# GPU-Accelerated Stochastic Predictive Control of Drinking Water Networks

Ajay Kumar Sampathirao, Pantelis Sopasakis, Alberto Bemporad, *Fellow, IEEE*, and Panagiotis (Panos) Patrinos

***Abstract*— Despite the proven advantages of scenario-based stochastic model predictive control for the operational control of water networks, its applicability is limited by its considerable computational footprint. In this paper, we fully exploit the structure of these problems and solve them using a proximal gradient algorithm parallelizing the involved operations. The proposed methodology is applied and validated on a case study: the water network of the city of Barcelona.**

***Index Terms*— Accelerated proximal gradient (APG) method, drinking water networks (DWNs), graphics processing units (GPUs), stochastic model predictive control (SMPC).**

## I. Introduction

### A. Motivation

WATER utilities involve energy-intensive processes, complex in nature (dynamics) and form (topology of the network), of rather large scale and with interconnected components, subject to uncertain water demands from the consumers and are required to supply water uninterruptedly. These challenges call for operational management technologies able to provide reliable closed-loop behavior in the presence of uncertainty. In 2014, the IEEE Control Systems Society identified many aspects of the management of complex water networks as emerging future research directions [1].

Stochastic model predictive control (SMPC) is an advanced control scheme which can address effectively the above-mentioned challenges and has already been used for the management of water networks [2], [3], power networks [4], and other uncertain systems. SMPC is also accompanied by a wealth of theoretical results regarding its stability and recursive feasibility properties [5], [6]. However, unless restrictive assumptions are adopted regarding the form of the disturbances, such problems are known to be computationally intractable [3], [7]. In this paper, we combine an accelerated dual proximal gradient algorithm with general-purpose graphics processing units (GPUs) to deliver a computationally feasible solution for the control of water networks.

### B. Background

The *pump scheduling problem* (PSP) is an optimal control problem for determining an open-loop control policy for the operation of a water network. Such open-loop approaches are known, since the 1980's [8], [9]. More elaborate schemes have been proposed, such as [10], where a nonlinear model is used along with a demand forecasting model to produce an optimal open-loop 24-h-ahead policy. Recently, the problem was formulated as a mixed-integer nonlinear program to account for the ON/OFF operation of the pumps [11]. Heuristic approaches using evolutionary algorithms, genetic algorithms, and simulated annealing have also appeared in the literature [12]. However, a common characteristic and shortcoming of these studies is that they assume to know the future water demand and they do not account for the various sources of uncertainty which may alter the expected smooth operation of the network.

The effect of uncertainty can be attenuated by feedback from the network combined with the optimization of a performance index taking into account the system dynamics and constraints as in PSP. This, naturally, gives rise to Model Predictive Control (MPC) which has been successfully used for the control of drinking water networks (DWNs) [13], [14]. Recently, Bakker *et al.* [15] demonstrated experimentally on five full-scale water supply systems that MPC will lead to a more efficient water supply and better water quality than a conventional level controller. Distributed and decentralized MPC formulations have been proposed for the control of large-scale water networks [16], [17], while MPC has also been shown to be able to address complex system dynamics such as the Hazen–Williams pressure-drop model [18].

Most MPC formulations either assume exact knowledge of the system dynamics and future water demands [14], [17] or endeavor to accommodate the worst case scenario [13], [19]–[21]. The former approach is likely to lead to adverse behavior in the presence of disturbances which inevitably act on the system, while the latter turns out to be too conservative as we will later demonstrate in this paper.

When probabilistic information about the disturbances is available it can be used to refine the MPC problem

Manuscript received February 11, 2016; revised October 27, 2016; accepted February 4, 2017. Date of publication March 21, 2017; date of current version February 8, 2018. Manuscript received in final form February 26, 2017. This work was supported by the EU FP7 Research Project EFFINET Efficient Integrated Real-time Monitoring and Control of Drinking Water Networks under Grant 318556. The work of P. Patrinos was supported by the KU Leuven Research Council under Grant BOF/STG-15-043. Recommended by Associate Editor E. Kerrigan.

A. K. Sampathirao is with the Control Systems Group, Technical University of Berlin, D-10587 Berlin, Germany (email: sampathiroa@control.tu-berlin.de).

P. Sopasakis is with the STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, Department of Electrical Engineering (ESAT), KU Leuven, 3001 Leuven, Belgium (email: pantelis.sopasakis@kuleuven.be).

A. Bemporad is with the IMT Institude for Advanced Studies Lucca, 55100 Lucca, Italy.

P. Patrinos is with the STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, Department of Electrical Engineering (ESAT), KU Leuven, 3001 Leuven, Belgium (e-mail: panos.patrinos@esat.kuleuven.be).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCST.2017.2677741

formulation. The uncertainty is reflected onto the cost function of the MPC problem deeming it a random variable; in SMPC, the index to minimize is typically the expectation of such a random cost function under the (uncertain) system dynamics and state/input constraints [22], [23].

SMPC leads to the formulation of optimization problems over spaces of random variables which are, typically, infinite-dimensional. Assuming that disturbances follow a normal probability distribution facilitates their solution [7], [24], [25]; however, such an assumption often fails to be realistic. The normality assumption has also been used for the stochastic control of DWNs aiming at delivering high quality of services—in terms of demand satisfaction—while minimizing the pumping cost under uncertainty [2].

An alternative approach, known as *scenario-based stochastic MPC*, treats the uncertain disturbances as discrete random variables without any restriction on the shape of their distribution [26]–[28]. The associated optimization problem in these cases becomes a discrete multistage stochastic optimal control problem [29]. Scenario-based problems can be solved algorithmically; however, their size can be prohibitively large making them impractical for control applications of water networks as pointed out by Goryashko and Nemirovski [20]. This is demonstrated by Grosso *et al.* [3] who provide a comparison of the two approaches. Although compression methodologies have been proposed—such as the scenario tree generation methodology of Heitsch and Römisch [30]—multistage stochastic optimal control problems may still involve up to millions of decision variables.

GPUs have been used for the acceleration of the algorithmic solution of various problems in signal processing [31], computer vision and pattern recognition [32], and machine learning [33], [34] leading to a manifold increase in computational performance. To the best of our knowledge, this paper is the first work in which GPU technology is used for the solution of a stochastic optimal control problem.

There have been proposed parallelizable interior point algorithms for two-stage stochastic optimal control problems, such as [35]–[38], and an *ad hoc* interior point solver for multistage problems [39]. However, interior point algorithms involve complex steps and are not suitable for an implementation on GPUs which can make most of the capabilities of the hardware.

### C. Contributions

In this paper, we address the above-mentioned challenges by devising an optimization algorithm which makes use of the problem structure and sparsity. We exploit the structure of the problem, which is dictated by the structure of the scenario tree, to parallelize the involved operations. The algorithm runs on a GPU hardware leading to a significant speed-up as we demonstrate in Section V.

We first formulate a stochastic MPC problem using a linear flow-based hydraulic model of the water network while taking into account the uncertainty which accompanies future water demands. We propose an accelerated dual proximal gradient algorithm for the solution of the optimal control problem and report results in comparison with a CPU-based solver.

Finally, we study the performance of the closed-loop system in terms of quality of service and process economics using the Barcelona DWN as a case study. We show that the number of scenarios can be used as a tuning parameter allowing us to refine our representation of uncertainty and trade the economic operation of the network for reliability and quality of service.

### D. Mathematical Preliminaries

Let $\bar{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$ denote the set of extended-real numbers. The set of of nonnegative integers $\{k_1, k_1+1, \ldots, k_2\}, k_2 \geq k_1$ is denoted by $\mathbb{N}_{[k_1,k_2]}$. For $x \in \mathbb{R}^n$, we define $[x]_+$ to be the vector in $\mathbb{R}^n$ whose $i$th element is $\max\{0, x_i\}$. For a matrix $A \in \mathbb{R}^{n\times m}$, we denote its transpose by $A'$.

The *indicator function* of a set $C \subseteq \mathbb{R}^n$ is the extended-real valued function $\delta(\cdot|C) : \mathbb{R}^n \to \bar{\mathbb{R}}$ and it is $\delta(x|C) = 0$ for $x \in C$ and $\delta(x|C) = +\infty$ otherwise. Indicator functions are used to encode constraints in the cost function of an optimization problem. A function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is called *proper* if there is a $x \in \mathbb{R}^n$ so that $f(x) < \infty$ and $f(x) > -\infty$ for all $x \in \mathbb{R}^n$. A proper convex function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is called *lower semicontinuous* or *closed* if for every $x \in \mathbb{R}^n$, $f(x) = \liminf_{z\to x} f(z)$. For a proper closed convex function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$, we define its *conjugate* as $f^*(y) = \sup_x\{y'x - f(x)\}$. We say that $f$ is $\sigma$*-strongly convex* if $f(x) - \frac{\sigma}{2}\|x\|_2^2$ is a convex function. Unless otherwise stated, $\|\cdot\|$ stands for the Euclidean norm.

## II. Modeling of Drinking Water Networks

### A. Flow-Based Control-Oriented Model

Dynamical models of DWNs have been studied in depth the last two decades [14], [17], [40]. Flow-based models are derived from simple mass balance equations of the network which lead to the following pair of equations:

$$x_{k+1} = Ax_k + Bu_k + G_d d_k \tag{1a}$$

$$0 = Eu_k + E_d d_k \tag{1b}$$

where $x \in \mathbb{R}^{n_x}$ is the state vector corresponding to the volumes of water in the storage tanks, the manipulated inputs $u \in \mathbb{R}^{n_u}$ are the flow set-points, which are provided as references to the pumping stations and valves of the network, $d \in \mathbb{R}^{n_d}$ is the vector of water demands, and $k \in \mathbb{N}$ indices the discrete time domain. The above-mentioned dynamical model is derived on the basis of mass balances: (1a) describes the transportation of water from the tanks to the demand nodes over the network topology, while (1b) models the flow at mixing nodes. Equation (1a) forms a linear time-invariant system with additive uncertainty and (1b) is an algebraic input-disturbance coupling equation with $E \in \mathbb{R}^{n_e\times n_u}$ and $E_d \in \mathbb{R}^{n_e\times n_d}$, where $n_e$ is the number of *junctions* in the network.

The maximum capacity of the tanks, the maximum pumping capacity of each pumping station, and the limits on the flow set-points, which correspond to the valves, are described by the following bounds:

$$u_{\min} \leq u_k \leq u_{\max} \tag{2a}$$

$$x_{\min} \leq x_k \leq x_{\max}. \tag{2b}$$

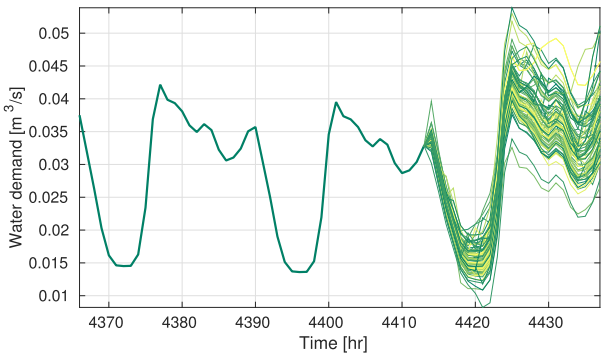


Fig. 1. Collection of possible upcoming demands at a given time instant. These results were produced using the SVM model and the data in [13].

The above-mentioned formulation has been widely used in the formulation of MPC problems for DWNs [2], [13], [17].

### B. Demand Prediction Model

The water demand is the main source of uncertainty that affects the dynamics of the network. Various time series models have been proposed for the forecasting of future water demands, such as seasonal Holt-Winters, seasonal ARIMA, BATS, and SVM [13], [41]. Such models can be used to predict nominal forecasts of the upcoming water demand along a horizon of $N$ steps ahead using measurements available up to time $k$, denoted by $\hat{d}_{k+j|k}$. Then, the actual future demands $d_{k+j}$—which are unknown to the controller at time $k$—can be expressed as

$$d_{k+j}(\epsilon_j) = \hat{d}_{k+j|k} + \epsilon_j \tag{3}$$

where $\epsilon_j$ is the demand prediction error which is a random variable on a probability space $(\Omega_j, \mathfrak{F}_j, \mathrm{P}_j)$ and for convenience, we define the tuple $\boldsymbol{\epsilon}_j = (\epsilon_0, \epsilon_1, \ldots, \epsilon_j)$, which is a random variable in the product probability space. We also define $\hat{\mathbf{d}}_k = (\hat{d}_{k|k}, \ldots \hat{d}_{k+N-1|k})$. A set of possible realizations of future water demands is shown in Fig. 1 for a demand node of the water network of Barcelona.

## III. Stochastic MPC for DWNs

In this section, we define the control objectives for the controlled operation of a DWN and we formulate the stochastic MPC problem.

### A. Control Objectives

We define the following three cost functions which reflect our control objectives. The *economic cost* quantifies the *production* and *transportation* cost

$$\ell^w(u_k, k) = W_\alpha(\alpha_1 + \alpha_{2,k})' u_k \tag{4}$$

where the term $\alpha_1' u_k$ is the water production cost, $\alpha_{2,k}' u_k$ is the (time-varying) pumping (electricity) cost, and $W_\alpha$ is a positive scaling factor. The cost for the operation of valves is negligible compared with the cost of pumping so it has not been accounted for here.

The *smooth operation cost* is defined as

$$\ell^\Delta(\Delta u_k) = \Delta u_k' W_u \Delta u_k \tag{5}$$

where $\Delta u_k = u_k - u_{k-1}$ and $W_u \in \mathbb{R}^{n_u \times n_u}$ is a symmetric positive definite weight matrix. It is introduced to penalize abrupt switching of the actuators (pumps and valves).

The *safety storage cost* penalizes the drop of water level in the tanks below a given *safety level*. An elevation above this safety level ensures that there will be enough water in unforeseen cases of unexpectedly high demand and also maintains a minimum pressure for the flow of water in the network. This is given by

$$\ell^S(x_k) = W_x \operatorname{dist}(x_k \mid \mathcal{C}_s) \tag{6}$$

where $\operatorname{dist}(x \mid \mathcal{C}) = \inf_{y \in \mathcal{C}} \|x - y\|_2$ is the distance-to-set function, $\mathcal{C}_s = \{x \mid x \geq x_s\}$, $x_s \in \mathbb{R}^{n_x}$ is the safety level, and $W_x$ is a positive scaling factor.

These cost functions have been used in many MPC formulations in the literature [13], [42]. A comprehensive discussion on the choice of these cost functions can be found in [14] and [43].

The total *stage cost* at a time instant $k$ is the summation of the above-mentioned costs and is given by

$$\ell(x_k, u_k, u_{k-1}, k) = \ell^w(u_k, k) + \ell^\Delta(\Delta u_k) + \ell^S(x_k). \tag{7}$$

### B. SMPC Formulation

We formulate the following stochastic MPC problem with a prediction horizon $N \in \mathbb{N}$, $N \geq 1$ and decision variables $\pi = \{u_{k+j|k}, x_{k+j+1|k}\}_{j \in \mathbb{N}_{[0,N-1]}}$:

$$V^\star(p, q, \hat{\mathbf{d}}_k, k) = \min_\pi \mathbb{E} V(\pi, p, q, k) \tag{8a}$$

where $\mathbb{E}$ is the expectation operator and

$$V(\pi, p, q, k) = \sum_{j=0}^{N-1} \ell(x_{k+j|k}, u_{k+j|k}, u_{k+j-1|k}, k+j) \tag{8b}$$

subject to the constraints

$$x_{k|k} = p, \quad u_{k-1|k} = q \tag{8c}$$

$$x_{k+j+1|k} = A x_{k+j|k} + B u_{k+j|k} + G_d d_{k+j|k}(\epsilon_j)$$
$$j \in \mathbb{N}_{[0,N-1]}, \ \epsilon_j \in \Omega_j \tag{8d}$$

$$E u_{k+j|k} + E_d d_{k+j|k}(\epsilon_j) = 0, \quad j \in \mathbb{N}_{[0,N-1]}, \ \epsilon_j \in \Omega_j \tag{8e}$$

$$x_{\min} \leq x_{k+j|k} \leq x_{\max}, \ j \in \mathbb{N}_{[1,N]} \tag{8f}$$

$$u_{\min} \leq u_{k+j|k} \leq u_{\max}, \ j \in \mathbb{N}_{[0,N-1]} \tag{8g}$$

where we stress out that the decision variables $\{u_{k+j|k}\}_{j=0}^{j=N-1}$ are required to be causal control laws of the form

$$u_{k+j|k} = \varphi_{k+j|k}(p, q, x_{k+j|k}, u_{k+j-1|k}, \boldsymbol{\epsilon}_j). \tag{8h}$$

Solving the above-mentioned problem would involve the evaluation of multidimensional integrals over an infinite-dimensional space which is computationally intractable. Hereafter, however, we shall assume that all $\Omega_j$, for $j \in \mathbb{N}_{[0,N-1]}$, are finite sets. This assumption will allow us to restate (8) as a finite-dimensional optimization problem.

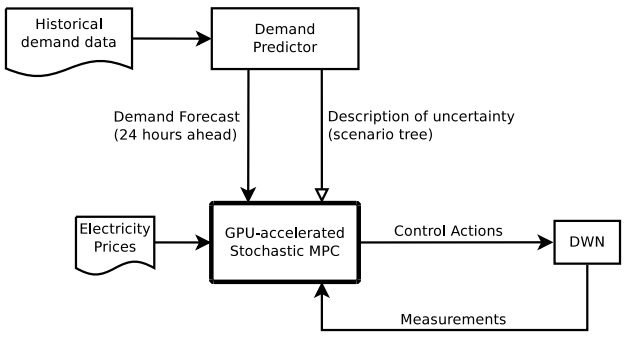


Fig. 2. Closed-loop system with the proposed stochastic MPC controller running on a GPU device.

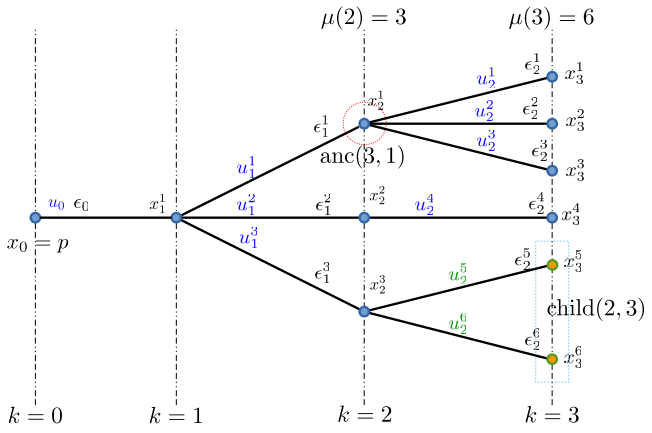


Fig. 3. Scenario tree describing the possible evolution of the system state along the prediction horizon. Future control actions are decided in a nonanticipative (causal) fashion; for example, $u_1^2$ is decided as a function of $\epsilon_1^2$ but not of any of $\epsilon_2^i$, $i \in \mathbb{N}_{[1,\mu(3)]}$.

### C. Scenario Trees

A scenario tree is the structure which naturally follows from the finiteness assumption of $\Omega_j$ and is shown in Fig. 3. A scenario tree describes a set of possible future evolutions of the state of the system known as *scenarios*. Scenario trees can be constructed algorithmically from raw data as in [30].

The nodes of a scenario tree are partitioned in *stages*. The (unique) node at stage $k = 0$ is called *root* and the nodes at the last stage are the *leaf nodes* of the tree. We denote the number of leaf nodes by $n_s$. The number of nodes at stage $k$ is denoted by $\mu(k)$ and the total number of nodes of the tree is denoted by $\mu$. A path connecting the root node with a leaf node is called a *scenario*. Nonleaf nodes define a set of *children*; at a stage $j \in \mathbb{N}_{[0,N-1]}$, for $i \in \mathbb{N}_{[1,\mu(j)]}$, the set of children of the $i$th node is denoted by $\text{child}(j,i) \subseteq \mathbb{N}_{[1,\mu(j+1)]}$. At stage $j \in \mathbb{N}_{[1,N]}$, the $i$th node $i \in \mathbb{N}_{[1,\mu(j)]}$ is reachable from a single node at stage $k-1$ known as its *ancestor*, which is denoted by $\text{anc}(j,i) \in \mathbb{N}_{[1,\mu(j-1)]}$.

The probability of visiting a node $i$ at stage $j$ starting from the root is denoted by $p_j^i$. For all for $j \in \mathbb{N}_N$, we have that $\sum_{i=1}^{\mu(j)} p_j^i = 1$ and for all $i \in \mathbb{N}_{[1,\mu(k)]}$, it is $\sum_{l \in \text{child}(j,i)} p_{j+1}^l = p_j^i$.

We define the maximum branching factor at stage $j$, $b_j$, to be the maximum number of children of the nodes at this stage. The maximum branching factor serves as a measure of the complexity of the tree at a given stage.

### D. Reformulation as a Finite-Dimensional Problem

We shall now exploit the above-mentioned tree structure to reformulate the optimal control problem (8) as a finite-dimensional problem. The water demand, given by (3), is now modeled as

$$d_{k+j|k}^i = \hat{d}_{k+j|k} + \epsilon_j^i \tag{9}$$

for all $j \in \mathbb{N}_{[0,N-1]}$ and $i \in \mathbb{N}_{[1,\mu(j+1)]}$. The input-disturbance coupling (8e) is then readily rewritten as

$$Eu_{k+j|k}^i + E_d d_{k+j|k}^i = 0 \tag{10}$$

for $j \in \mathbb{N}_{[0,N-1]}$ and $i \in \mathbb{N}_{[1,\mu(j+1)]}$.

The system dynamics is defined across the nodes of the tree by

$$x_{k+j+1|k}^l = Ax_{k+j|k}^i + Bu_{k+j|k}^l + G_d d_{k+j|k}^l \tag{11}$$

for $j \in \mathbb{N}_{[0,N-1]}$, $i \in \mathbb{N}_{[1,\mu(j)]}$, and $l \in \text{child}(j,i)$, or, alternatively

$$x_{k+j+1|k}^i = Ax_{k+j|k}^{\text{anc}(j+1,i)} + Bu_{k+j|k}^i + G_d d_{k+j|k}^i \tag{12}$$

for $j \in \mathbb{N}_{[0,N-1]}$ and $i \in \mathbb{N}_{[1,\mu(j+1)]}$.

Now, the expectation of the objective function (8b) can be derived as a summation across the tree nodes

$$\mathbb{E}V(\pi, p, q, k) = \sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} p_j^i \ell\left(x_{k+j|k}^i, u_{k+j|k}^i, u_{k+j-1|k}^{\text{anc}(j,i)}, k+j\right) \tag{13}$$

where $x_{k|k}^1 = p$ and $u_{k-1|k} = q$.

In order to guarantee the recursive feasibility of the control problem, the state constraints (8f) are converted into *soft constraints*, that is, they are replaced by a penalty of the form

$$\ell^d(x) = \gamma_d \,\text{dist}(x, \mathcal{C}_1) \tag{14}$$

where $\gamma_d$ is a positive penalty factor and $\mathcal{C}_1 = \{x \mid x_{\min} \le x \le x_{\max}\}$. Using this penalty, we construct the *soft state constraint penalty*

$$V_s(\pi, p) = \sum_{j=0}^{N} \sum_{i=1}^{\mu(j)} \ell^d\left(x_{k+j|k}^i\right). \tag{15}$$

The modified, soft-constrained, SMPC problem can be now written as

$$\tilde{V}^\star(p, q, \hat{\mathbf{d}}, k) = \min_{\pi} \mathbb{E}V(\pi, p, q, k) + V_s(\pi, p) \tag{16a}$$

$$\text{s.t. } x_{k|k}^1 = p, \; u_{k-1|k} = q \tag{16b}$$

$$u_{\min} \le u_{k+j|k}^i \le u_{\max} \quad j \in \mathbb{N}_{[0,N-1]}, i \in \mathbb{N}_{[1,\mu(j)]} \tag{16c}$$

and system equations (10) and (12).

## IV. Solution of the Stochastic Optimal Control Problem

In this section, we extend the GPU-based proximal gradient method proposed in [44] to solve the SMPC problem (16). For ease of notation, we will focus on the solution of the SMPC problem at $k = 0$ and denote $x_{j|0} = x_j$, $u_{j|0} = u_j$, and $\hat{d}_{j|0} = \hat{d}_j$.

### A. Proximal Gradient Algorithm

For a closed, proper extended-real valued function $g : \mathbb{R}^n \to \bar{\mathbb{R}}$, we define its *proximal operator* with parameter $\gamma > 0$, $\text{prox}_{\gamma g} : \mathbb{R}^n \to \mathbb{R}^n$ as [45]

$$\text{prox}_{\gamma g}(v) = \underset{x \in \mathbb{R}^n}{\arg\min} \left\{ g(x) + \frac{1}{2\gamma}\|x - v\|_2^2 \right\}. \tag{17}$$

The proximal operator of many functions is available in closed form [45], [46]. When $g$ is given in a *separable sum* form, that is

$$g(x) = \sum_{i=1}^{\kappa} g_i(x_i) \tag{18a}$$

then, for all $i \in \mathbb{N}_{[1,\kappa]}$

$$(\text{prox}_{\gamma g}(v))_i = \text{prox}_{\gamma g_i}(v_i). \tag{18b}$$

This is known as the *separable sum property* of the proximal operator.

Let $z \in \mathbb{R}^{n_z}$ be a vector encompassing all states $x_j^i$ for $j \in \mathbb{N}_{[0,N]}$ and $i \in \mathbb{N}_{[1,\mu(j)]}$ and inputs $u_j^i$ for $j \in \mathbb{N}_{[0,N-1]}$, $i \in \mathbb{N}_{[1,\mu(j+1)]}$; this is the decision variable of problem (16).

Let $f : \mathbb{R}^{n_z} \to \bar{\mathbb{R}}$ be defined as

$$\begin{aligned} f(z) = \sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} & p_j^i\big(\ell^w(u_j^i) + \ell^\Delta(\Delta u_j^i)\big) \\ & + \delta\big(u_j^i | \Phi_1(d_j^i)\big) \\ & + \delta\big(x_{j+1}^i, u_j^i, x_j^{\text{anc}(j+1,i)} | \Phi_2(d_j^i)\big) \end{aligned} \tag{19}$$

where $\Delta u_j^i = u_j^i - u_{j-1}^{\text{anc}(j,i)}$ and $\Phi_1(d)$ is the affine subspace of $\mathbb{R}^{n_u}$ induced by (10), that is

$$\Phi_1(d) = \{u : Eu + E_d d = 0\} \tag{20}$$

and $\Phi_2(d)$ is the affine subspace of $\mathbb{R}^{2n_x+n_u}$ defined by the system dynamics (12)

$$\Phi_2(d) = \{(x_{k+1}, x_k, u) : x_{k+1} = Ax_k + Bu + G_d d\}. \tag{21}$$

We define the auxiliary variable $\zeta$ which serves as a copy of the state variable $x_j^i$—that is we require $\zeta_j^i = x_j^i$. The reason for the introduction of this copy will be clarified in Section IV-C.

We introduce the variable $t = (\zeta, x, u) \in \mathbb{R}^{n_t}$ and define an extended-real valued function $g : \mathbb{R}^{n_t} \to \bar{\mathbb{R}}$ as

$$g(t) = \sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} \ell^S\big(x_{j+1}^i\big) + \ell^d\big(\zeta_{j+1}^i\big) + \delta\big(u_j^i | \mathcal{U}\big) \tag{22}$$

where $\mathcal{U} = \{u \in \mathbb{R}^{n_u} : u_{\min} \le u \le u_{\max}\}$.

Now the finite-dimensional optimization problem (16) can be written as

$$\tilde{V}^\star = \min_{z,t} f(z) + g(t) \tag{23a}$$

$$\text{s.t. } Hz = t \tag{23b}$$

where

$$H = \begin{bmatrix} I_{n_x} & 0 \\ I_{n_x} & 0 \\ 0 & I_{n_u} \end{bmatrix}. \tag{24}$$

The Fenchel dual of (23) is written as [47, Corollary 31.2.1]

$$\tilde{D}^\star = \min_y f^*(-H'y) + g^*(y) \tag{25}$$

where $y$ is the dual variable. The dual variable $y$ can be partitioned as $y = (\tilde{\zeta}_{j+1}^i, \tilde{x}_{j+1}^i, \tilde{u}_j^i)$, where $\tilde{\zeta}_{j+1}^i$, $\tilde{x}_{j+1}^i$, and $\tilde{u}_j^i$ are the dual variables corresponding to $\zeta_{j+1}^i$, $x_{j+1}^i$, and $u_j^i$, respectively. We also define the auxiliary variable $\tilde{\xi}_j^i := \tilde{\zeta}_j^i + \tilde{x}_j^i$.

According to [48, Th. 11.42], since function $f(z) + g(Hz)$ is proper, convex, and piecewise linear-quadratic, then the primal problem (23) is feasible whenever the dual problem (25) is feasible and, furthermore, strong duality holds, i.e., $\tilde{V}^\star = \tilde{D}^\star$. Moreover, the optimal solution of (23) is given by $z^\star = \nabla f^*(-H'y^\star)$, where $y^\star$ is *any* solution of (25). Applying [48, Proposition 12.60] to $f^*$ and since $f$ is lower semicontinuous, proper and $\sigma$-strongly convex—as shown at the end of Appendix A—its conjugate $f^*$ has Lipschitz-continuous gradient with a constant $1/\sigma$.

An accelerated version of proximal gradient method, which was first proposed by Nesterov [49], is applied to the dual problem. This leads to the following algorithm:

$$w^\nu = y^\nu + \theta_\nu(\theta_{\nu-1}^{-1} - 1)\big(y^\nu - y^{\nu-1}\big) \tag{26a}$$

$$z^\nu = \underset{z}{\arg\min}\, \{\langle z, H'w^\nu\rangle + f(z)\} \tag{26b}$$

$$t^\nu = \text{prox}_{\lambda^{-1}g}(\lambda^{-1}w^\nu + Hz^\nu) \tag{26c}$$

$$y^{\nu+1} = w^\nu + \lambda(Hz^\nu - t^\nu) \tag{26d}$$

$$\theta_{\nu+1} = \tfrac{1}{2}\big(\sqrt{\theta_\nu^4 + 4\theta_\nu^2} - \theta_\nu^2\big) \tag{26e}$$

starting from a dual-feasible vector $y^0 = y^{-1} = 0$ and $\theta_0 = \theta_{-1} = 1$. In (26), $\lambda > 0$ is a constant step length, which will be discussed in Section IV-D.

It is worth noting that Algorithm 26 may be interpreted as an accelerated version of [50].

In the first step (26a), we compute an extrapolation of the dual vector. In the second step (26b), we calculate the dual gradient, that is $z^\nu = \nabla f^*(-H'w^\nu)$, at the extrapolated dual vector using the conjugate subgradient theorem [47, Th. 23.5]. The third step comprises of (26c) and (26d) where we update the dual vector $y$ and in the final step of the algorithm, we compute the scalar $\theta_\nu$, which is used in the extrapolation step.

This algorithm has a convergence rate of $\mathcal{O}(1/\nu^2)$ for the dual iterates as well as for the ergodic primal iterate defined through the recursion $\bar{z}^\nu = (1 - \theta_\nu)\bar{z}^{(\nu-1)} + \theta_\nu z^\nu$, i.e., a weighted average of the primal iterates [51].

The splitting given in (23), with this particular choice of $f$ and $g$, is not unique; for example, Shapiro *et al.* [29] have proposed a dual decomposition scheme where the scenario tree is decomposed into a set of independent scenarios, while the tree structure is imposed through a linear equation of the form (23b). However, such a splitting may increase considerably the total number of variables without facilitating the implementation or offering some advantage in terms of convergence speed.

### *B. Computation of Primal Iterate*

The most critical step in the algorithm is the computation of $z^\nu$, which accounts for most of the computation time required by each iteration. This step boils down to the solution of an unconstrained optimization problem by means of dynamic programming, where certain matrices (which are independent of $w^\nu$) can be computed once before we run the algorithm to facilitate the online computations. These are: 1) the vectors $\beta_j^i, \hat{u}_j^i, e_j^i$, which are associated with the update of the time-varying cost (see Appendix A) and 2) the matrices $\Lambda, \Phi, \Psi, \bar{B}$ (see Appendix B). The latter are referred to as the *factor step* of the algorithm and matrices $\Lambda, \Phi, \Psi$, and $\bar{B}$ are independent of the complexity of the scenario tree.

The computation of $z^\nu$ at each iteration of the algorithm requires the computation of the aforementioned matrices and is computed using Algorithm 1 to which we refer as the *solve step*. Computations involved in the solve step are merely matrix-vector multiplications. As the algorithm traverses the nodes of the scenario tree stagewise backward (from stage $N-1$ to stage 0), computations across the nodes at a given stage can be performed in parallel. Hardware such as GPUs, which enable us to parallelizable such operations lead to a great speed-up as we demonstrate in Section V.

The dynamic programming approach, which gives rise to Algorithm 1, is equivalent to taking the KKT optimality conditions of (26b) and formulating the costate dynamical equations. Algorithm 1 is also reminiscent of the Riccati-type recursion in [51].

### *C. Computation of Dual Iterate*

Function $g$ given in (22) is given in the form of a separable sum

$$g(t) = g(\zeta, x, u) = g_1(\zeta) + g_2(x) + g_3(u) \tag{27}$$

where

$$g_1(\zeta) = \sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} \ell^S\left(\zeta_{j+1}^i\right) \tag{28a}$$

$$g_2(x) = \sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} \ell^d\left(x_{j+1}^i\right) \tag{28b}$$

$$g_3(u) = \sum_{j=0}^{N-1} \sum_{i=1}^{\mu(j)} \delta\left(u_j^i \mid \mathcal{U}\right). \tag{28c}$$

Functions $g_1(\cdot)$ and $g_2(\cdot)$ are in turn separable sums of distance functions from a set, and $g_3(\cdot)$ is an indicator function. Their proximal mappings can be easily computed as in Appendix C and essentially are elementwise operations on the vector $t$ that can be fully parallelized.

**Algorithm 1** Solve Step

**Require:** Output of the factor step (See Appendices A and B), i.e., $\Lambda, \Phi, \Psi, \bar{B}, \hat{u}_j^i, \beta_j^i, e_j^i$, $p, q$ and $w^\nu = (\tilde{\zeta}_{j+1}^i, \tilde{x}_{j+1}^i, \tilde{u}_j^i)$.

$q_N^i \leftarrow 0$, and $r_N^i \leftarrow 0, \forall i \in \mathbb{N}_{[1,n_s]}$,

**for** $j = N-1, \ldots, 0$ **do**

  **for** $i = 1, \ldots, \mu(k)$ **do** {in parallel}

    $\sigma_j^l \leftarrow r_{j+1}^l + \beta_j^l, \forall l \in \text{child}(j,i)$

    $v_j^l \leftarrow \frac{1}{2p_j^l}\left(\Phi_j^l(\tilde{\xi}_{j+1}^l + q_{j+1}^l) + \Psi_j^l \tilde{u}_j^l + \Lambda_j^l \sigma_j^l\right)$, $\forall l \in \text{child}(j,i)$

    $r_j^i \leftarrow \sum_{l \in \text{child}(j,i)} \sigma_j^l + \bar{B}'(\tilde{\xi}_{j+1}^l + q_{j+1}^l) + L\tilde{u}_j^l$

    $q_j^i \leftarrow A' \sum_{l \in \text{child}(j,i)} \tilde{\xi}_{j+1}^l + q_{j+1}^l$

  **end for**

**end for**

$x_0^1 \leftarrow p$, $u_{-1} \leftarrow q$,

**for** $j = 0, \ldots, N-1$ **do**

  **for** $i = 1, \ldots, \mu(k)$ **do** {in parallel}

    $v_j^i \leftarrow v_{j-1}^{\text{anc}(j,i)} + v_j^i$

    $u_j^i \leftarrow Lv_j^i + \hat{u}_j^i$

    $x_{j+1}^i \leftarrow Ax_j^{\text{anc}(j,i)} + \bar{B}v_j^i + e_j^i$

  **end for**

**end for**

**return** $\{x_j^i\}_{j=1}^N$, $\{u_j^i\}_{j=0}^{N-1}$

### *D. Preconditioning and Choice of $\lambda$*

First-order methods are known to be sensitive to scaling and preconditioning can remarkably improve their convergence rate. Various preconditioning method such as [52] and [53] have been proposed in the literature. Additionally, a parallelizable preconditioning method tailored to stochastic programs for use with interior point solvers has been proposed in [54]. Here, we employ a simple diagonal preconditioning, which consists in computing a diagonal matrix $\tilde{H}_D$ with positive diagonal entries, which approximates the dual Hessian $H_D$ and use $\tilde{H}_D^{-1/2}$ to scale the dual vector [55, 2.3.1]. Since the uncertainty does not affect the dual Hessian, we take this preconditioning matrix for a single branch of the scenario tree and use it to scale all dual variables.

In a similar way, we compute the parameter $\lambda$. We choose $\lambda = 1/L_{H_D}$, where $L_{H_D}$ is the Lipschitz constant of the dual gradient which is computed as $\|H\|^2/\sigma$ as in [55]. It again suffices to perform the computation for a single branch of the scenario tree.

### *E. Termination*

The termination conditions for the above-mentioned algorithm are based on the ones provided in [51]. However, rather than checking these conditions at every iteration, we perform always a fixed number of iterations, which is dictated by the sampling time. We may then check the quality of the solution *a posteriori* in terms of the duality gap and the term $\|Hz^\nu - t^\nu\|_\infty$.

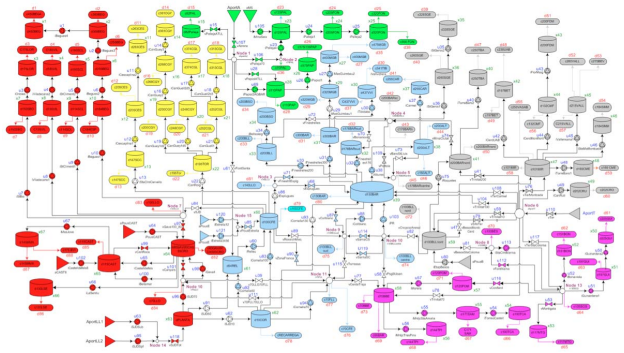

Fig. 4. Structure of the DWN of Barcelona.

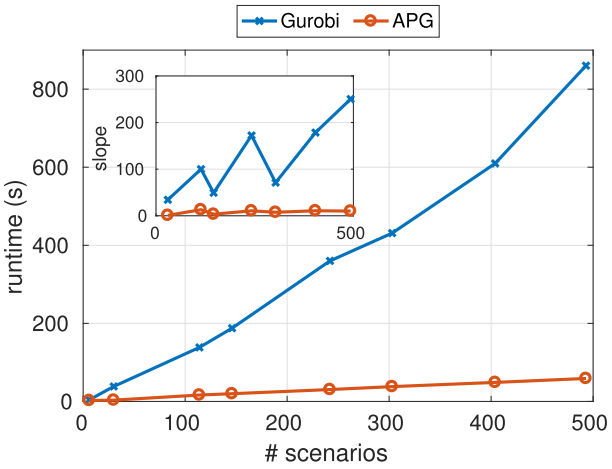


Fig. 5. Runtime of the CUDA implementation against the number of scenarios considered in the optimization problem. Comparison with the runtimes of Gurobi.

## V. Case Study: The Barcelona DWN

We now apply the proposed control methodology to the DWN of the city of Barcelona using the data provided in [2] and [13]. The topology of the network is shown in Fig. 4. The system model consists of 63 states corresponding to the level of water in each tank, 114 control inputs, which are pumping actions and valve positions, 88 demand nodes, and 17 junctions. The prediction horizon is $N = 24$ with sampling time of 1 h. The future demands are predicted using the SVM time series model developed in [13].

In our subsequent analysis, the initial state of the system, $x_0 = p$, is selected uniformly randomly between $x_s$ and $x_{\max}$.

### A. Performance of GPU-Accelerated Algorithm

The accelerated proximal gradient (APG) algorithm was implemented in CUDA-C v6.0 and the matrix-vector computations were performed using cuBLAS. We compared the GPU-based implementation with the interior point solver of Gurobi. Active-set algorithms exhibited very poor performance and we did not include the respective results.

All computations on CPU were performed on a $4\times2.60$GHz Intel i5 machine with 8GB of RAM running 64-b Ubuntu v14.04 and GPU-based computations were carried out on an NVIDIA Tesla C2075.

The relative tolerance of Gurobi was set to $2\cdot10^{-2}$ instead of the default tolerance of $10^{-8}$. The dependence of the computation time on the size of the scenario tree is reported in Fig. 5, where it can be noticed that APG running on GPU compared with Gurobi is 10 to 25 times faster. Furthermore, the speed-up increases with the number of scenarios as we may tell by looking at Fig. 5 (inset). The runtimes shown are averages over 100 random initial points $x_0$.

The optimization problems we are solving here are of noticeably large size. Indicatively, the scenario tree with 493 scenarios counts approximately 2.52 million dual decision variables (1.86 million primal variables), and while Gurobi requires 860 s to solve it, our CUDA implementation solves it in 58.8 s; this corresponds to a speed-up of 14.6.

In all of our simulations, we obtained a sequence of control actions across the tree nodes $U^\star_{apg} = \{u^i_j\}$ which was, elementwise, within $\pm0.029$ m$^3$/s (1.9%) of the solution produced by Gurobi with relative tolerance $10^{-8}$. Moreover, we should note that the control action $u^\star_0$ computed by APG with 500 iterations was consistently within $\pm0.0025$ m$^3$/s (0.08%) of the Gurobi solution. Given that only $u^\star_0$ is applied to the system, while all other control actions $u^i_j$ for $j \in \mathbb{N}_{[1,N-1]}$ and $i \in \mathbb{N}_{[1,\mu(j)]}$ are discarded, 500 iterations are well sufficient for convergence.

### B. Closed-Loop Performance

In this section, we analyze the performance of SMPC with different scenario-trees. This analysis is carried for a period of 7 days ($H_s = 168$) from July 1, 2007 to July 8, 2007. Here, we compare the operational cost and the quality of service of various scenario-tree structures.

The weighting matrices in the operational cost are chosen as $W_\alpha = 2\cdot10^4$, $W_u = 10^5\cdot I$, and $W_x = 10^7$, respectively, and $\gamma_d = 5\cdot10^7$. The demand is predicted using the SVM model presented in [13]. The steps involved in SMPC using GPU-based APG in closed-loop is summarized in Algorithm 2.

**Algorithm 2** Closed-Loop of DWN With SMPC With Proximal Operator

**Require:** Scenario tree, current state measurement $x_0$ and previous control $u_{-1}$.
Compute $\Lambda$, $\Phi$, $\Psi$ and $\bar{B}$ as in Appendix B
Precondition the original optimization problem and compute $\lambda$ as in Section IV-D.
**loop**
*Step 1.* Predict the future water demands $\hat{\mathbf{d}}_k$ using current and past demand data.
*Step 2.* Compute $\hat{u}^i_j$, $\beta^i_j$, $e^i_j$ as in Appendix A.
*Step 3.* Solve the optimization problem using APG on GPU using iteration (26) and Algorithm 1.
*Step 4.* Apply $u^1_0$ to the system, update $u_{-1} = u^1_0$
**end loop**

For the performance assessment of the proposed control methodology, we used various controllers summarized

TABLE I

VARIOUS CONTROLLERS USED TO ASSESS THE CLOSED-LOOP PERFORMANCE OF THE PROPOSED METHODOLOGY. THE NUMBERS IN THE BRACKETS DENOTE THE FIRST MAXIMUM BRANCHING FACTORS, $b_j$, OF THE SCENARIO TREE, WHILE ALL SUBSEQUENT BRANCHING FACTORS ARE EQUAL TO 1

| Controller | $b_k$ | scenarios | primal variables | dual variables |
|---|---|---|---|---|
| CE-MPC | 1 | 1 | 4248 | 5760 |
| $\text{SMPC}_1$ | $[3, 2]$ | 6 | 24072 | 32540 |
| $\text{SMPC}_2$ | $[6, 5]$ | 30 | 118059 | 160080 |
| $\text{SMPC}_3$ | $[6, 5, 5]$ | 114 | 430287 | 583440 |
| $\text{SMPC}_4$ | $[8, 5, 5]$ | 146 | 551355 | 747600 |
| $\text{SMPC}_5$ | $[10, 8, 5]$ | 242 | 915621 | 1241520 |
| $\text{SMPC}_6$ | $[12, 8, 5]$ | 303 | 1145544 | 1553280 |
| $\text{SMPC}_7$ | $[12, 8, 8]$ | 404 | 1520961 | 2062320 |
| $\text{SMPC}_8$ | $[12, 10, 8]$ | 493 | 1856022 | 2516640 |

in Table I. The corresponding computation times are presented in Fig. 5.

To assess the performance of closed-loop operation of the SMPC-controlled network, we used the *key performance indicators* (KPIs) reported in [2], [3], and [56]. For a simulation time length $H_s$, the performance indicators are computed by

$$\text{KPI}_E = \frac{1}{H_s}\sum_{k=1}^{H_s}(\alpha_1 + \alpha_{2,k})'|u_k| \tag{29a}$$

$$\text{KPI}_{\Delta U} = \frac{1}{H_s}\sum_{k=1}^{H_s}\|\Delta u_k\|^2 \tag{29b}$$

$$\text{KPI}_S = \sum_{k=1}^{H_s}\|[x_s - x_k]_+\|_1 \tag{29c}$$

$$\text{KPI}_R = \frac{\|x_s\|_1}{\frac{1}{H_s}\sum_{k=1}^{H_s}\|x_k\|_1}\cdot 100\%. \tag{29d}$$

$\text{KPI}_E$ is the average economic cost, $\text{KPI}_{\Delta U}$ measures the average smoothness of the control actions, $\text{KPI}_S$ corresponds to the total amount of water used from storage, that is the amount of water that is removed from a tank when the volume in that tank is below the safe level $x_s$, and $\text{KPI}_R$ is the percentage of the safety volume $x_s$ contained into the average volume of water.

*1) Risk Versus Economic Utility:* Fig. 6 shows the tradeoff between economic and safe operation: The more scenarios we use to describe the distribution of demand prediction error, the safer the closed-loop operation becomes as it is reflected by the decrease of $\text{KPI}_S$. Stochastic MPC leads to a significant decrease of economic cost compared with the certainty-equivalence approach; however, the safer we require the operation to be, the higher the operating cost we should expect. For example, if the designer opts for 30 scenarios, they will have struck a low operating cost which, nevertheless, comes with a high value of $\text{KPI}_S$, that is, operation under high risk. In order to decrease this risk, one needs to consider a higher number of scenarios which comes at a higher operating cost. We may also observe that for too few scenarios, the operation of the network will be both expensive and will incur a rather high risk.

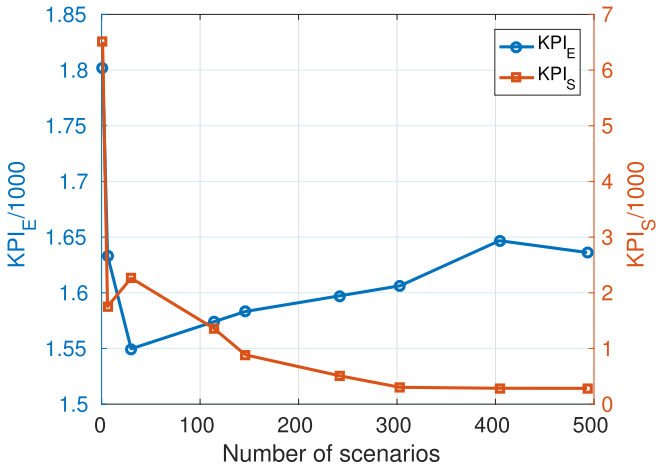


Fig. 6. Tradeoff between risk and economic utility in terms of scenarios. $\text{KPI}_E$ represent the economical utility and $\text{KPI}_S$ shows the risk of violation.

TABLE II

KPIS FOR PERFORMANCE ANALYSIS OF THE DWN WITH DIFFERENT CONTROLLERS. THE LOWEST AND THE HIGHEST VALUES IN EACH COLUMN ARE HIGHLIGHTED

| Controller | $\text{KPI}_E$ | $\text{KPI}_{\Delta U}$ | $\text{KPI}_S$ | $\text{KPI}_R$ |
|---|---|---|---|---|
| CE-MPC | **1801.4** | 0.2737 | **6507.7** | 64.89% |
| $\text{SMPC}_1$ | 1633.5 | 0.3896 | 1753.7 | **67.96%** |
| $\text{SMPC}_2$ | **1549.7** | 0.4652 | 2264.0 | 61.81% |
| $\text{SMPC}_3$ | 1574.0 | 0.4135 | 1360.0 | 49.65% |
| $\text{SMPC}_4$ | 1583.2 | 0.4088 | 885.7 | 48.13% |
| $\text{SMPC}_5$ | 1597.3 | 0.4470 | 508.5 | 46.05% |
| $\text{SMPC}_6$ | 1606.3 | 0.4878 | 302.3 | **44.93%** |
| $\text{SMPC}_7$ | 1646.8 | 0.5189 | 285.4 | 47.89% |
| $\text{SMPC}_8$ | 1636.2 | 0.5068 | **283.9** | 47.41% |

*2) Quality of Service:* A measure of the reliability and quality of service of the network is $\text{KPI}_S$, which reflects the tendency of water levels to drop under the safety storage levels. As expected, the CE-MPC controller leads to the most unsafe operation, whereas $\text{SMPC}_8$ leads to the lowest value.

*3) Network Utility:* Network utility is defined as the ability to utilize the water in the tanks to meet the demands rather than pumping additional water and is quantified by $\text{KPI}_R$. In Table II, we see the dependence of $\text{KPI}_R$ on the number of scenarios of the tree. The decrease in $\text{KPI}_R$ one may observe is because the more scenarios are introduced, the more accurate the representation of uncertainty becomes and the system does not need to operate, on average, too far away from $x_s$. Of course, when operating close to $x_s$, we need to take into account the value of $\text{KPI}_S$ to check whether the operation violates the safety storage limit.

*4) Smooth Operation:* We may notice that the introduction of more scenarios results in an increase in $\text{KPI}_{\Delta U}$. Then, the controller becomes more responsive to accommodate the need for a less risky operation, although the value of $\text{KPI}_{\Delta U}$ is not greatly affected by the number of scenarios.

In Fig. 7, we show two pumping actions during 168 h (1 week) of operation. We may observe the tendency of the controller to be thrifty with pumping when the corresponding pumping cost is high.

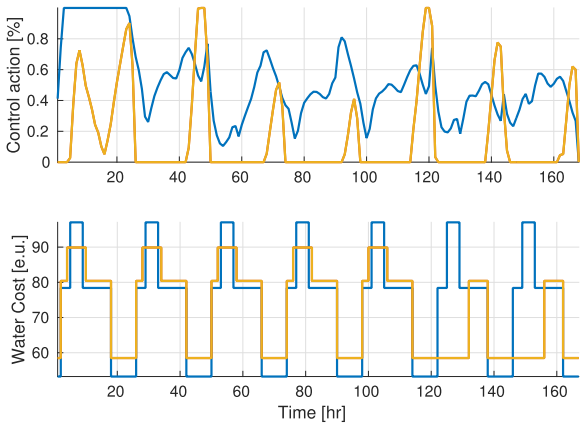


Fig. 7. Pumping actions using $\text{SMPC}_4$ (expressed in % of $u_{\max}$) and the corresponding weighted time-varying cost $W_\alpha \alpha_{2,k}$ in economic units.

### C. Implementation Details

At every time instant $k$, we need to load onto the GPU the state measurement and a sequence of demand predictions (see Fig. 2), that is $\hat{\mathbf{d}}_k$. This amounts to 8.4 kB and is rapidly uploaded on the GPU (less than 0.034 ms). In case we need to update the scenario-tree values, that is $\boldsymbol{\epsilon}_k$, and for the case of $\text{SMPC}_8$, we need to upload 3.52 MB which is done in 3.74 ms. Therefore, the time needed to load these data on the GPU is not a limiting factor.

The parallelization of matrix-vector multiplications required in Algorithm 1 are implemented using method `cublasSgemmBatched` of cuBLAS. Vector additions are performed using `cublasSaxpy` and summations over the set of children of a node were done using a custom kernel.

Compared with an MATLAB implementation of APG, the presented GPU implementation was found to be 60 times faster for $\text{SMPC}_7$ and 95 times faster for $\text{SMPC}_8$. However, we believe that the comparison to an MATLAB implementation is not totally fair and a comparison to an optimized C implementation is beyond the scope of this paper. It is evident that GPU technology enables very high speed-ups, notwithstanding. The source code of our implementation is publicly available with an LGPL v3 license at https://github.com/ajaykumarsampath/GPU- APG-water-network.

## VI. Conclusion and Future Work

In this paper, we have presented a framework for the formulation of an SMPC problem for the operational management of DWNs and we have proposed a novel approach for the efficient numerical solution of the associated optimization problem on a GPU. The proposed algorithm achieves remarkably high speed as we fully exploit the structure of the optimization problem and we parallelize its execution to a large extent. At the same time, it involves only matrix-vector operations and it is easy to implement.

We demonstrated the computational feasibility of the algorithm and the benefits for the operational management of the system in terms of performance (which we quantified using certain KPIs from the literature). Given that the proposed algorithmic scheme can solve linear stochastic optimal control problems so fast, future research should focus on the use of nonlinear dynamical models accounting for the pressure drops across the network. We believe that the algorithmic developments presented in this paper pave the way for the application of SMPC methodologies to DWNs.

## Appendix A
## Elimination of Input-Disturbance Coupling

In this section, we discuss how the input-disturbance equality constraints can be eliminated by a proper change of input variables and we compute the parameters $\beta_j^i, \hat{u}_j^i, e_j^i$ $\forall i \in \mathbb{N}_{[1,\mu(j)]}$, $j \in \mathbb{N}_{[0,N]}$, which are then provided as input to Algorithm 1. These depend on the nominal demand forecasts $\hat{d}_{k+j|k}$ and on the time-varying economic cost parameters $\alpha_{2,k+j}$ for $j \in \mathbb{N}_{[0,N-1]}$, therefore, they need to be updated at every time instant $k$.

The affine space $\Phi_1(d)$ introduced in (20) can be written as

$$\Phi_1(d) = \{v \in \mathbb{R}^{n_v} : u = Lv + \hat{u}(d)\} \tag{30}$$

where $L \in \mathbb{R}^{n_u \times n_v}$ is a *full rank* matrix whose range spans the nullspace of $E$, i.e., for every $v \in \mathbb{R}^{n_v}$, we have $Lv$ is in the kernel of $E$ and $\hat{u}(d)$ satisfies $E\hat{u}(d) + E_d d = 0$.

Substituting $u_j^i = Lv_j^i + \hat{u}_j^i$, $\forall i \in \mathbb{N}_{[1,\mu(j)]}$, $j \in \mathbb{N}_{[0,N]}$ in the manifold defined by the system dynamics $\Phi_2(d)$ as in (21) gives

$$\begin{aligned}\Phi_2(d) = \{(x_{j+1}, x_j, v) : x_{j+1} = Ax_j + \bar{B}v + e,\\ \bar{B} = BL, e = B\hat{u} + G_d d\}\end{aligned} \tag{31}$$

and we define

$$e_j^i = B\hat{u}_j^i + G_d d_j^i. \tag{32}$$

Now the cost in (19) is transformed as

$$\begin{aligned}&\sum_{j=0}^{N-1}\sum_{i=1}^{\mu(j)} p_j^i\left(\ell^w(u_j^i) + \ell^\Delta(\Delta u_j^i)\right)\\ &\quad= \sum_{j=0}^{N-1}\sum_{i=1}^{\mu(j)} p_j^i\left(\ell^w(v_j^i) + \ell^\Delta(\Delta v_j^i, \hat{u}_j^i)\right)\end{aligned} \tag{33}$$

where

$$\hat{R} = W_u L \tag{34a}$$

$$\bar{R} = L'\hat{R} \tag{34b}$$

$$\bar{\alpha}_j = W_\alpha(\alpha_1 + \alpha_{2,j+k})L \tag{34c}$$

$$\ell^w(v_j^i) = \bar{\alpha}_j' v_j^i \tag{34d}$$

$$\Delta v_j^i = v_j^i - v_{j-1}^{\text{anc}(j,i)} \tag{34e}$$

$$\Delta \hat{u}_j^i = \hat{u}_j^i - \hat{u}_{j-1}^{\text{anc}(j,i)} \tag{34f}$$

$$\ell^\Delta(\Delta v_j^i, \Delta \hat{u}_j^i) = \Delta v_j^i \bar{R} \Delta v_j^i + 2\Delta \hat{u}_j^{i\prime} \hat{R} \Delta v_j^i. \tag{34g}$$

By substituting and expanding $\Delta v_j^i$ and $\Delta \hat{u}_j^i$ in $\ell^\Delta(\Delta v_j^i, \Delta \hat{u}_j^i)$ the cost in (34g) becomes

$$\sum_{j=0}^{N-1}\sum_{i=1}^{\mu(j)} p_j^i\big(\ell^w\big(u_j^i\big)+\ell^\Delta\big(\Delta u_j^i\big)\big) = \sum_{j=0}^{N-1}\sum_{i=1}^{\mu(j)} \bar{p}_j^i v_j^{i\prime}\bar{R}v_j^i - 2p_j^i v_{j-1}^{\mathrm{anc}(j,i)\prime}\bar{R}v_j^i + \beta_j^{i\prime}v_j^i \quad (35)$$

where

$$\bar{p}_j^i = p_j^i + \sum_{l\in\mathrm{child}(j,i)} p_{j+1}^l \quad (36a)$$

$$\beta_j^i = p_j^i\bar{\alpha}_j + 2p_j^i\hat{R}\left(\bar{p}_j^i\hat{u}_j^i - \hat{u}_{j-1}^{\mathrm{anc}(j,i)} - \sum_{l\in\mathrm{child}(j,i)} p_{j+1}^l\hat{u}_{j+1}^l\right). \quad (36b)$$

Now $\hat{u}_j^i$, $e_j^i$, and $\beta_j^i$ are calculated by (30), (32), and (36b), respectively. Using our assumption that $L$ is full rank, we can see that $\bar{R}$ is a positive definite and symmetric matrix, therefore, $f$ is strongly convex.

## Appendix B
## Factor Step

Algorithm 1 solves the unconstrained minimization problem (26b), that is

$$z^\star = \arg\min_z \{\langle z, H'y\rangle + f(z)\} \quad (37)$$

where $z = \{x_{j+1}^i, u_j^i\}$, $y = \{\tilde{\zeta}_{j+1}^i, \tilde{x}_{j+1}^i, \tilde{u}_j^i\}$ for $i \in \mathbb{N}_{[1,\mu(j)]}$ and $j \in \mathbb{N}_{[0,N-1]}$, $f(z)$ is given by (19) and $H$ is given by (24). Substituting $H$ the optimization problem becomes

$$\begin{aligned} z^\star = \arg\min_z \sum_{j=0}^{N-1}\sum_{i=1}^{\mu(j)} & p_j^i\big(\ell^w\big(u_j^i\big)+\ell^\Delta\big(\Delta u_j^i\big)\big) \\ & + \tilde{\xi}_{j+1}^{i\prime}x_{j+1}^i + \tilde{u}_j^{i\prime}u_j^i + \delta\big(u_j^i|\Phi_1\big(d_j^i\big)\big) \\ & + \delta\big(x_{j+1}^i, u_j^i, x_j^{\mathrm{anc}(j+1,i)}|\Phi_2\big(d_j^i\big)\big) \end{aligned} \quad (38)$$

where $\tilde{\xi}_j^i := \tilde{x}_j^i + \tilde{\zeta}_j^i$.

The input-disturbance coupling constraints imposed by $\delta(u_j^i|\Phi_1(d_j^i))$ in the above-mentioned problem are eliminated as discussed in Appendix A. This changes the input variable from $u_j^i$ to $v_j^i$ given by (30) and the cost function as in (35). We, therefore, replace the decision variable $z$ with $\bar{z} := \{x_j^i, v_j^i\}$ and the optimization problem (38) reduces to

$$\begin{aligned} \bar{z}^\star = \arg\min_{\bar{z}} \sum_{j=0}^{N-1}\sum_{i=1}^{\mu(j)} & \bar{p}_j^i v_j^{i\prime}\bar{R}v_j^i - 2p_j^i v_{j-1}^{\mathrm{anc}(j,i)\prime}\bar{R}v_j^i \\ & + \beta_j^{i\prime}v_j^i + \tilde{\xi}_{j+1}^{i\prime}x_{j+1}^j + \tilde{u}_j^{i\prime}Lv_j^i \\ & + \delta\big(x_{j+1}^i, v_j^i, x_j^{\mathrm{anc}(k,i)}\big|\Phi_2\big(d_j^i\big)\big) \end{aligned} \quad (39)$$

where $u_j^i = Lv_j^i + \hat{u}_j^i$.

The above-mentioned problem is an unconstrained optimization problem with quadratic stage cost which is solved using dynamic programming [57]. This method transforms the complex problem into a sequence of subproblems solved at each stage.

Using dynamic programming, we find that the transformed control actions $v_j^{i\star}$ have to satisfy the recursive formula

$$v_j^{i\star} = v_{j-1}^{\mathrm{anc}(j,i)} + \frac{1}{2p_j^i}\big(\Phi\big(\tilde{\xi}_{j+1}^i + q_{j+1}^i\big) + \Psi\tilde{u}_j^i + \Lambda(\beta_j^i + r_{j+1}^i)\big) \quad (40)$$

where

$$\Lambda = -\bar{R}^{-1} \quad (41a)$$

$$\Phi = \Lambda\bar{B}' \quad (41b)$$

$$\Psi = \Lambda L. \quad (41c)$$

Matrix $\bar{R}$ is symmetric and positive definite; therefore, we can compute once its Cholesky factorization so that we obviate the computation of its inverse.

$q_{j+1}^i$ and $r_{j+1}^i$ in (40) correspond to the linear cost terms in the cost-to-go function at node $i$ of stage $j+1$. At stage $j$, these terms are updated by substituting $v_j^{i\star}$ as

$$r_j^s = \sum_{l\in\mathrm{child}(j-1,s)} \sigma_j^l + \bar{B}'\big(\tilde{\xi}_{j+1}^l + q_{j+1}^l\big) + L\tilde{u}_j^l \quad (42a)$$

$$q_j^s = A'\sum_{l\in\mathrm{child}(j-1,s)} \tilde{\xi}_{j+1}^l + q_{j+1}^l \quad (42b)$$

where $s = \mathrm{anc}(j,i)$.

Equations (40) and (42) form the solve step as in Algorithm 1. Matrices $\Lambda$, $\Phi$, and $\Psi$ are required to be computed once.

## Appendix C
## Proximal Operators

Function $g$ in (27) is a separable sum of distance and indicator functions and its proximal is computed according to (18). The proximal operator of the indicator of a convex closed set $C$, that is

$$\chi_C(x) = \begin{cases} 0, & \text{if } x \in \mathcal{C} \\ +\infty, & \text{otherwise} \end{cases} \quad (43)$$

is the projection operator onto $C$, that is

$$\mathrm{prox}_{\lambda\chi_C}(v) = \mathrm{proj}_{\mathcal{C}}(v) = \arg\min_{y\in C} \|v - y\|. \quad (44)$$

When $g$ is the distance function from a convex closed set $C$, that is

$$\begin{aligned} g(x) &= \mu\,\mathrm{dist}(x \mid \mathcal{C}) = \inf_{y\in\mathcal{C}} \mu\|x-y\| \\ &= \mu\|x - \mathrm{proj}_{\mathcal{C}}(x)\| \end{aligned} \quad (45)$$

then proximal operator of $g$ given by [46]

$$\mathrm{prox}_{\lambda g}(v) = \begin{cases} v + \dfrac{\mathrm{proj}_{\mathcal{C}}(v) - v}{\mathrm{dist}(v \mid \mathcal{C})}, & \text{if } \mathrm{dist}(v \mid \mathcal{C}) > \lambda\mu \\ \mathrm{proj}_{\mathcal{C}}(v), & \text{otherwise.} \end{cases} \quad (46)$$

## REFERENCES


[1] V. Havlena, P. Trnka, and B. Sheridan, "Management of complex water networks," in *The Impact of Control Technology*, T. Samad and A. Annaswamy, Eds., 2nd ed. IEEE Control Systems Society, 2014. [Online]. Available: http//:www.ieeecss.org

[2] J. Grosso, C. Ocampo-Martínez, V. Puig, and B. Joseph, "Chance-constrained model predictive control for drinking water networks," *J. Process Control*, vol. 24, no. 5, pp. 504–516, 2014.

[3] J. Grosso, J. Maestre, C. Ocampo-Martinez, and V. Puig, "On the assessment of tree-based and chance-constrained predictive control approaches applied to drinking water networks," in *Proc. 19th Conf. (IFAC)*, Cape town, South Africa, Aug. 2014, pp. 6240–6245.

[4] C. Hans, P. Sopasakis, A. Bemporad, J. Raisch, and C. Reincke-Collon, "Scenario-based model predictive operation control of islanded microgrids," in *Proc. 54th IEEE Conf. Decision Control*, Osaka, Jpn, Dec. 2015, pp. 3272–3277.

[5] P. Sopasakis, D. Herceg, P. Patrinos, and A. Bemporad. (Oct. 2016). "Stochastic economic model predictive control for Markovian switching systems." [Online]. Available: https://arxiv.org/abs/1610.10014

[6] P. Patrinos, P. Sopasakis, H. Sarmiveis, and A. Bemporad, "Stochastic model predictive control for constrained discrete-time Markovian switching systems," *Automatica*, vol. 50, pp. 2504–2514, Oct. 2014.

[7] A. Nemirovski and A. Shapiro, "Convex approximations of chance constrained programs," *SIAM J. Opt.*, vol. 17, no. 4, pp. 969–996, 2006.

[8] V. Nitivattananon, E. C. Sadowski, and R. G. Quimpo, "Optimization of water supply system operation," *J. Water Resour. Planning. Manage.*, vol. 122, pp. 374–384, Sep. 1996.

[9] U. Zessler and U. Shamir, "Optimal operation of water distribution systems," *J. Water Resour. Planning Manage.*, vol. 115, no. 6, pp. 735–752, 1989.

[10] G. Yu, R. Powell, and M. Sterling, "Optimized pump scheduling in water distribution systems," *J. Optim. Theory Appl.*, vol. 83, no. 3, pp. 463–488, 1994.

[11] A. Bagirov *et al.*, "An algorithm for minimization of pumping costs in water distribution systems using a novel approach to pump scheduling," *Math. Comput. Model.*, vol. 57, nos. 3–4, pp. 873–886, 2013.

[12] G. McCormick and R. S. Powell, "Derivation of near-optimal pump schedules for water distribution by simulated annealing," *J. Oper. Res. Soc.*, vol. 55, no. 7, pp. 728–736, 2004.

[13] A. Sampathirao, J. Grosso, P. Sopasakis, C. Ocampo-Martinez, A. Bemporad, and V. Puig, "Water demand forecasting for the optimal operation of large-scale drinking water networks: The Barcelona case study," in *Proc. 19th World Congr. (IFAC)*, 2014, pp. 10457–10462.

[14] C. Ocampo-Martinez, V. Puig, G. Cembrano, R. Creus, and M. Minoves, "Improving water management efficiency by using optimization-based control strategies: The Barcelona case study," *Water Sci. Technol. Water Supply*, vol. 9, no. 5, pp. 565–575, 2009.

[15] M. Bakker, J. H. G. Vreeburg, L. J. Palmen, V. Sperber, G. Bakker, and L. C. Rietveld, "Better water quality and higher energy efficiency by using model predictive flow control at water supply systems," *J. Water Supply, Res. Technol. Aqua*, vol. 62, no. 1, pp. 1–13, 2013.

[16] S. Leirens, C. Zamora, R. Negenborn, and B. De Schutter, "Coordination in urban water supply networks using distributed model predictive control," in *Proc. Amer. Control Conf. (ACC)*, Baltimore, MD, USA, Jun. 2010, pp. 3957–3962.

[17] C. Ocampo-Martinez, V. Fambrini, D. Barcelli, and V. Puig, "Model predictive control of drinking water networks: A hierarchical and decentralized approach," in *Proc. Amer. Control Conf. (ACC)*, Baltimore, MD, USA, Jun. 2010, pp. 3951–3956.

[18] G. S. Sankar, S. M. Kumar, S. Narasimhan, S. Narasimhan, and S. M. Bhallamudi, "Optimal control of water distribution networks with storage facilities," *J. Process Control*, vol. 32, pp. 127–137, Apr. 2015.

[19] V. Tran and M. Brdys, "Optimizing control by robustly feasible model predictive control and application to drinking water distribution systems," in *Artificial Neural Networks—ICANN* (Lecture Notes in Computer Science), vol. 5769, C. Alippi, M. Polycarpou, C. Panayiotou, and G. Ellinas, Eds. Berlin, Germany: Springer, 2009, pp. 823–834.

[20] A. Goryashko and A. Nemirovski, "Robust energy cost optimization of water distribution system with uncertain demand," *Autom. Remote Control*, vol. 75, no. 10, pp. 1754–1769, 2014.

[21] J. D. Watkins and D. McKinney, "Finding robust solutions to water resources problems," *J. Water Resour. Planning Manage.*, vol. 123, no. 1, pp. 49–58, 1997.

[22] M. Cannon, B. Kouvaritakis, and X. Wu, "Probabilistic constrained MPC for multiplicative and additive stochastic uncertainty," *IEEE Trans. Autom. Control*, vol. 54, no. 7, pp. 1626–1632, Jul. 2009.

[23] D. Bernardini and A. Bemporad, "Stabilizing model predictive control of stochastic constrained linear systems," *IEEE Trans. Autom. Control*, vol. 57, no. 6, pp. 1468–1480, Jun. 2012.

[24] D. van Hessem and O. Bosgra, "A conic reformulation of model predictive control including bounded and stochastic disturbances under state and input constraints," in *Proc. 41st IEEE Conf. Decision Control*, vol. 4. Las Vegas, NV, USA, Dec. 2002, pp. 4643–4648.

[25] D. Bertsimas and D. B. Brown, "Constrained stochastic LQC: A tractable approach," *IEEE Trans. Autom. Control*, vol. 52, no. 10, pp. 1826–1841, Oct. 2007.

[26] G. Calafiore and M. Campi, "The scenario approach to robust control design," *IEEE Trans. Autom. Control*, vol. 51, no. 5, pp. 742–753, May 2006.

[27] M. C. Campi, S. Garatti, and M. Prandini, "The scenario approach for systems and control design," *Annu. Rev. Control*, vol. 33, no. 2, pp. 149–157, 2009.

[28] M. Prandini, S. Garatti, and J. Lygeros, "A randomized approach to stochastic model predictive control," in *Proc. IEEE 51st Annu. Conf. Decision Control (CDC)*, Dec. 2012, pp. 7315–7320.

[29] A. Shapiro, D. Dentcheva, and A. Ruszczynski, *Lectures on Stochastic Programming: Modeling and Theory*. Philadelphia, PA, USA: SIAM, 2009.

[30] H. Heitsch and W. Römisch, "Scenario tree modeling for multistage stochastic programs," *Math. Program.*, vol. 118, no. 2, pp. 371–406, 2009.

[31] M. D. Mccool, "Signal processing and general-purpose computing and GPUs [exploratory DSP]," *IEEE Signal Process. Mag.*, vol. 24, no. 3, pp. 109–114, May 2007.

[32] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool, "Pedestrian detection at 100 frames per second," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 2903–2910.

[33] H. Jang, A. Park, and K. Jung, "Neural network implementation using CUDA and OpenMP," in *Proc. Digital Image Comput. Techn. Appl. (DICTA)*, Dec. 2008, pp. 155–161.

[34] A. Guzhva, S. Dolenko, and I. Persiantsev, "Multifold Acceleration of Neural Network Computations Using GPU," in *Proc. 19th Int. Conf. Artif. Neural Netw. (ICANN)*, Limassol, Cyprus, Sep. 2009, pp. 373–380.

[35] H. Jang, A. Park, and K. Jung, "Neural network implementation using CUDA and OpenMP," in *Proc. Digital Image Computing: Techn. Appl.*, Dec. 2008, pp. 155–161.

[36] M. Lubin, C. G. Petra, M. Anitescu, and V. Zavala, "Scalable stochastic optimization of complex energy systems," in *Proc. Int. Conf. High Perform. Comput. Netw., Storage Anal.*, New York, NY, USA, 2011, pp. 64:1–64:64.

[37] J. Kang, Y. Cao, D. P. Word, and C. D. Laird, "An interior-point method for efficient solution of block-structured NLP problems using an implicit Schur-complement decomposition," *Comput. Chem. Eng.*, vol. 71, pp. 563–573, Dec. 2014.

[38] N. Chiang, C. G. Petra, and V. M. Zavala, "Structured nonconvex optimization of large-scale energy systems using PIPS-NLP," in *Proc. Power Syst. Comput. Conf. (PSCC)*, Aug. 2014, pp. 1–7.

[39] J. Hübner, M. Schmidt, and M. C. Steinbach, "A distributed interior-point KKT solver for multistage stochastic optimization," Tech. Rep., Sep. 2016. [Online]. Available: http://www.optimizationonline.org/DB_HTML/2016/02/5318.html.

[40] S. Miyaoka and M. Funabashi, "Optimal control of water distribution systems by network flow theory," *IEEE Trans. Autom. Control*, vol. 29, no. 4, pp. 303–311, Apr. 1984.

[41] Y. Wang, C. Ocampo-Martinez, V. Puig, and J. Quevedo, "Gaussian-process-based demand forecasting for predictive control of drinking water networks," in *Proc. 9th Int. Conf. Critical Inf. Infrastruct. Secur. (CRITIS)*, 2014, pp. 69–80.

[42] S. Cong Cong, S. Puig, and G. Cembrano, "Combining CSP and MPC for the operational control of water networks: Application to the richmond case study," in *Proc. 19th World Congr. (IFAC)*, 2014, pp. 6246–6251.

[43] C. Ocampo-Martinez and R. Negenborn, *Transport of Water Versus Transport Over Water : Exploring the Dynamic Interplay of Transport and Water*. Cham, Switzerland: Springer, 2015.

[44] A. Sampathirao, P. Sopasakis, A. Bemporad, and P. Patrinos, "Distributed solution of stochastic optimal control problems on GPUs," in *Proc. 54th IEEE Conf. Decision Control*, Osaka, Jpn., Dec. 2015, pp. 7183–7188.

[45] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, Jan. 2014.

[46] P. Combettes and J.-C. Pesquet. (Dec. 2010). "Proximal splitting methods in signal processing." [Online]. Available: https://arxiv.org/abs/0912.3522
[47] R. Rockafellar, *Convex Analysis*. Princeton, NJ, USA: Princeton Univ. Press, 1972.
[48] R. Rockafellar and J. Wets, *Variational Analysis*, 3rd ed. Berlin, Germany: Springer-Verlag, 2009.
[49] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Soviet Math. Doklady*, vol. 72, no. 2, pp. 372–376, 1983.
[50] P. Tseng, "Applications of a splitting algorithm to decomposition in convex programming and variational inequalities," *SIAM J. Control Optim.*, vol. 29, pp. 119–138, Jan. 1991.
[51] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control," *IEEE Trans. Autom. Control*, vol. 59, no. 1, pp. 18–33, Jan. 2014.
[52] P. Giselsson and S. Boyd, "Metric selection in fast dual forward–backward splitting," *Automatica*, vol. 62, pp. 1–10, Dec. 2015.
[53] A. Bradley, "Algorithms for equilibration of matrices and their application to limited-memory quasi-Newton methods," Ph.D. dissertation, Inst. Comput. Math. Eng., Stanford Univ., Stanford, CA, USA, 2010.
[54] Y. Cao, C. Laird, and V. Zavala, "Clustering-based preconditioning for stochastic programs," *Comput. Optim. Appl.*, vol. 64, no. 2, pp. 379–406, 2016.
[55] D. P. Bertsekas, *Nonlinear Programming*, vol. 1. Belmont, MA, USA: Athena Scientific, 1999.
[56] H. Algre, J. Baptista, E. Cabrera, Jr., and F. Cubillo, *Performance Indicators for Water Supply Services* (Manuals of best practice series). London, U.K.: IWA Publishing, 2006.
[57] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed. Nashua, NH, USA: Athena Scientific, 2000.

**Ajay Kumar Sampathirao** received the B.Sc. degree in electrical engineering from the National Institute of Technology, Warangal, India, in 2010, and the M.Tech. degree in control and automation from IIT Delhi, New Delhi, India, in 2012, and defended his Ph.D. degree in computer science and engineering at IMT School for Advanced Studies Lucca, Lucca, Italy, in 2016.

He joined the Control Systems Group, Technical University of Berlin, Berlin, Germany, as a Post-Doctoral Researcher. His current research includes stochastic model predictive control, graphics processing unit-based optimization methods, in particular, for large-scale systems, such as drinking water and power networks.

**Pantelis Sopasakis** was born in Athens, Greece, in 1985. He received the Diploma (M.Eng.) degree in chemical engineering and the M.Sc. degree (Hons.) in applied mathematics from the National Technical University of Athens (NTU Athens), Athens, in 2007 and 2009, respectively. In 2012, he defended the Ph.D. thesis titled Modeling and Control of Biological and Physiological Systems from the School of Chemical Engineering, NTU Athens.

From 2013 to 2016, he was with the Dynamical Systems, Control and Optimization Research Unit, IMT School for Advanced Studies Lucca, Lucca, Italy, as a Post-Doctoral Researcher. Since 2016, he holds a post-doctoral position with the Department of Electrical Engineering, KU Leuven, Leuven, Belgium. His current research interests include the development of numerical algorithms for large-scale stochastic optimal control problems.

**Alberto Bemporad** (F'10) received the master's degree in electrical engineering and the Ph.D. degree in Control Engineering from the University of Florence, Florence, Italy, in 1993 and 1997, respectively.

From 1996 to 1997, he was with the Center for Robotics and Automation, Department of Systems Science and Mathematics, Washington University in St. Louis, St. Louis, MO, USA. From 1997 to 1999, he held a post-doctoral position with the Automatic Control Laboratory, ETH Zürich, Zürich, Switzerland, where he collaborated as a Senior Researcher until 2002. From 1999 to 2009, he was with the Department of Information Engineering, University of Siena, Siena, Italy, becoming an Associate Professor in 2005. From 2010 to 2011, he was with the Department of Mechanical and Structural Engineering, University of Trento, Trento, Italy. He spent visiting periods with the University of Michigan, Ann Arbor, MI, USA, and Zhejiang University, Hangzhou, China. Since 2011, he has been a Full Professor with the IMT Institute for Advanced Studies Lucca, Lucca, Italy, where he served as the Director of the Institute from 2012 to 2015. In 2011, he cofounded ODYS S.r.l., Lucca, a consulting and software development company specialized in advanced controls and embedded optimization algorithms. He has authored over 300 papers in the areas of model predictive control, automotive control, hybrid systems, multiparametric optimization, computational geometry, robotics, and finance. He holds seven patents. He has authored or co-authored various MATLAB toolboxes for model predictive control design, including the model predictive control toolbox (The Mathworks, Inc.) and the hybrid toolbox.

Dr. Bemporad received the IFAC High-Impact Paper Award for the 2011–2014 Triennial. He was an Associate Editor of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL from 2001 to 2004, and the Chair of the Technical Committee on Hybrid Systems of the IEEE Control Systems Society from 2002 to 2010.

**Panagiotis (Panos) Patrinos** received the M.Eng. degree in chemical engineering, the M.Sc. degree in applied mathematics, and the Ph.D. degree in control and optimization from the National Technical University of Athens, Athens, Greece.

He held post-doctoral positions with the University of Trento, Trento, Italy, and also with the IMT School of Advanced Studies Lucca, Lucca, Italy, where he became an Assistant Professor in 2012. In 2014, he held a visiting assistant professor position with the Department of Electrical Engineering, Stanford University, Stanford, CA, USA. He is currently an Assistant Professor with the Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium. His current research interests include the theory and algorithms of optimization and predictive control with a focus on large-scale, distributed, stochastic, and embedded optimization with a wide range of application areas, including smart grids, water networks, aerospace, and machine learning.