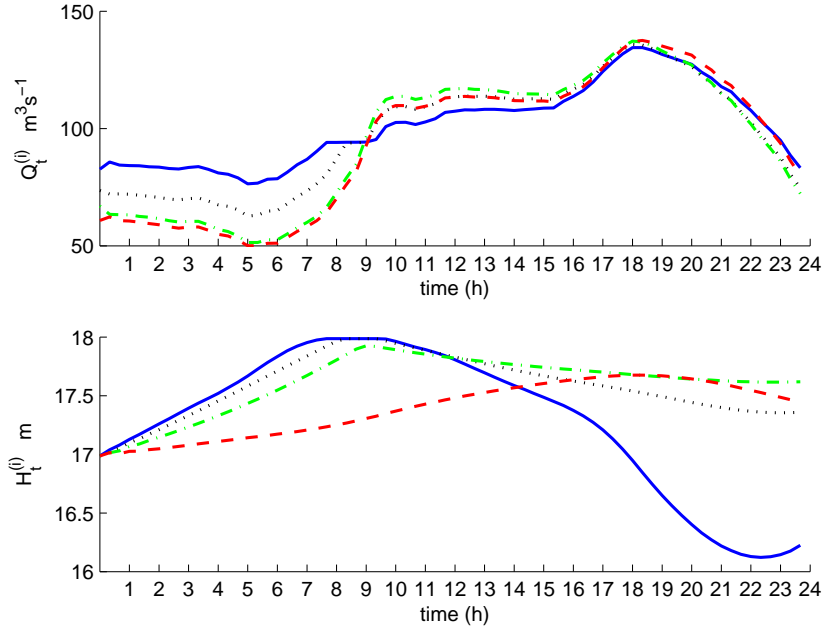


Figure 4: The top graph shows the turbines water discharges, while the bottom graph shows the turbine heads. The solid, dotted, dash-dotted and dashed lines represent quantities corresponding to the first, second, third and fourth reaches, respectively. The allowed intervals where the discharges and the levels in the graph can take their values are $[50, 150]$ and $[16, 18]$, respectively.

QP consists of 3245 decision variables, the same number of lower and upper bounds, 2880 equality constraints and 146 inequality constraints. The solution for the full QP which has a sparse structure takes 0.7 s using CPLEX. In the second variant we applied condensing, i.e. we eliminated all the state variable expressing them in terms of the inputs and initial condition. QP condensing in MATLAB and the QP solution using qpOASES require 1.15 and 1.7 seconds, respectively. For both the variants of the SQP method applied to DMS we explicitly eliminated the variables corresponding to the couplings. This results in the same QP size for all the three shooting methods implemented.

4.3.2 Integration and linearization

The current software implementation of SS, MS and DMS is not running on a parallel cluster. However, the timing of operations we performed allows us to discuss the advantages of the three methods.

The integration and linearization of the dynamics of the whole system for a time interval of 20 minutes takes 1.15 s. In SS and MS this operation is repeated

72 times for a total computational time of 82.8 s. In MS these 72 tasks can be carried out in parallel while in SS they need to be executed in series. This well known fact shows that MS can be really advantageous w.r.t. SS, especially when many intervals are used in the time discretization.

When only one of the four subsystem is considered the time needed for integration and linearization reduces to 0.16 s. Note that this is less than $1.15/4$ s due to fewer directional derivatives. This fact shows that DMS can give big advantages for both a parallelized and even a serial implementation of the algorithm:

- in a parallelized implementation integration and linearization of the dynamics on the complete horizon corresponds to 288 independent tasks;
- on a single CPU integration and linearization of the four subsystems for one time interval gives a computational time of only $4 \times 0.16 = 0.64$ s. This is considerably smaller than the time of 1.15 s needed for the corresponding operation implemented in SS or MS. In larger scale settings this advantage can be even more pronounced.

5 Conclusions

In this paper we proposed a variant of direct multiple shooting which exploits the system structure to achieve a highly parallelizable algorithm. Being based on multiple shooting, our new method inherits its advantages but it has a higher parallelizability and allows us to use different integrators for the subsystems.

The main idea behind the method is that the couplings between the subsystems can have a finite dimensional representation using a set of basis functions. This however corresponds to an approximation in the system dynamics which introduces a trade-off between the computational load (which is low when a basis containing few functions is used) and the accuracy of the solution (which is high if many basis functions are used).

The preliminary simulation results show that the proposed method performs well in terms of convergence while decreasing the computational load required per SQP iteration.

As a final remark we would like to point out that DMS can be used as a basis for the derivation of fully distributed multiple shooting schemes where several local optimizers exchange information to achieve a common control goal, e.g. using the ideas proposed in [17].

References

- [1] J. Albersmeyer and M. Diehl. The Lifted Newton Method and its Application in Optimization. *SIAM Journal on Optimization*, 20(3):1655–1684, 2010.

- [2] I. Bauer, H.G. Bock, S. Körkel, and J.P. Schlöder. Numerical methods for optimum experimental design in DAE systems. *J. Comput. Appl. Math.*, 120(1-2):1–15, 2000.
- [3] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress Budapest*, pages 243–247. Pergamon Press, 1984.
- [4] N.R. Bussmann, S. Jennings and M. Wooldridge. *Multiagent Systems for Manufacturing Control*. Springer, 2004.
- [5] E. Camponogara, D. Jia, B.H. Krogh, and S. Talukdar. Distributed model predictive control. *Control Systems Magazine*, 22(1):44–52, 2002.
- [6] M. Diehl, H.G. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and Nonlinear Model Predictive Control of Processes governed by differential-algebraic equations. *J. Proc. Contr.*, 12(4):577–585, 2002.
- [7] Q. Du and M.D. Gunzburger. A gradient method approach to optimization-based multidisciplinary simulations and nonoverlapping domain decomposition algorithms. *SIAM Journal on Numerical Analysis*, 37(5):1513–1541, 2000.
- [8] H. J. Ferreau, H. G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008.
- [9] W. Findeisen, F. N. Bailey, K. Malinowski, P. Tatjewski, and A. Wozniak. *Control and Coordination in Hierarchical Systems*. Wiley, 1980.
- [10] J. Geiser. *Decomposition Methods fo Differential Equations*. Chapman & Hall/CRC, 2009.
- [11] M.D. Gunzburger and H. Kwon Lee. An optimization-based domain decomposition method for the navier-stokes equations. *SIAM Journal on Numerical Analysis*, 37(5):1455–1480, 2000.
- [12] B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, (DOI: 10.1002/oca.939), 2010. (in print).
- [13] C. D. Laird and L. T. Biegler. Large-scale nonlinear programming for multi-scenario optimization. In H. G. Bock, E. Kostina, H-X Phu, and R. Ranache, editors, *Modeling, Simulation and Optimization of Complex Processes*. Springer, 2008.
- [14] L. Magni and R. Scattolini. Stabilizing decentralized model predictive control of nonlinear systems. *Automatica*, 42:1231–1236, 2006.

- [15] J.C. Mason and D.C. Handscomb. *Chebyshev Polynomials*. Chapman & Hall/CRC, 2003.
- [16] M.D. Mesarovic, D. Macko, and Y. Takahara. *Theory of Hierarchical, Multilevel Systems*. Academic Press, 1970.
- [17] I. Necoara, C. Savorgnan, Q. Tran Dinh, J. A. K. Suykens, and M. Diehl. Distributed Nonlinear Optimal Control Using Sequential Convex Programming and Smoothing Techniques. In *Proceedings of the 48th IEEE Conference on Decision and Control*, Shanghai, China, 2009.
- [18] L. Petzold, S. Li, Y. Cao, and R. Serban. Sensitivity analysis of differential-algebraic equations and partial differential equations. *Computers and Chemical Engineering*, 30:1553–1559, 2006.
- [19] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford University Press, 1999.
- [20] J.B. Rawlings and B.T. Stewart. Coordinating multiple optimization-based controllers: New opportunities and challenges. *Journal of Process Control*, 18:839–845, 2008.
- [21] A. Richards and J.P. How. Robust distributed model predictive control. *International Journal of Control*, 80(9):1517–1531, 2007.
- [22] A. Sahin and M. Morari. Decentralized model predictive control for a cascade of river power plants. In *Intelligent Systems, Control and Automation: Science and Engineering*, volume 42, pages 463–486. Springer, 2010.
- [23] R. Scattolini. Architectures for distributed and hierarchical model predictive control. *Journal of Process Control*, 19:723–731, 2009.
- [24] C. Setz, A. Heinrich, P. Rostalski, G. Papafotiou, and M. Morari. Application of model predictive control to a cascade of river power plants. In *Proceedings of the 17th World Congress The International Federation of Automatic Control*, Seoul, Korea, 2008.
- [25] J.S. Shamma, editor. *Cooperative Control of Distributed Multi-Agent Systems*. Wiley, 2007.
- [26] A.N. Venkat. *Distributed Model Predictive Control: Theory and Applications*. PhD thesis, University of Wisconsin-Madison, 2006.
- [27] Y. Zhu, D. Word, J. Sirola, and C.D. Laird. 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, pages 783–788, 2009.