POLITECNICO DI MILANO
Facoltà di Ingegneria dell'Informazione
Corso di Laurea Specialistica in Ingegneria dell'Automazione

Model Predictive Control of a Hydro Power Valley

Relatore: Prof. Riccardo Scattolini
Correlatore: Dott. Damien Faille

Tesi di Laurea di:
Francesco Petrone
Matr. 731605

Anno Accademico 2009-2010

Ai miei genitori

Abstract

This Thesis represents the first step of a larger work in the context of the European project "7th Framework STREP project Hierarchical and Distributed Model Predictive Control (HD-MPC)", which involves the study and the development of innovative methods for distributed control of large-scale systems [17].
In particular, in this work the realistic example of a hydropower valley has been considered and a control-oriented numerical model has been derived. This model has allowed the design and implementation of a centralized Model Predictive Control law (MPC), which represents the starting point for further developments of hierarchical and distributed MPC control laws.
The project has been developed in partnership with the French company Électricité de France (EDF), through both correspondence working and internship.

The first step of the project has focused on the analysis of the overall system and on the study of the physical equations describing its behavior. In particular, in view of the design specifications, the most relevant dynamics and those which can be neglected have been identified and the manipulated variables and the controlled ones have been defined.
Then, the whole system has been conceptually decomposed into several subsystems, according to the future design of a distributed control structure.
Subsequently, the physical equations describing the system have been analyzed, and an overall mathematical model has been obtained.
Particular attention has been paid to the de Saint Venant partial differential equations which describe the open-channel hydraulic systems [18] [40] [33] [34] [27] [48] [16] [38]. These equations have been elaborated with a finite difference approach for their numerical solution.

After the derivation of the mathematical model, a simulator has been developed into the Matlab-Simulink software environment [2], maintaining the division into the previously defined subsystems. The most important aspects

of this phase have been the implementation of the de Saint-Venant equations into an efficient vectorial form, and the verification of their compliance with the conditions required for their numerical simulation with explicit integration methods [18].
The simulator has been tested by comparing it with a similar, but more complex and unsuitable for control purposes, simulator developed by EDF into the Scicos-Mascaret environment [13] [1]. In this context, the unknown parameters have been tuned by means of different automatic and manual procedures.

The next step of the work has been the derivation of a tangent linear model at an equilibrium point corresponding to a realistic working condition. The linearization procedure has been undertaken with two different approaches, the first one is numerical and based on an available software tool, the second one is based on the symbolic derivation of the model equations. The two different methods have been then compared and their correspondence with the non linear model has been made around one point.

With the linear model it has been possible to perform some preliminary analyses, such as the direct inspection of the system matrices, the computation of the Relative Gain Array (RGA) and the Singular Value analysis [39] [28], which have led to some interesting conclusions and have allowed a deeper knowledge of the system structure and of the couplings between its variables.

The most complex part of the work consisted in the design and implementation of a centralized Model Predictive Control algorithm [28] [8] [9] [22] [31] [35] for the system. In view of the requirements expressed by the general guidelines of the project, namely to be able to act in several working conditions, with very different control objectives and to produce a software easily adaptable to a subsequent distributed control scheme, it has been necessary to design from scratch a new MPC controller, which had to be flexible enough to allow the use either of a linear or a nonlinear system as prediction model, leaving also the end user the freedom to define the control objective function. First, a software architecture implementing a classical MPC algorithm, operating with the "receding horizon" logic, has been developed. Subsequently, the software used to solve the optimization problem implied by the MPC algorithm has been chosen, evaluating and comparing several available solvers through an appropriate test. Finally, the controller has been provided by an integral action. The problems due to the use of the classical solution for MPC in the considered case have been analyzed and an alternative approach

has been proposed.

The developed controller has been initially tested on the linearized system, using the same model for prediction. Then, it has been possible to compare the results obtained with and without the integral action, highlighting its benefits on the control performances. Finally, the controller has been applied to the nonlinear system, using the linear prediction model. The performance loss due to the imperfect matching between the prediction model used by the controller and that of the controlled system have been evaluated and commented.

The last part of this Thesis presents some expected future developments of the project, which consist in the design of a hierarchical and/or distributed control system for the considered hydro power valley. The reasons that make this approach suitable for large-scale systems, such as the ones previously discussed, have been presented [17], [4]. Finally, some possible ways to organize the distributed structure of the controller are described, together with the necessary assumptions for their implementation and their pros and cons.

Estratto

Il presente elaborato di Tesi vuole essere il primo passo di un lavoro più ampio che si colloca nell'ambito del progetto europeo "7th framework STREP project Hierarchical and Distributed Model Predictive Control (HD-MPC)", il quale prevede lo studio e lo sviluppo di metodi di controllo distribuito innovativi per sistemi con grandi dimensioni [17].
In particolare, in questo lavoro di Tesi si è preso in considerazione l'esempio realistico di una valle per la produzione di energia idroelettrica, della quale si è sviluppato un modello numerico orientato al controllo. Tale modello ha successivamente permesso l'implementazione di una legge di controllo predittivo (Model Predictive Control) centralizzata, punto di partenza per la successiva implementazione del controllo distribuito.
Il progetto è stato svolto in sintonia con la società francese EDF, collaborando sia a distanza sia tramite stage aziendale.

Il primo passo del progetto (Capitoli 2, 3) verte sull'analisi del sistema complessivo e sullo studio delle equazioni fisiche che lo governano.
In particolare, ci si è soffermati sulle specifiche di progetto, valutando quali dinamiche trascurare e quali considerare ai fini delle richieste e ponendo particolare attenzione alla scelta delle variabili di controllo e delle variabili controllate.
Il sistema complessivo è stato quindi scomposto concettualmente in diversi sottosistemi, al fine di predisporne la struttura ad un successivo controllo distribuito.
Successivamente, si sono analizzate le equazioni fisiche che descrivono il sistema in esame, ricavandone quindi il modello matematico complessivo.
Di particolare rilevanza si sono mostrate le equazioni differenziali a derivate parziali di de Saint Venant, che descrivono i sistemi idrici di tipo fluviale [18] [40] [33] [34] [27] [48] [16] [38], le quali sono state trattate con un approccio alle differenze finite al fine di poterle risolvere numericamente.

Una volta ottenuto il modello matematico, si è scelto di implementarlo nell'ambiente

software Matab-Simulink [2], mantenendo la suddivisione in sottosistemi definita precedentemente (Capitoli 4, 5). Gli aspetti più rilevanti di questa fase si sono dimostrati la scrittura delle equazioni di de Saint-Venant nella più efficiente forma vettoriale, al posto della classica formulazione ricorsiva, e la verifica del soddisfacimento della condizione necessaria alla loro simulazione numerica con metodi di integrazione espliciti [18].
Il software sviluppato è poi stato testato mediante un confronto con un analogo simulatore più complesso, quindi inadatto ai fini del controllo, sviluppato da EDF in ambiente Scicos-Mascaret [13] [1]. In questo contesto, i parametri non noti sono stati tarati, analizzando e confrontando diversi metodi di taratura automatica e manuale.

Dopo aver verificato l'attendibilità del modello non lineare sviluppato, si è quindi proceduto alla derivazione di un modello lineare tangente un punto di equilibrio corrispondente ad una verosimile condizione di lavoro (Capitolo 6).
La procedura di linearizzazione è stata intrapresa seguendo due diversi approcci, il primo consiste nel calcolo numerico del modello linearizzato, ottenuto automaticamente tramite un apposto software, il secondo prevede la derivazione simbolica delle equazioni del modello. Le due diverse modalità sono state quindi confrontate tra loro e l'attinenza al modello non lineare è stata verificata.

Sul più semplice modello lineare così ottenuto è stato possibile eseguire alcune analisi di sistema classiche, come l'ispezione diretta delle matrici, l'analisi della matrice dei guadagni relativi (RGA) e l'analisi dei valori singolari [39] [28], le quali hanno condotto a conclusioni interessanti e hanno permesso una più profonda conoscenza della struttura del sistema e degli accoppiamenti tra le sue variabili.

La parte più articolata del presente lavoro di Tesi è consistita nel progetto e nell'implementazione di una logica di controllo centralizzato di tipo predittivo (MPC) [28] [8] [9] [22] [31] [35] per il sistema studiato (Capitolo 7). Viste le necessità, espresse dalle consegne generali del progetto, di poter operare in diverse condizioni di lavoro, con obiettivi di controllo anche molto differenti e di poter adattare il software sviluppato ad un successivo controllo distribuito, si è reso necessario progettare per intero un nuovo controllore MPC abbastanza flessibile da permettere di utilizzare come modello di predizione tanto un sistema lineare quanto uno non lineare, lasciando inoltre all'utente finale la libera definizione della funzione obiettivo del controllo.
In primo luogo, si è sviluppata un'architettura software che implementa

l'algoritmo MPC classico, operante con logica "receding horizon" e con inseguimento di traiettoria.
Successivamente si è scelto il software adibito alla soluzione del problema di ottimizzazione implicito nella logica MPC, valutando e confrontando diversi solutori disponibili mediante un apposito test.
Dopodiché, si è provveduto a fornire il controllore di un'azione integrale, valutando i problemi scaturiti dall'utilizzo della classica soluzione per MPC nel caso trattato e proponendo un approccio alternativo.

Il controllore così costituito è stato inizialmente testato sul sistema linearizzato, utilizzando lo stesso come modello di predizione, in una situazione operativa verosimile. É stato così possibile confrontare i risultati ottenuti con e senza azione integrale, evidenziando i vantaggi apportati da quest'ultima.
Infine, lo stesso controllore è stato messo in opera sul sistema non lineare, utilizzando come modello di predizione quello lineare. La perdita di prestazioni dovuta alla non perfetta corrispondenza tra il modello usato dal controllore ed il sistema controllato è stata quindi valutata e commentata.

L'ultima parte di questo elaborato di Tesi presenta le prospettive future del lavoro svolto, che consisteranno nello sviluppo di un sistema di controllo gerarchico e distribuito per la valle per la produzione di energia elettrica considerata.
Le motivazioni che rendono appetibile un tale approccio per sistemi su larga scala, come quello trattato, vengono inizialmente presentate [17] [4].
Per concludere, vengono esposti alcuni possibili approcci per l'organizzazione della struttura distribuita del controllore, specificandone le ipotesi necessarie all'implementazione e discutendone i rispettivi vantaggi e svantaggi.

Contents

List of Figures

List of Symbols

Symbol	Unit of measurement	Description
$P1$		subsystem including Lake 1 and Valve 1
$P2$		subsystem including Lake 2 and Plant 1
$P3$		subsystem including Lake 3 and Plant 2
$P45$		subsystem including Lake 4, Lake 5 and Plant 3
$P6$		subsystem including Reach 0 and Plant 4
$P7$		subsystem including Reach 1 and Plant 5
$P8$		subsystem including Reach 2 and Plant 6
$P9$		subsystem including Reach 3 and Plant 7
Pe_2	$[W]$	absorbed electrical power of Plant 1
Pe_3	$[W]$	generated electrical power of Plant 2
Pe_5	$[W]$	generated/absorbed electrical power of Plant 3
Pe_6	$[W]$	generated electrical power of Plant 4
Pe_7	$[W]$	generated electrical power of Plant 5
Pe_8	$[W]$	generated electrical power of Plant 6
Pe_9	$[W]$	generated electrical power of Plant 7
L_1	$[m\ o.s.l.]$	absolute water level of Lake 1
L_2	$[m\ o.s.l.]$	absolute water level of Lake 2
L_3	$[m\ o.s.l.]$	absolute water level of Lake 3
L_4	$[m\ o.s.l.]$	absolute water level of Lake 4
L_5	$[m\ o.s.l.]$	absolute water level of Lake 5
L_6	$[m\ o.s.l.]$	absolute water level of Reach 0
L_7	$[m\ o.s.l.]$	absolute water level of Reach 1
L_8	$[m\ o.s.l.]$	absolute water level of Reach 2
L_9	$[m\ o.s.l.]$	absolute water level of Reach 3

Symbol	Unit of measurement	Description
q_{12}	$[m^3/s]$	flow rate from Lake 1 to Lake 2
q_{23}	$[m^3/s]$	flow rate from Lake 2 to Lake 3
q_{34}	$[m^3/s]$	flow rate from Lake 3 to Lake 4
q_{54}	$[m^3/s]$	flow rate from Lake 5 to Lake 4
q_{57}	$[m^3/s]$	flow rate from Lake 5 to Reach 1
q_{58}	$[m^3/s]$	flow rate from Lake 5 to Reach 2
q_{67}	$[m^3/s]$	flow rate from Reach 0 to Reach 1
q_{78}	$[m^3/s]$	flow rate from Reach 1 to Reach 2
q_{89}	$[m^3/s]$	flow rate from Reach 2 to Reach 3
q_{out}	$[m^3/s]$	outlet flow rate of Reach 3
d_1	$[m^3/s]$	inlet flow rate disturb of Lake 1
d_6	$[m^3/s]$	inlet flow rate disturb of Reach 0
d_9	$[m^3/s]$	inlet flow rate disturb of Reach 3
u_1	$[m^3/s]$	flow rate reference for q_{12}
u_2	$[m^3/s]$	flow rate reference for q_{23}
u_3	$[m^3/s]$	flow rate reference for q_{34}
u_{57}	$[m^3/s]$	flow rate reference for q_{57}
u_{58}	$[m^3/s]$	flow rate reference for q_{58}
u_6	$[m^3/s]$	flow rate reference for q_{67}
u_7	$[m^3/s]$	flow rate reference for q_{78}
u_8	$[m^3/s]$	flow rate reference for q_{89}
u_9	$[m^3/s]$	flow rate reference for q_{out}
g	$[m/s^2]$	gravitational acceleration
A_i	$[m^2]$	surface area of Lake i
Zb_i	$[m\ o.s.l.]$	absolute altitude of the bottom of Lake i
Zin_i	$[m\ o.s.l.]$	absolute altitude of the point of inflow in Lake i
$Zout_i$	$[m\ o.s.l.]$	absolute altitude of the point of outflow from Lake i
dZ_{ij}	$[m]$	height difference of the duct from Lake i to Lake j
Kt_i	$[J/m^4]$	power coefficient of the turbine in subsystem Pi
Kp_i	$[J/m^4]$	power coefficient of the pump in subsystem Pi
Ad_{54}	$[m^2]$	section of the duct between Lake 5 and Lake 4
Av_1	$[m^2]$	nominal section of Valve 1
Cr_1		recovery coefficient of Valve 1
η_1		linear opening rate coefficient of Valve 1

Symbol	Unit of measurement	Description
N_i		discretization slots number for subsystem i Reach
dx_i	$[m]$	discretization slots length for subsystem i Reach
X_i	$[m]$	total length of subsystem i Reach
Xin_i	$[m]$	point of lateral inlet flow rate in subsystem i Reach
W_i	$[m]$	width of subsystem i Reach
I_{0_i}		bed slope of subsystem i Reach
I_{f_i}		friction slope of subsystem i Reach
k_{str_i}	$[m^{1/3}/s]$	Strickler coefficient of subsystem i Reach
R_i	$[m]$	hydraulic radius of subsystem i Reach
H_{av_i}	$[m]$	average water depth of subsystem i Reach
H_i	$[m]$	water depth of subsystem i Reach
S_i	$[m^2]$	wetted section of subsystem i Reach
Q_i	$[m^3/s]$	flow rate of subsystem i Reach
q_{l_i}	$[m^2/s]$	lateral inlet flow rate per space unit in subsystem i
c_i	$[m/s]$	celerity of subsystem i Reach
dT	$[s]$	sampling time of the MPC controller
H		prediction receding horizon
Hc		control horizon
nu		number of the control inputs
ny		total number of the system outputs
nU		number of the control inputs in a prediction horizon
nY		number of the system outputs in a prediction horizon
J		cost function of the MPC control
SP_{loc}	$[W]$	set of the local power set points
SP_{glob}	$[W]$	global power set point
SP_L	$[m\ o.s.l.]$	set of the local level set points
Q_P	$[1/W^2]$	powers weighting matrix for local set points
q_P	$[1/W^2]$	total power weight for global set point
R	$[s^2/m^6]$	control inputs weighting matrix
Q_{Pi}	$[1/W^2]$	integral action weighting matrix
$\underline{L_i}$	$[m\ o.s.l.]$	lower bound for the level L_i
$\overline{L_i}$	$[m\ o.s.l.]$	upper bound for the level L_i
$\underline{u_i}$	$[m^3/s]$	lower bound for the control input u_i
$\overline{u_i}$	$[m^3/s]$	upper bound for the control input u_i
$\underline{v_i}$	$[m^3/s^2]$	lower bound for the control input variation
$\overline{v_i}$	$[m^3/s^2]$	upper bound for the control input variation

Chapter 1

Introduction

1.1 Motivation

Hydro power is an essential renewable mean of power generation that in the year 2006 represented the 16% of the world production [17]. In some countries, like in Norway and Brazil, the percentage of hydro can even exceed the 80%; in these operating conditions, hydro power plants must be flexible enough to adjust the electrical power production to the demand. Moreover, in order to reduce the CO2 emission, the contribution of hydro power plants will probably be larger in the future and, at the same time, climate changes will impact the availability of water, so that an efficient management of this resource will be mandatory.
The technology of hydro power plants is already well developed, but significant improvements can still be achieved by using an optimal real-time management of the resources in order to maximize their efficiency.
This management must be compatible with other uses of the hydro resources, such as irrigation, navigation, tourism, with environmental, ecological and safety requirements which impose constraints on levels and flow rates in reservoirs and rivers.
For these reasons, the coordination of the hydro power plants of a given basin, or of a Hydro Power Valley (HPV), can produce a gain in efficiency and increase the power production and the flexibility on the management of the available resources while respecting the constraints. Basically the problem of optimizing the management of a hydro power valley is to follow a given set point for the overall power generated by the plants distributed over the valley while respecting operating constraints. Objectives related to safety and environmental requirements are generally formulated as domain limitations on levels and flow rates. Therefore, the definition of the management

strategy can be formulated as a typical constrained optimal control problem which can be solved with a Model Predictive Control (MPC) approach, see e.g. [8], [22], [31] .

1.2 Overview

In view of the previous motivations, the first objective of this work is to develop and tune a control-oriented model of a hydro power valley. This model is then used to design a control strategy, based on MPC, for the optimal management of all the HPV systems, namely ducts, penstocks, dams, pumps, valves and turbines. The research has been developed in the framework of the European 7th framework STREP project "Hierarchical and Distributed Model Predictive Control (HD-MPC)", where the modeling and control of the HPV considered in this Thesis represents a significant benchmark for the analysis of innovative control methods for large scale systems.

In Chapter 2, the hydro power valley is described, with particular emphasis on the hypotheses and the choices which have been made, such as the parameters definition, the grouping of the subcomponents of the valley, the choice of the control and controlled variables.

Chapter 3 describes the model of the system, which is mainly composed by lakes and reaches, and shows how the electric power, which can be either produced or absorbed by the components of the valley, is calculated. The de Saint Venant equations, which are used to represent the behavior of the reaches, are introduced together with their spatial discretization and the required conditions for their explicit simulation.
Then, the form employed for these equations, the way they are solved and the Simulink model which implements them are described and explained in Chapter 4.
Finally, the tuning procedure of the model so obtained is discussed in Chapter 5.

The obtained nonlinear model is linearized at a given equilibrium point in Chapter 6. The linearization procedure is performed according to two different methods; the results achieved with the two procedures are compared and some further analyses are reported.

In Chapter 7 it is shown how to design a MPC controller, which can rely either on linear or nonlinear models for prediction of the system future behavior. The main problems due to the choice of the optimization solver and the solutions adopted are presented.
Subsequently, the controller is tested in two different situations, representing two real possible set points for the produced power. In the first case, only one global set point, representing the sum of the powers produced by the single units, is considered. In the second case, a list of set points, one for each power plant of the HPV, is defined.

The last chapter (Chapter 8) draws some conclusions and proposes future possible developments of this work, that is the development and the realization of a distributed control structure, which can be designed by properly modifying the centralized MPC controller presented in the previous Chapters.

Chapter 2

System description

2.1 The Valley

A hydro power valley is a system of lakes, reaches, ducts, penstocks, dams, pumps, valves and turbines which are interconnected together and controlled in order to generate electric power, while respecting several constraints, such as the levels of lakes and reaches or the flow rates in pipes and rivers.

The hydro power valley considered in this work is composed by five lakes and three reaches, see Fig.2.1, which describes the layout of the valley.
The lakes are connected together by ducts, where the flow rate is imposed by a valve, a turbine or a pump. The fifth lake feeds two reaches by two ducts and one of them can either turbine or pump the water. The three reaches are separated by plants, each one consisting in a dam and in a power house where a turbine generates the electric power proportionally to the height difference between the upstream and downstream levels.

2.2 Hypothesis and variables definition

The model has to be developed for control purposes, therefore it is important to correctly choose the variables to be controlled and the control inputs. It is also important to define the disturbances which could affect the system.
Recalling that the objective is to generate the electric power according to a slowly-varying set point, while respecting the constraints on the levels and flow rates, the following choices are made:

Controlled variables:

- flow rates in ducts and reaches (q_{ij});

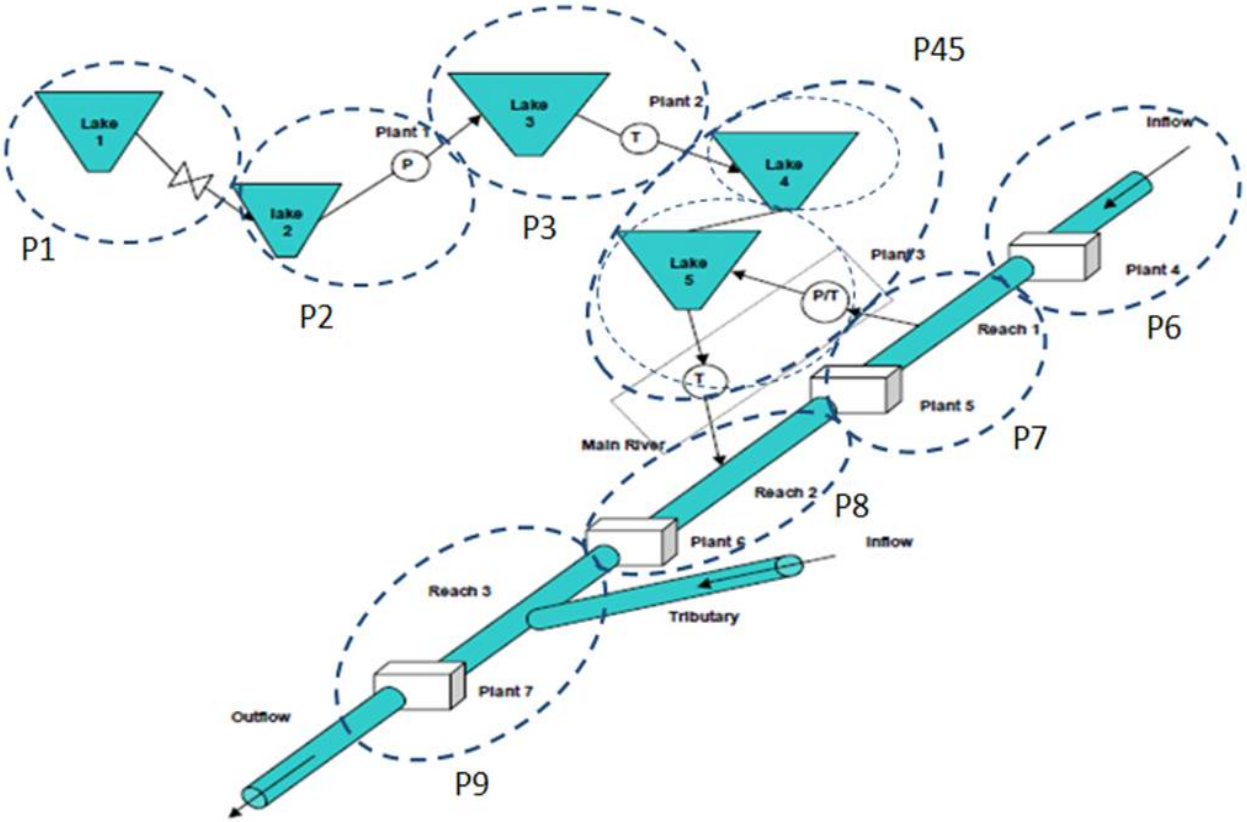

Fig. 2.1: Scheme of the valley.

- levels of lakes and reaches (L_i);
- generated/absorbed electric power (Pe_i).

Control inputs:

- rate opening for valves and turbines and imposed flow rate for pumps, all uniformly expressed in the form of flow rate references (u_i).

Disturbances:

- water inflow due to the upstream rivers, the rain and the thaws (d_i).

Hypothesis:

- the fast dynamics between the set points of the flow rates and the effective value of the discharges of every lake and reach is neglected, so every flow rate (q_{ij}) is always assumed to be equal to his own set point (u_i).

$$q_{ij} = u_i \tag{2.1}$$

  This assumption is reasonable because there are local controllers that regulate the flow rates and compensate the influence of the level variations.

In view of the previous choices, the block-representation of the system is shown in Fig.2.2. In this figure as in the whole Thesis, the following notation is adopted:
For each subsystem i, noted as Pi:

- q_{ij} is the algebraic flow rate from subsystem Pi to subsystem Pj;
- L_i is the level of the lake or of the reach;
- Pe_i is the algebraic electrical power (negative if the plant consumes, positive if it produces);
- u_i is the control input, that is the discharge reference;
- d_i is the water inflow perturbation.

As the duct between Lake 4 and Lake 5 has no devices able to impose the flow rate, such as a pump, a turbine or a valve, the subsystem *P4* has no control input, so it is necessary to merge it with the subsystem *P5*, so generating the block *P45*.

Once the previous choices have been made, it is possible to analyze and draw a mathematical model for each subsystem of the valley.

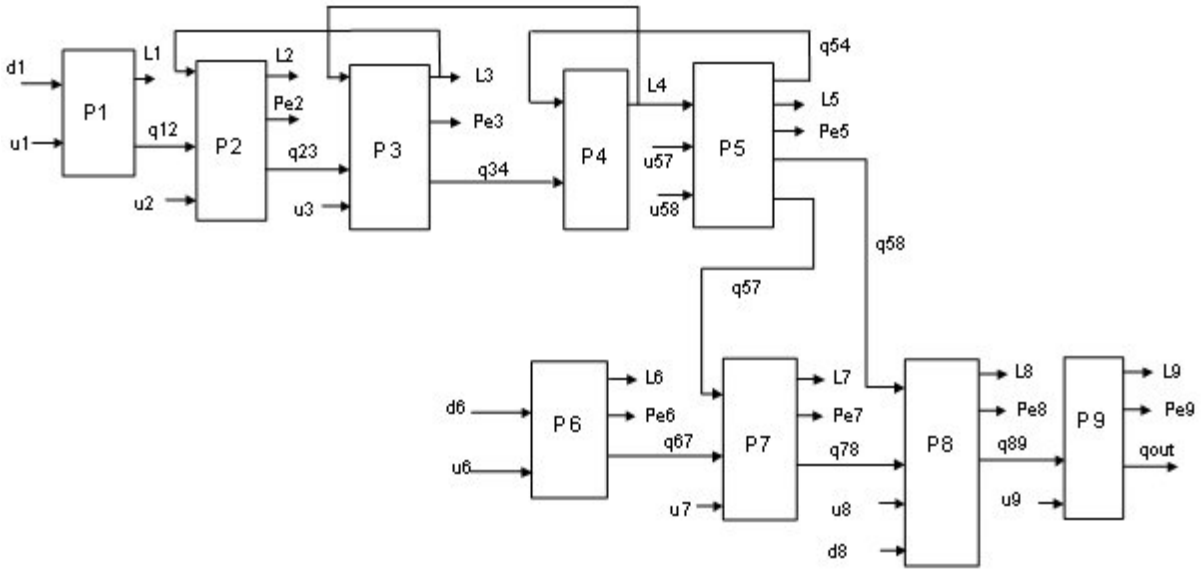

Fig. 2.2: Block scheme of the valley.

Chapter 3

Model of the hydro power valley

3.1 Model of the lakes

In this section the equations and the variables defined to describe the first five blocks (*P1* to *P5*) are presented.

3.1.1 Auxiliary variables

In order to easily write the several equations of the model, a standard nomenclature is defined:

- L_i is the absolute altitude of the water level of Lake i;
- Zin_i is the absolute altitude of the point of inflow in Lake i;
- $Zout_i$ is the absolute altitude of the point of outflow from Lake i;
- dZ_{ij} is the height difference of the duct from Pi to Pj;
- Zb_i is the absolute altitude of the bottom of Lake i;
- A_i is the surface area of Lake i;
- Kt_i is the turbine coefficient of Pi (positive);
- Kp_i is the pump coefficient of Pi (negative).

For example, these variables referred to *P2* and *P3* are described in Fig.3.1.

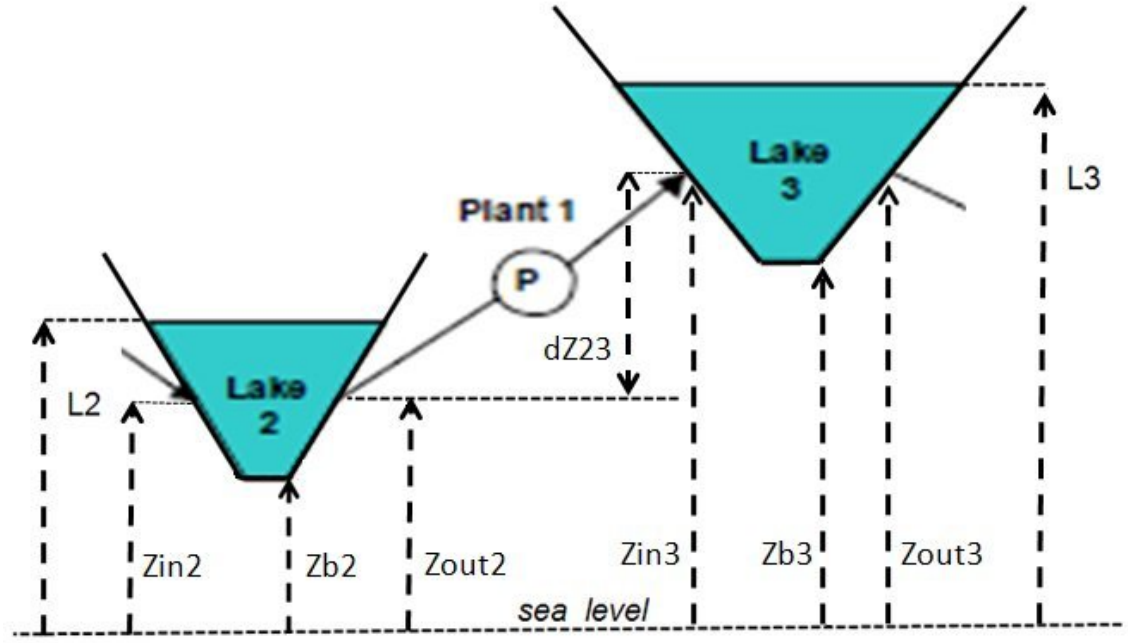

Fig. 3.1: Variables definition in a section of the system.

3.1.2 Model equations

Lakes

Since among the objectives of control there are the inlet and outlet flow rates and the levels, the mass conservation is the only equation needed to describe the behavior of the lakes:

$$\dot{m} = w_{in} - w_{out} \tag{3.1}$$

where m is the mass of the water in a lake, while w_{in} and w_{out} are the (mass) inlet and outlet flow rates. Dividing both terms of the previous equation by the density of the water ρ, the corresponding volumetric equation is obtained, where V is the volume of the water and q_{in} and q_{out} are the inlet and the outlet volumetric flow rates:

$$\dot{V} = q_{in} - q_{out} \tag{3.2}$$

Since the range of variation of the level of a lake is expected to be far lower than the extension of the lake surface, the approximation of vertical banks can be made and the area of the lake can be considered as constant along the

level variation. Under this hypothesis, the level L in the mass conservation equation can be isolated from the area A, obtaining the final relation:

$$\dot{L} = \frac{q_{in} - q_{out}}{A} \tag{3.3}$$

Electric Powers

The electric power generated by the turbines or absorbed by the pumps is calculated in the same way, with the equation (3.4) which depends on the flow rate and the height:

$$P = q \cdot K \cdot H_h \tag{3.4}$$

where q is the flow rate (which flows through the turbine or which is imposed by the pump), K is the turbine or pump coefficient (positive for turbines and negative for pumps), H_h is the difference between the head heights upstream and downstream the turbine or pump.
Under the hypothesis that every inflow always is placed above the lake level, the power calculated in one block is not influenced by the level of the lake/reach in the subsequent block.

Ducts

The pipe which links Lake 4 and Lake 5 is the only one whose discharge is not imposed by a set point, but it is defined by the value of the levels $L4$ and $L5$. The equation which describes the flow rate q_{54} between Lake 5 and Lake 4 is:

$$q_{54} = -sign(L_4 - L_5) \cdot Ad_{54} \cdot \sqrt{2 \cdot g \cdot |L_4 - L_5|} \tag{3.5}$$

where Ad_{54} is the duct section.

3.2 Model of the reaches

In this section, the equations adopted and the variables defined to describe the last four blocks of the valley (*P6* to *P9*) are presented. In particular, the de Saint Venant equations for the reaches and their spatial discretization are introduced.

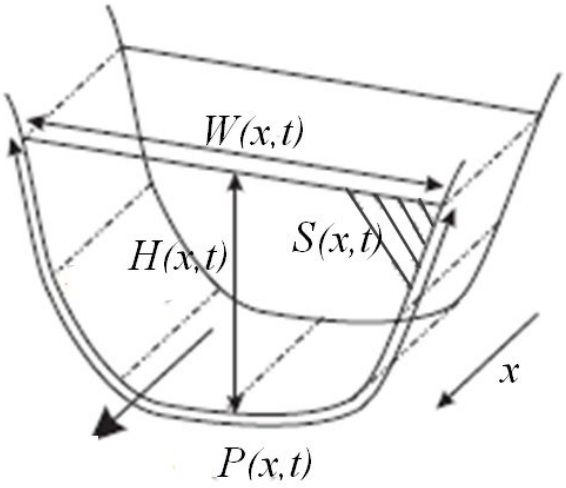

Fig. 3.2: Variables definition in a river cross section.

3.2.1 The de Saint Venant equations

The general equations

De Saint Venant equations represent the state of the art for modeling one-dimensional river hydraulics with constant fluid density, see [38] [16] [48] [27] [40] [18] [33]. They are first order nonlinear equations describing mass and momentum balances in terms of two variables, the water cross section $S(x,t)$ and the flow rate $Q(x,t)$, both depending on the main spatial coordinate of the river x, and on the time t.

Referring to Fig.3.2, which shows all the physical quantities involved in the equations, the de Saint Venant system can be expressed in several ways, but the most famous two are:

1.
$$\begin{cases} \frac{\partial Q}{\partial x} + \frac{\partial S}{\partial t} & = & 0 \\ \frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}(\frac{Q^2}{S}) + gS \cdot \frac{\partial H}{\partial x} + gS \cdot (I_f - I_0) & = & 0 \end{cases} \tag{3.6}$$

2.
$$\begin{cases} \frac{\partial Q}{\partial x} + \frac{\partial S}{\partial t} & = & 0 \\ \frac{1}{g} \cdot \frac{\partial}{\partial t}(\frac{Q}{S}) + \frac{1}{2g} \cdot \frac{\partial}{\partial x}(\frac{Q^2}{S^2}) + \frac{\partial H}{\partial x} + I_f - I_0 & = & 0 \end{cases} \tag{3.7}$$

where $S(x,t)$ is the wetted area $[m^2]$, $Q(x,t)$ the discharge $[m^3/s]$ across the section S, $H(x,t)$ the water depth $[m]$, $I_f(x,t)$ the friction slope, I_0 the bed

slope and g the gravitational acceleration $[m/s^2]$.
The friction slope I_f is defined by the Manning-Strickler formula:

$$I_f(x,t) = \frac{(Q/S)^2}{k_{str}^2 \cdot R^{4/3}} \qquad R = \frac{S}{P} \tag{3.8}$$

where $k_{str}(x)$ is the Strickler coefficient $[m^{1/3}/s]$, $R(x,t)$ the hydraulic radius $[m]$ and $P(x,t)$ the wetted perimeter $[m]$.

In both forms (3.6) and (3.7), the first equation originates from the conservation of mass while the second equation results from the conservation of momentum. It is possible to demonstrate the equivalence of the two forms (see Appendix A), but the first one will be used in the sequel, because in its momentum equation the time derivative of the discharge can easily be isolated from the other variables.

Assumptions

In order to first produce and test an initial and quite simple model of the reaches, the assumptions of constant river width W and rectangular cross section S are made:

$$W(x,t) = W \tag{3.9}$$

$$P(x,t) = 2H(x,t) + W \tag{3.10}$$

Under these hypothesis, it is possible to isolate the river width W from the time derivative of the cross section $S(t,x)$, so that the water depth $H(x,t)$ becomes a state variable:

$$\frac{\partial S(x,t)}{\partial t} = W \cdot \frac{\partial H(x,t)}{\partial t} \tag{3.11}$$

The substitution of all these equations in the first de Saint Venant form allows one to obtain the full PDE system used for the reach model:

$$\begin{cases} \frac{\partial H}{\partial t} &= -\frac{1}{W} \cdot \frac{\partial Q}{\partial x} \\ \frac{\partial Q}{\partial t} &= -\frac{2Q}{WH} \cdot \frac{\partial Q}{\partial x} + [\frac{1}{W}(\frac{Q}{H})^2 - gWH] \cdot \frac{\partial H}{\partial x} + gWI_0H - \frac{gWH}{k_{str}^2} \cdot (\frac{W+2H}{WH})^{4/3} \cdot (\frac{Q}{WH})^2 \end{cases} \tag{3.12}$$

3.2.2 Spatial discretization

A simple way to implement and simulate a PDE model is to discretize it into several ODE systems, by substituting the space derivatives with their corresponding finite differences. It is therefore advisable to discretize the reach into N sections along the flow direction, each one with the length of $dx = X/N$, where X is the total length of the reach. In order to avoid stiffness in the simulation and to correctly describe the physics of the system, the discretization steps of the two variables Q and H are overlapped, so that each height depends on the previous and forward discharges, and viceversa, see [38],[16],[48],[18]. Following this principle, the reach discretization scheme adopted and the finite difference approximation equations are reported in Fig.3.3 and in equation (3.13).

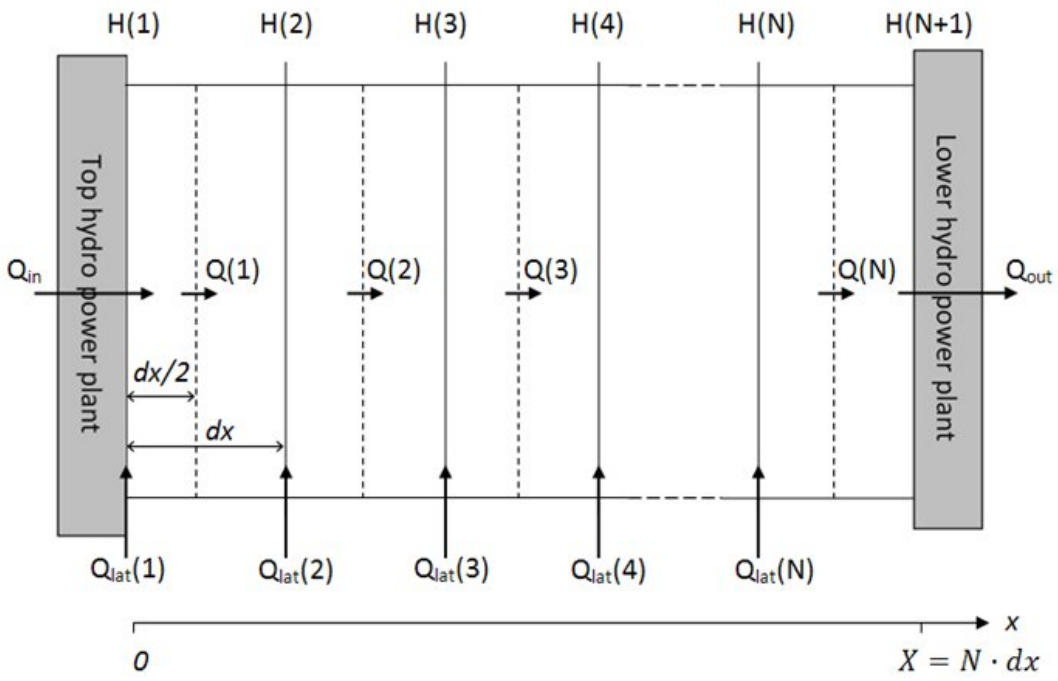

Fig. 3.3: Spatial discretization.

$$\frac{\partial H_i}{\partial x} \simeq \frac{(H_{i+1} - H_i)}{dx} \qquad \frac{\partial Q_i}{\partial x} \simeq \frac{(Q_i - Q_{i-1})}{dx} \tag{3.13}$$

The corresponding de Saint Venant system is therefore:

$$
\begin{cases}
\frac{\partial H_1}{\partial t} &= -\frac{1}{W} \cdot \frac{Q_1 - Q_{in}}{dx/2} \\
\frac{\partial Q_1}{\partial t} &= -\frac{2Q_1}{WH_1} \cdot \frac{Q_1 - Q_{in}}{dx} + [\frac{1}{W}(\frac{Q_1}{H_1})^2 - gWH_1] \cdot \frac{H_2 - H_1}{dx} + \\
& \quad + gWI_0H_1 - \frac{gWH_1}{k_{str}^2} \cdot (\frac{W+2H_1}{WH_1})^{4/3} \cdot (\frac{Q_1}{WH_1})^2
\end{cases}
$$

$$
\begin{cases}
\frac{\partial H_i}{\partial t} &= -\frac{1}{W} \cdot \frac{Q_i - Q_{i-1}}{dx} \\
\frac{\partial Q_i}{\partial t} &= -\frac{2Q_i}{WH_i} \cdot \frac{Q_i - Q_{i-1}}{dx} + [\frac{1}{W}(\frac{Q_i}{H_i})^2 - gWH_i] \cdot \frac{H_{i+1} - H_i}{dx} + \\
& \quad + gWI_0H_i - \frac{gWH_i}{k_{str}^2} \cdot (\frac{W+2H_i}{WH_i})^{4/3} \cdot (\frac{Q_i}{WH_i})^2 \\
i \in [2, N]
\end{cases}
\tag{3.14}
$$

$$
\frac{\partial H_{N+1}}{\partial t} = -\frac{1}{W} \cdot \frac{Q_{out} - Q_N}{dx/2}
$$

The reaches of the considered hydro power valley are not isolated, but they are also connected with the rest of the HPV by lateral inflow or outflow channels. Therefore, in each section it is necessary to take into account also the contribution of a possible additive lateral flow rate, which is supposed to get into the reach perpendicularly to the main flow direction.
Hence, the lateral inflow affects only the mass conservation equation in (3.6) and (3.7), which is modified as follows, [27],[40]:

$$
\frac{\partial Q}{\partial x} + \frac{\partial S}{\partial t} = q_l \tag{3.15}
$$

where $q_l(x)$ is the lateral discharge per space unit $[m^2/s]$.

In conclusion, the full ODE system which has to be implemented to simulate the behavior of each reach is:

$$
\begin{cases}
\frac{\partial H_1}{\partial t} &= -\frac{1}{W} \cdot \frac{Q_1 - Q_{in} - Qlat_1}{dx/2} \\
\frac{\partial Q_1}{\partial t} &= -\frac{2Q_1}{WH_1} \cdot \frac{Q_1 - Q_{in}}{dx} + [\frac{1}{W}(\frac{Q_1}{H_1})^2 - gWH_1] \cdot \frac{H_2 - H_1}{dx} + \\
& \quad + gWI_0H_1 - \frac{gWH_1}{k_{str}^2} \cdot (\frac{W+2H_1}{WH_1})^{4/3} \cdot (\frac{Q_1}{WH_1})^2
\end{cases}
$$

$$
\begin{cases}
\frac{\partial H_i}{\partial t} &= -\frac{1}{W} \cdot \frac{Q_i - Q_{i-1} - Qlat_i}{dx} \\
\frac{\partial Q_i}{\partial t} &= -\frac{2Q_i}{WH_i} \cdot \frac{Q_i - Q_{i-1}}{dx} + [\frac{1}{W}(\frac{Q_i}{H_i})^2 - gWH_i] \cdot \frac{H_{i+1} - H_i}{dx} + \\
& \quad + gWI_0H_i - \frac{gWH_i}{k_{str}^2} \cdot (\frac{W+2H_i}{WH_i})^{4/3} \cdot (\frac{Q_i}{WH_i})^2 \\
i \in [2, N]
\end{cases}
\tag{3.16}
$$

$$
\frac{\partial H_{N+1}}{\partial t} = -\frac{1}{W} \cdot \frac{Q_{out} - Q_N}{dx/2}
$$

Chapter 4

Simulink Implementation

The model of the hydro power valley described in the previous chapters has been implemented in the Matlab/Simulink environment. This is a fundamental step in the development of the work, since the availability of an accurate model of the HPV is mandatory to design a reliable control scheme coping with the goals of the project and fulfilling the physical constraints on the system variables.

4.1 Simulation of the lakes

Levels

This section shows how to implement a Simulink scheme simulating the behavior of a lake level according to the modeling assumptions described in Section 3.1.
As already noted, there is only one state equation for each lake and its form which better fits our purposes is given by equation (3.3). In order to easily implement this equation in the Simulink model, a simple Matlab function which computes the level derivative is employed. Its exit value is then integrated, so finally obtaining the level value.
For example, the Simulink scheme which computes the level of Lake 2 is shown in Fig.4.1.

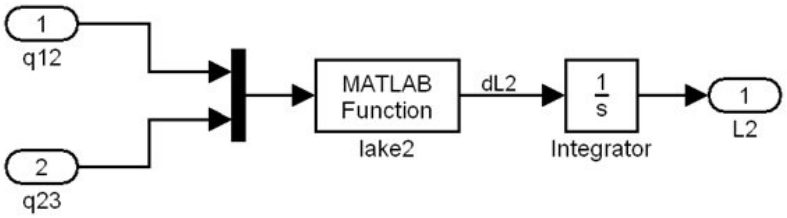

Fig. 4.1: Simulink implementation for Lake 2.

function out=lake2 (u,A2)

out(1)=(u(1)-u(2))/A2;

% u(1)=q12
% u(2)=q23
% out(1)=dL2

The Matlab function is the same for Lake 1, Lake 2, Lake 3 and Lake 4, while the function of Lake 5 simply has one more input, corresponding to the second outlet water flow.

Electric Powers

As for the lakes, it is possible to implement the equation (3.4) in Simulink with a simple Matlab function. As an example, the function which computes the power absorbed by the pump in block $P2$ is:

function out=power2(u, Zout2, dZ23, K2)

*out(1)= u(2)*K2*(dZ23-(u(1)-Zout2));*

% u(1)=L2
% u(2)=q23
% out(1)=Pe2

In the previous chapter, the hypothesis that every inflow always arrives above the lake level has been introduced. This assumption is usually verified; however, in order to correctly describe all the possible situations, it is necessary to modify the Matlab function for the power computation in the following way:

```
function out=power2(u, Zout2, dZ23, K2)

if u(3)>Zin3
     out(1)= u(2)*K2*(u(3)-u(1));
else
      out(1)= u(2)*K2*(dZ23-(u(1)-Zout2));
end

% u(1)=L2
% u(2)=q23
% u(3)=L3
% out(1)=Pe2
```

Linking the functions described above, the whole Simulink structure of block *P2* is shown in Fig.4.2.

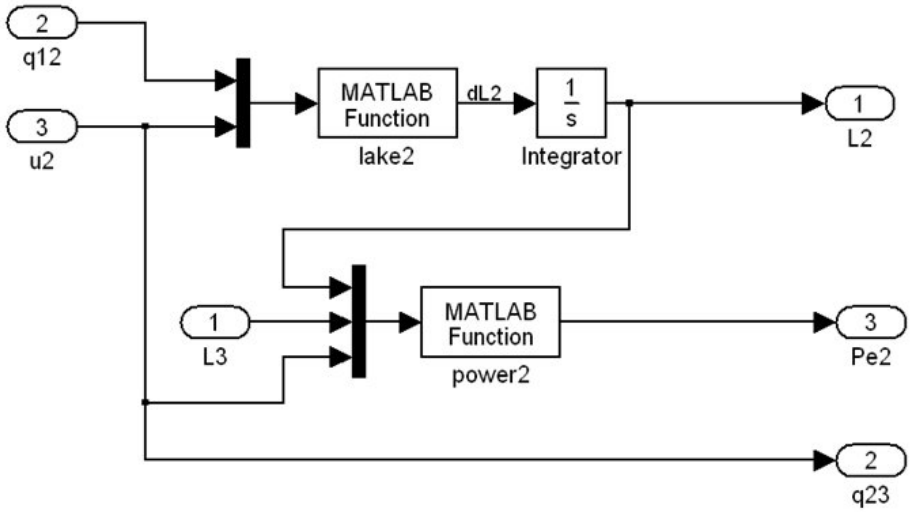

Fig. 4.2: Simulink implementation of block *P2*.

The same structure is used to implement block *P3*, which differs from *P2* only for the sign of the power coefficient K.
The block *P45* contains a reversible group that can either pump or turbine the water, therefore the total algebraic power depends on the sign of the flow rate in the penstock between the subsystems *P5* and *P7* and is so calculated:

```
function out=power5(u, Zout57, Zout58, dZ57, dZ58, K57t, K57p, K58)

if u(2)>0
     out(1)= u(2)*K57t*(dZ57+u(1)-Zout57) + u(3)*K58*(dZ58+u(1)-Zout58);
else
     out(1)= u(2)*K57p*(dZ57+u(1)-Zout57) + u(3)*K58*(dZ58+u(1)-Zout58);
end

% u(1)=L5
% u(2)=q57
% u(3)=q58
% out(1)=Pe5
% K57t generated power coefficient of the turbine between P5 and P7(positive)
% K57p absorbed power coefficient of the pump between P7 and P5 (negative)
% K58 generated power coefficient of the turbine between P5 and P8(positive)
```

Flow rates in ducts

The equation which computes the flow rate through the duct between Lake 5 and Lake 4, presented in (3.5), is implemented as follows:

```
function out=duct54(u,g,Ad54)

out(1)=-sign(u(1)-u(2))*Ad54*sqrt(g*2*abs(u(1)-u(2)));

% u(1)=L4
% u(2)=L5
% out(1)=q54
% Ad54 section of the duct
```

The complete Simulink diagram that describes the part of the valley contained in the block *P45* is therefore presented in Fig.4.3.

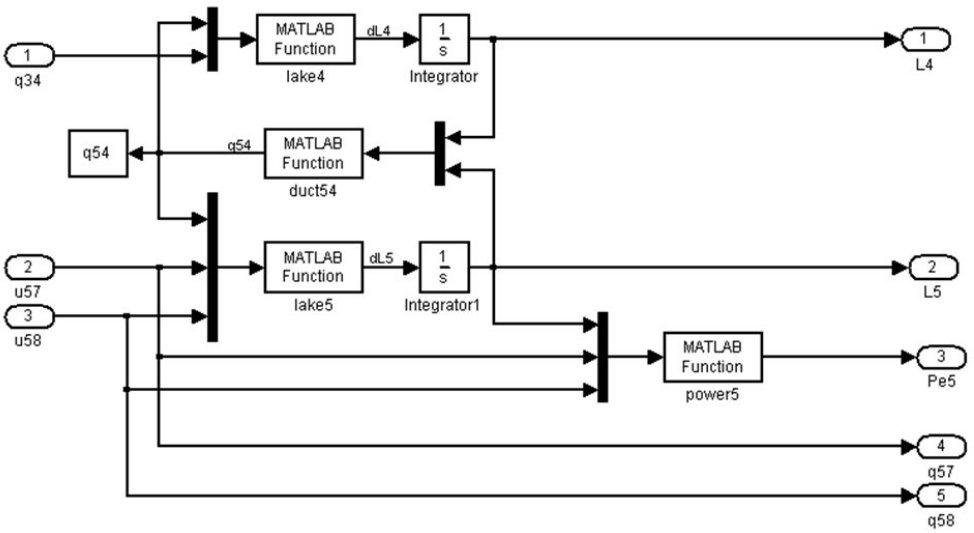

Fig. 4.3: Simulink implementation of block *P45*.

4.2 Simulation of the reaches

4.2.1 Integration of the de Saint Venant equations

There are two main approaches to implement in Simulink the *2N+1* dimensional system of ordinary differential equations described by the equations (3.16) in Section 3.2.2.
The first one is to create an iterative loop which, at each iteration, covers half discretization section of the reach, calculating either the height or the discharge and sequentially saving them in one vector. The second one is to directly use the Matlab vector operations, which allow one to compute the full vector of the heights and the full vector of the discharges in a one-shot operation. This approach is computationally much more efficient than the iterative one and for this reason it has been used in the Simulink implementation.

In any case, the computational burden is heavy and initially a S-functions has been employed, because of its pre-compiled structure, which allows for faster simulations. However, running the simulator several times and with different values of the parameters, it was possible to notice that the simulation used to break down with a numerical error when the values of the river width (W_i) or the values of the number of discretization sections (N_i) were too high.
This problem has been solved by using a Matlab function which computes

the derivatives of both levels H_i and flow rates Q_i, coupled with a pair of integrators, which integrate separately the levels derivatives and the flow rates derivatives. In this way, each integrator works on similar numerical values and it is possible to define different -and more coherent- values of saturation for the integrated variables. A proper setting of the saturation levels, sufficiently far from the expected operating point in order to not influence the simulation, is sufficient to avoid the numerical problems encountered with the S-functions.

According to the previous considerations, the Simulink scheme which implements the behavior of one reach is shown in Fig.4.4 and its Matlab function, which computes the derivatives of the state variables of the reach, is presented below.

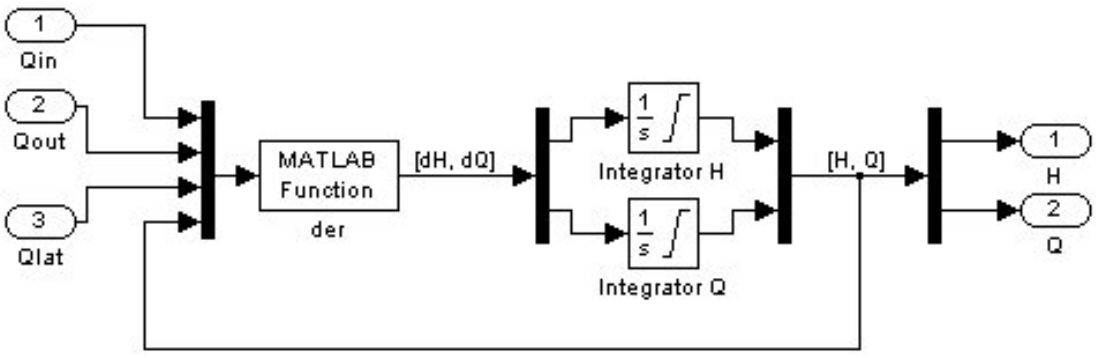

Fig. 4.4: Simulink implementation of block *P7*.

```
function dy=der(t,y,u,X,N,W,g,kstr,I0,Xin)

dx=X/N;
n=ceil(N*Xin/X+0.01);

Qin=u(1);
Qout=u(2);
Qlat=linspace(0,0,N);
Qlat(n)=u(3);

dH(1)= -1/W*(Q(1)-Qin -Qlat(1))/(dx/2);
dQ(1)= -2*Q(1)/W/H(1)*(Q(1)-Qin)/dx +
   + ((Q(1)/H(1))^2/W - g*W*H(1))*(H(2)-H(1))/dx +
   + g*W*I0*H(1) +
   - g*W*H(1)/kstr^2*((W+2*H(1))/W/H(1))^(4/3)*(Q(1)/W/H(1))^2;

dH(2:N)= -1/W*(Q(2:N)-Q(1:N-1)-Qlat(2:N)')/dx;
dQ(2:N)= -2*Q(2:N)./(W*H(2:N)).*(Q(2:N)-Q(1:N-1))/dx +
   + ((Q(2:N)./H(2:N)).^2/W - g*W*H(2:N)).*(H(3:N+1)-H(2:N))/dx +
   + g*W*I0*H(2:N) +
   - g*W*H(2:N)/kstr^2.*((W+2*H(2:N))/W./H(2:N)).^(4/3).*(Q(2:N)/W./H(2:N)).^2;

dH(N+1)= -1/W*(Qout-Q(N))/(dx/2);

dy=[dH dQ];
```

4.2.2 The Courant-Friedrichs-Lewy Stability Criterion

Simulink employs mainly explicit simulation methods, therefore it is necessary to guarantee the stability of the simulation when the simulation step increases. This problem is particularly critical in the integration of the de Saint Venant equations, and many studies have been devoted to its solution. A general criterion for the choice of the number N of discretization sections in order to guarantee the stability of the integration method has been described in [18]. It is called the "Courant Criterion" and is defined as follows:

$$\Delta t < \frac{dx}{c} \tag{4.1}$$

where dx is the length of each discretization section, Δt is the integration step and c is the celerity defined as:

$$c = \sqrt{g \cdot H_{av}} \tag{4.2}$$

where H_{av} is the average height of the water.
Therefore, before running the simulation, it is necessary to verify that the adopted number of discretizations N for each reach is sufficiently small to satisfy the Courant Criterion, using for H_{av} the average height of the equilibrium regime:

$$\Delta t < \frac{X/N}{c} \quad \Rightarrow \quad \Delta t < \frac{X/N}{\sqrt{g \cdot H_{av}}} \quad \Rightarrow \quad N < \frac{X}{\Delta t \cdot \sqrt{g \cdot H_{av}}} \tag{4.3}$$

4.3 Full Simulink scheme

Following all the previous considerations, the hydro power valley considered in this Thesis can be described by the Simulink scheme with nine subsystems presented in Fig.4.5.

4.4 Control of the fast dynamics

It has been previously inferred that it is reasonable to neglect the fast dynamics between the discharge references (u_i) and the outlet flow rates (q_{ij}). In order to verify this hypothesis, a simple PI control on the Lake 1 discharge (q_{12}) has been implemented.
The opening rate (x) of the valve is chosen as control variable, while the control error is the difference between q_{12} and the reference u_1.

The valve is modeled by the following equations:

$$q_{12} = sign(L_1 - L_2) \cdot kv_1 \cdot \eta_1(x) \cdot \sqrt{g \cdot |L_1 - L_2|} \tag{4.4}$$

$$kv_1 = \frac{Av_1}{Cr_1} \cdot \sqrt{2} \tag{4.5}$$

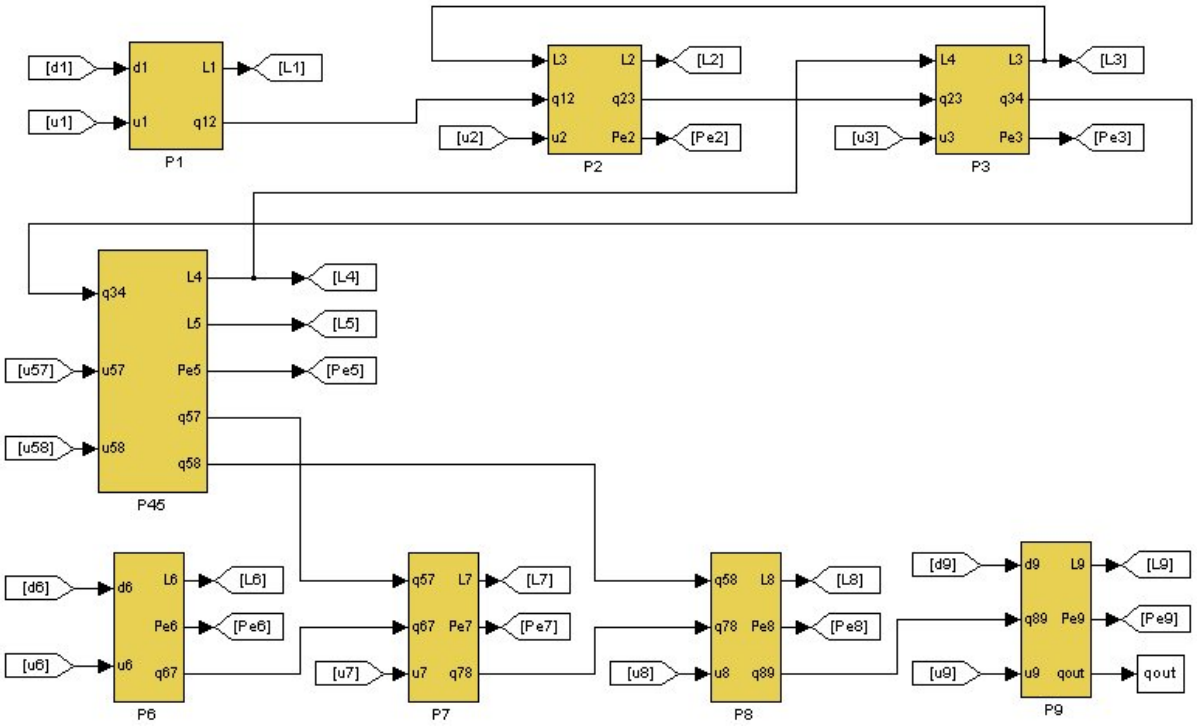

Fig. 4.5: Full Simulink scheme of the hydro power valley.

where Av_1 is the valve nominal section, Cr_1 is the recovery coefficient and η_1 is the function which links the opening rate x to the effective valve section. Both x and η_1 assume values between 0 and 1.
Under the hypothesis of linear relation, η_1 can be considered as a linear coefficient (for example η_1=1). With this assumption, the Simulink scheme which implements the PI control between the control input (u_1) and the discharge from Lake 1 (q_{12}) is shown in Fig.4.6 and the Matlab function of the valve, which considers also the difference in altitude between Lake 1 and Lake 2, is presented below.

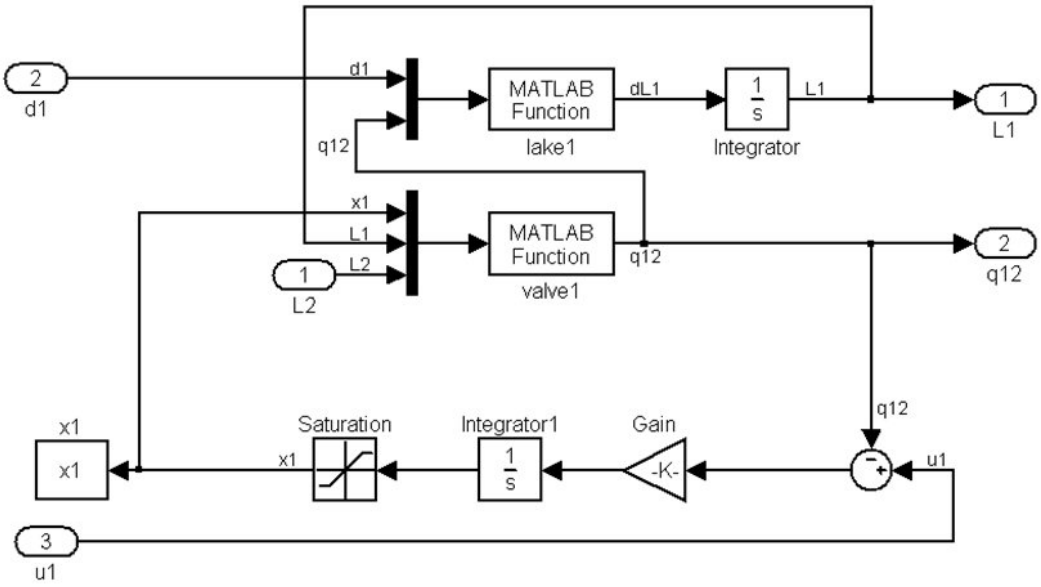

Fig. 4.6: Simulink implementation of block *P1* with PI control.

```
function out=valve1(u,g,Av1,Cr1,eta1,Zout1,Zin2)

kv1=sqrt(2)*Av1/Cr1;
if u(3)>Zin2
      out(1)=kv1*eta1*u(1)*sqrt(g*(u(2)-u(3)));
else
      out(1)=kv1*eta1*u(1)*sqrt(g*(u(2)-Zout1));
end

% u(1)=x1
% u(2)=L1
% u(3)=L2
% out(1)=q12
```

Simulating the whole system with a proper assignment of the gain of the valve control loop, it has been verified that the dynamic between the discharge reference u_1 and the flow rate q_{12} is much faster than those between the system inputs u_i and the other considered variables (levels L_i and powers Pe_i), as long as the opening of the valve remains in its non saturated domain. Then, the valve control loop has been removed from the model and each flow rate q_{ij} has been posed equal to its reference u_i.

as long as the opening of the valve remains in its non saturated domain.

Chapter 5

Model Tuning

The Simulink model presented in Chapter 4 has been developed for control synthesis purposes, therefore it has to be simple enough to easily permit the project of a control law coping with the requirements and constraints previously defined. At the same time, it must be sufficiently detailed to capture the dynamics of the real system we are interested in. This trade-off leads to a model whose parameters have to be tuned so as to duplicate the dynamics of a more detailed one, which can be usefully employed for simulation and validation, but which is not well suited for model-based control due to its complexity.
In the problem analyzed in this Thesis, a non-linear complex model of the considered hydro power valley has been provided by EDF R&D-STEP. This model is implemented in the software environment Scilab-Scicos and uses nonlinear sub-blocks coded in the Mascaret language [1] [13] in order to solve the de Saint Venant equations.
Despite the Scicos model was already available when this Thesis started, and its development has been completely uncorrelated from the Simulink model of Chapter 4, they obviously have two similar sets of parameters, which can both be set at the same values. The only exceptions are the Strickler coefficient k_{str} and the river width W:

- the Strickler coefficient is not set into the Scicos model as it is automatically calculated by the internal algorithm;
- the river width is set into the Scicos model as a matrix of values which defines the shape of the river section along the stream.

In the adopted tuning procedure, the river width has been initially approximated with a linear interpolation of the shape matrix and only the Strickler coefficient has been tuned (Section 5.1).

Subsequently, the hypothesis of a well-fitting linear interpolation has been relaxed and a sort of tuning also for the river width has been made (Section 5.2).

5.1 Tuning of the Strickler coefficient

5.1.1 Assumptions and problem definition

The Scicos model used to test and tune the Simulink one is very detailed and takes the information of the river shape from a 3D-matrix of values. This allows one to obtain detailed simulation results, but this information is too complex to be taken into account by our control-oriented model. For this reason, a first attempt can be done by approximating the section at each discretization step of the Simulink model as a rectangle, whose dimensions are computed by linear interpolation of the parameter matrix of the Scicos model. This approximation implies the assumptions that the behavior of the reach is indeed well-described by a lower number of parameters and that the linear approximation does not move far away from the real shape of the river.

According to the previous statements, for each reach the only parameter which has to be tuned is the Strickler coefficient k_{str}. Hence, the parameter estimation procedure consists in looking for the value of k_{str} that minimizes an objective function based on the difference between the output value of the Simulink reach model, namely the level measured at the end of the reach, and the corresponding output returned by a reference Scicos simulation, when the two models are set with the same values of parameters, they are stimulated by the same configuration of inputs and they work at the same operating point.
The objective function J can easily be chosen as a quadratic one:

$$J = \sum_{t=0}^{T_{sim}} (L(\hat{k}_{str}, t) - L_r(t))^2 \tag{5.1}$$

where:

- T_{sim} is the simulation time of the Simulink model called at each function evaluation;
- $\hat{k}_{str}$ is the current guess for the Strickler coefficient value;
- $L(\hat{k}_{str}, t)$ is the scalar level output at time t of the Simulink reach model;

- $L_r(t)$ is the scalar level output at time t of the Scicos reach model.

The parameter estimation scheme is presented in Fig.5.1, where k_{str} is the unknown parameter, α is the vector of known parameters and $u(t)$ is the time-varying vector of the system inputs in a defined configuration (scenario).

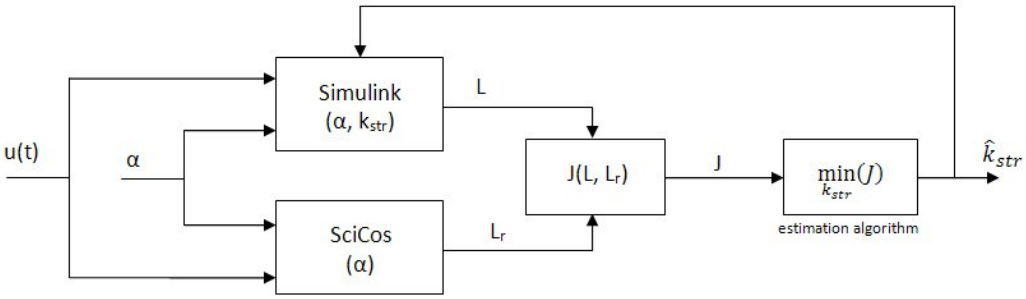

Fig. 5.1: Parameter estimation scheme.

5.1.2 Minimization methods

In order to find the value of the unknown parameter which minimizes the defined objective function, different non linear minimization methods can be used, such as:

- Gradient descent method [44]:
  The estimate of the unknown parameter x is iteratively computed by upgrading the current value with the gradient of the goal function, weighted by a coefficient μ.

$$x_{i+1} = x_i - \mu_i \cdot \left[\frac{\partial J}{\partial x}\right]_{x=x_i} \tag{5.2}$$

- Adjoint state method [7]:
  It is an extension of the gradient descent method, where an auxiliary unknown variable p is introduced in the upgrade equation; another equation, which extends the state of the discrete system, for p is necessary.

- Matlab routine *lsqnonlin* (various methods) [2]:
  It is a Matlab function explicitly created for the solution of the minimization of nonlinear problems whose objective function is defined as a sum of squares.
  It employs different methods, depending on the selected options [2]:

– by default, it uses a large-scale algorithm. This algorithm is a subspace trust region method and is based on the interior-reflective Newton method. Each iteration involves the approximate solution of a large linear system using the method of Preconditioned Conjugate Gradients (PCG);

– if the option "LargeScale" is set to "off", the *lsqnonlin* routine uses the Levenberg-Marquardt method with line-search.

– alternatively, a Gauss-Newton method with line-search may be selected, which is generally faster when the residual is small.

5.1.3 Tuning algorithm

The tuning algorithm consists in finding, independently for each reach, the value of k_{str} which minimizes the sum of squares of the difference between the time responses of the same reach in the two models and with the same scenario.
According to previous simulations, the selected tuning scenario is set as a sequence of two ramps of the flow rate at the inlet section of the reach. First, a positive ramp which rise from $0m^3/s$ to $40m^3/s$ in $600s$, then, $1600s$ later, a negative ramp with the same features. The variable to be considered in setting the optimization procedure is the water level at the end of the reach. In order to use a reliable and already developed minimization routine, the *lsqnonlin* optimization function in its default configuration has been chosen.

The kernel of the Matlab script which performs the minimization (tuning) is the command:

```
k=lsqnonlin(@(k) J(k,tr,Lr),k0);
```

where *tr* and *Lr* are the time and the level of the time response of the reference model, the Scicos one, stored in a Matlab variable, while *k0* is the first guess value for the Strickler coefficient.
J is the Matlab function which simulates the Simulink model of the reach in the chosen scenario and with the current value of the unknown parameter k and computes the difference between its response L and Lr. At each function evaluation of the minimization method implemented by *lsqnonlin*, a simula-

tion is called passing to the Simulink model a different value for the Strickler coefficient.
The full script which defines this function is presented below:

function out=J(k,tr,Lr)

*L=sim('reachsimID',[ ],[ ],[tr,ones(length(tr),1)*k]);*

out=Lr-L;

where:

- *reachsimID.mdl* is the Simulink model which simulates only the reach currently analyzed;
- *[tr,ones(length(tr),1)*k]* is the time vector of the external input, namely the current value of the unknown parameter k, which is constant during the simulation.

A sample of a Simulink scheme used for the tuning of one reach is shown in Fig.5.2

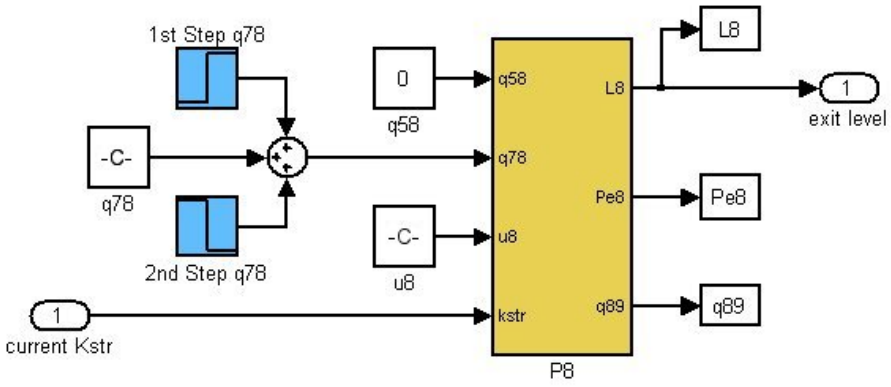

Fig. 5.2: Simulink model which simulates the reach in *P8* during the tuning procedure.

If the simulation is forced to use a fixed step time, the length of L is the same as Lr and no other expedients are requested. On the contrary, if the achievement of a satisfactory precision in the considered scenario using a fixed step method implies a too large simulation time, it is necessary to simulate the Simulink model with a variable step method. In this case, the

base algorithm would not run, due to the difference in size and timing of the two vectors of the time responses. It is however possible to run a further algorithm which interpolates the two time responses referring to a defined time vector, for example that returned by the Scicos simulation. In this case, the function which computes the residuals used by *lsqnonlin* is no longer *J*, but *Jint* computed as follows:

```
function out=Jint(k,tr,Lr)

[t,S,L]=sim('reachsimID',[ ],[ ],[tr,ones(length(tr),1)*k]);
Li=interp1(t,L,tr,'spline');

out=Lr-Li;
```

However the above procedure, proposed to circumvent the problems related to the use of a variable step method, implies a very high computational burden for the *lsqnonlin* optimization solver, so that no satisfactory solutions have been reached.
For this reason, it is better to perform the model tuning based only on the Strickler coefficient with the first and simpler algorithm, simulating the two models with a fixed step method and adopting the same step time and the same simulation time.

5.1.4 Results

By applying several times the algorithm presented above, each time starting from a different initial guess for the unknown parameter, it can be noticed that the optimal solution given by the optimization solver never makes the objective function to be close to zero. In order to understand the cause, it is worth to analyze the results obtained and to perform a few more simulations of the Simulink model using different values of the Strickler coefficient. Looking at these simulations, it can be noticed that the difference between the two time responses L_r and L can be split into two main components:

1) the difference in the final value of the transient, starting from the same point;

2) the difference in the amplitude of the oscillations in the last part of the transient.

In particular, observing the results, the second difference can be directly imputed to the Strickler coefficient, while the first one seems to be independent from this parameter. This leads to infer that the Strickler coefficient is not the only unknown parameter which contributes to the difference between the two models, as it affects only the oscillations at the end of the transient. So there must be some other parameters which influence the height of the time response.
As stated above in Section 5.1.1, these parameters are necessarily those that define the shape of the river, which has been approximated as rectangular and dimensioned with a linear interpolation of the Scicos input data. Therefore, it can be argued that this approximation is not sufficient to allow the two models to have the same behavior and, of course, to perform a satisfactory parameter identification based only on the Strickler coefficient.
It is so necessary to perform a more refined tuning, based also on the river shape parameters.

5.2 Tuning based on river width

In order to perform a tuning procedure which takes in account both the shape parameters and the Strickler coefficient, two different approaches can be followed:

1) to use all the information provided on the river shape by Scicos, building a new 3D-discretized model for the reaches;

2) to maintain the current approximation of rectangular section and 1D discretization, while approximating the river width no longer with a linear interpolation, but with a constant equivalent value which has to be tuned in order to make the two responses fit.

Of course the first solution is too complex and is far from the objective of the present work, which is to obtain a simple but realistic control-oriented model. Furthermore, it would imply the development of a model with the same parameters of the Scicos one, while the goal is to produce a model as general as possible, in order to allow further uses.

For these reasons, the second option has been chosen and a new tuning algorithm has been developed. It consists in two main steps:

1) to find the equivalent value of the river width which makes the heights of the two time responses fit;

2) to find the value of Strickler coefficient which makes the oscillations at the end of the transient of the two time responses fit.

In order to perform the first step without loosing all the information on the river shape, and so to have a more realistic model, the river width along the stream is not taken as a pure constant, but as a straight line whose parameters are tuned. In this way, the main trend (increasing or decreasing) of the width along the river is maintained and the parameters to be tuned for each reach are only the starting point and the angular coefficient of the line, in addition to the Strickler coefficient. These parameters can easily be found by running several simulations and comparing the obtained results with the reference time response of the Scicos model.

It is important to note that, due to the high uncertainty introduced by the variability of the river shape, no automatic algorithm has been used, so that the comparisons between the two responses have been not performed by a solver which minimizes an objective function, but simply by direct inspection of the results.

Once the width parameters have been tuned, the second step consists in tuning the Strickler coefficient of each reach.
Even if an automatic procedure, like the one presented in Section 5.1.3, might now work better because the widths have been correctly tuned, the overall uncertainty makes the choice of a man-made tuning be simpler and more reliable also in this case.

The results obtained by the tuning procedure for one reach are shown in Fig.5.3, where the oscillations at the beginning of the Scicos response can be due to a problem in the Scicos simulator initialization. From the figure, it is apparent that both the main dynamics and the gain of the Scicos model are properly duplicated by the Simulink environment, so that this last model can be used for the control synthesis described in the following Chapters.

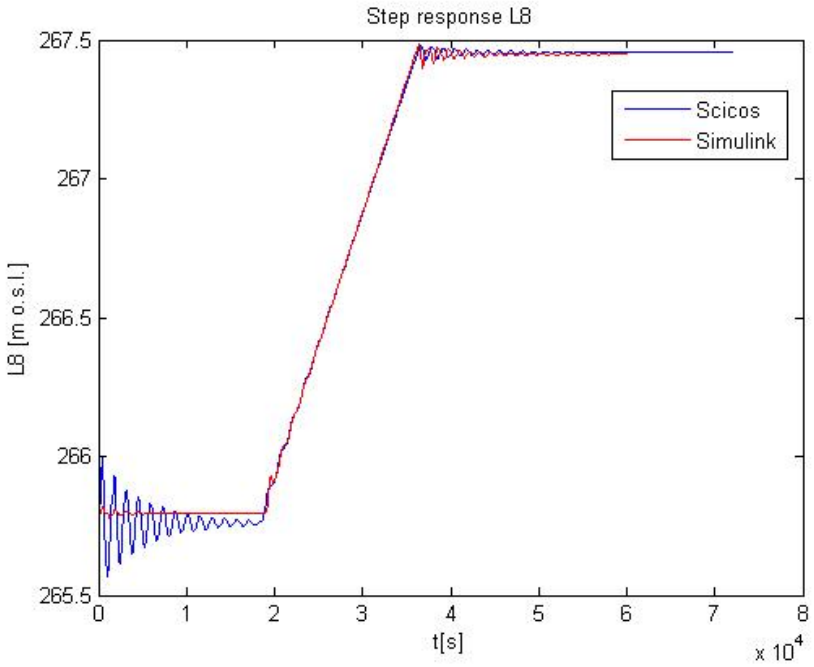

Fig. 5.3: Comparison between the Scicos and Simulink models after the tuning procedure.

Chapter 6

Linearization and Analysis

Once the nonlinear model of the hydro power valley has been built and tuned, it is possible to start the development of a MPC algorithm which uses the same model for the prediction of the future behavior of the system. However, before proceeding, it is worth to derive a linearized model first, in order to have the possibility to analyze the system with some standard tools, such as the RGA or the Singular Value analysis. In fact, these methods allow one to have a deeper understanding of the system dynamics and interactions and can help in the design of the control law. Moreover, in so doing, it will be possible to test the different solutions adopted for the controller on a linear model, with faster simulations with respect to the use of a nonlinear one. Finally, the availability of a linearized model allows one to compare the performance of the controller when using a simpler (linear) model for the prediction of the nonlinear system. For these reasons, in the following sections, the linearization of the nonlinear model of the HPV and the corresponding analysis are discussed.

6.1 Model Linearization

The first step in order to derive a linearized model is to choose the linearization point, that is the combination of states and inputs which corresponds to the plant steady state of interest. Then, the variables of the linearized model will correspond to the variations of the real variables with respect to the steady state conditions.
In the problem considered here, the steady state operating point where to perform the linearization has been provided by EDF R&D-STEP and corresponds to the same used for the model tuning and mentioned in Chapter 5.

Subsequently, the system linearization can be performed according to two different strategies:

- the Simulink tool *Timed-Based Linearization* [2] can be used; it is based on a numeric method which automatically computes the linearized system matrices through system simulation;

- the matrices of the linearized system can be manually determined by evaluating the partial derivatives of the system equations at the steady state.

Both methods have pros and cons but, due to the complexity of the system equations, the automatic (numeric) one seems easier and more suitable for our purposes. However, since it is based on an automatic procedure, it is necessary to check its results with whose produced by the second method. Therefore, both methods have been followed, by considering the first as the main one, while the second is used for validation.

6.1.1 Linearization with the Simulink Tool

In Simulink it is possible to linearize a system with the *Timed-Based Linearization* (*TBL*) tool, included into *Model Wide Utilities*. This is a Simulink block which has to be added to the Simulink model to be linearized. When the simulation is running, it calls the Matlab function *linmode* [2] and computes the matrices of the linearized system around the point reached at a specified time. In order to define the input and output variables of the linearized model to be computed, it is necessary to provide an *Input* port for each system input and an *Output* port for each output. In this way, it is possible to get the linearized system around a specified steady state by simply organizing the simulation scenario so that the system states reach the required steady state in a prescribed time and by setting the linearization time to a value close to the end of the simulation. At the end of the simulation, the *Timed-Based Linearization* block saves in the workspace a structure containing the matrices of the linearized system, which can now be used to create and simulate a new linear system, for example with a *State-Space* block.
Fig.6.1 shows how a single block describing a river reach can be linearized. The same has been done for the whole system.

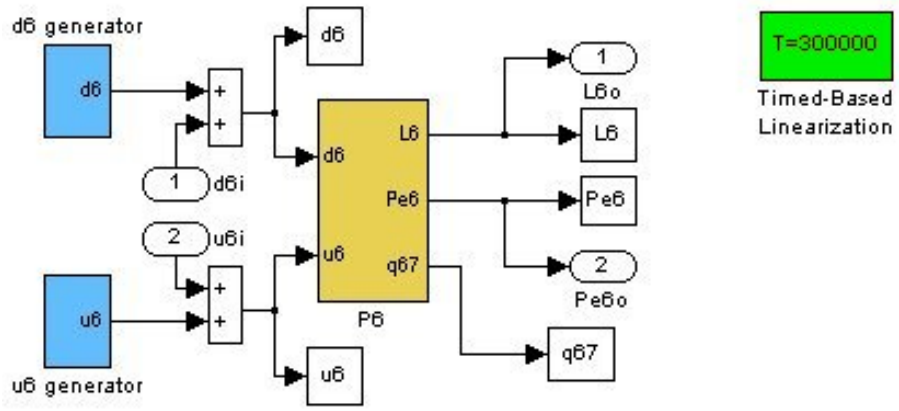

Fig. 6.1: Linearization of block *P6* with *TBL*.

6.1.2 Analytical Linearization

The linearization approach based on analytical developments implies to analyze one by one all the system equations and, for each one of them, to perform a linearization procedure at the prescribed operating point. The so obtained linear equations can be organized in a set of four matrices A, B, C and D, which define a generic linear system (6.1) and can be used to analyze and simulate it with Simulink.

$$\begin{cases} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{cases} \tag{6.1}$$

How to linearize the system equations is discussed in the following paragraphs, where they have been divided by type and rule in the system description, since each kind of equation needs specific comments.

Lakes state equations

All the state equations of the lake models are linear, so that the linearization procedure is trivial and simply consists of replacing the variables with their variations.

$$\begin{array}{rcl} \dot{L} & = & \frac{q_{in}-q_{out}}{A} \\ & \Downarrow & \\ \Delta\dot{L} & = & \frac{\Delta q_{in}-\Delta q_{out}}{A} \end{array} \tag{6.2}$$

Reaches state equations

As shown in Section 3.2, each reach is described by a number of state equations which depends on the number of cells used for discretization. These

equations can be divided in those where the state variable is the water height and those where the state variable is the flow rate.
The first ones are already linear and, as for lakes equations, they can be treated as in equation (6.3).

$$\begin{array}{rcl} \frac{\partial H_i}{\partial t} & = & -\frac{1}{W} \cdot \frac{Q_i - Q_{i-1} - Q_{lat}}{dx} \\ & \Downarrow & \\ \frac{\partial \Delta H_i}{\partial t} & = & -\frac{1}{W} \cdot \frac{\Delta Q_i - \Delta Q_{i-1} - \Delta Q_{lat}}{dx} \end{array} \tag{6.3}$$

The second ones are strongly nonlinear and their linearization requires attention. However, they also can be linearized as described in the following. Initially, the equation of the flow rate for a generic slot of discretization, already presented in equation (3.16), is reported in equation (6.4).

$$\begin{array}{rcl} \frac{\partial Q_i}{\partial t} & = & -\frac{2Q_i}{WH_i} \cdot \frac{Q_i - Q_{i-1}}{dx} + [\frac{1}{W}(\frac{Q_i}{H_i})^2 - gWH_i] \cdot \frac{H_{i+1} - H_i}{dx} + \\ & & +gWI_0H_i - \frac{gWH_i}{k_{str}^2} \cdot (\frac{W+2H_i}{WH_i})^{4/3} \cdot (\frac{Q_i}{WH_i})^2 \end{array} \tag{6.4}$$

In order to simplify the notation, the function f can be defined as in equation (6.5).

$$f = \frac{\partial Q_i}{\partial t} \tag{6.5}$$

So, the linearized equation can be written as in (6.6).

$$\frac{\partial}{\partial t}\Delta Q_i = \left.\frac{\partial f}{\partial Q_i}\right|_{ss} \cdot \Delta Q_i + \left.\frac{\partial f}{\partial Q_{i-1}}\right|_{ss} \cdot \Delta Q_{i-1} + \left.\frac{\partial f}{\partial H_{i+1}}\right|_{ss} \cdot \Delta H_{i+1} + \left.\frac{\partial f}{\partial H_i}\right|_{ss} \cdot \Delta H_i \tag{6.6}$$

where *ss* means *Steady State* and the value of each partial derivative is presented in equations (6.7).

$$
\begin{aligned}
\left.\frac{\partial f}{\partial Q_i}\right|_{ss} &= -\frac{6}{dx\cdot W}\frac{\overline{Q_i}}{\overline{H_i}} + \frac{2}{dx\cdot W}\frac{\overline{Q_{i-1}}}{\overline{H_i}} + \frac{2}{dx\cdot W}\frac{\overline{H_{i+1}}}{\overline{H_i}^2}\overline{Q_i} - \frac{2g}{k_{str}^2\cdot W}\left(\frac{1}{\overline{H_i}} + \frac{2}{W}\right)^{\frac{4}{3}}\frac{\overline{Q_i}}{\overline{H_i}} \\
\left.\frac{\partial f}{\partial Q_{i-1}}\right|_{ss} &= \frac{2}{dx\cdot W}\frac{\overline{Q_i}}{\overline{H_i}} \\
\left.\frac{\partial f}{\partial H_{i+1}}\right|_{ss} &= \frac{1}{dx\cdot W}\frac{\overline{Q_i}^2}{\overline{H_i}^2} - \frac{gW}{dx}\overline{H_i} \\
\left.\frac{\partial f}{\partial H_i}\right|_{ss} &= \frac{3}{dx\cdot W}\frac{\overline{Q_i}^2}{\overline{H_i}^2} - \frac{2}{dx\cdot W}\frac{\overline{Q_i}\,\overline{Q_{i-1}}}{\overline{H_i}^2} - \frac{2}{dx\cdot W}\frac{\overline{Q_i}^2\overline{H_{i+1}}}{\overline{H_i}^3} - \frac{gW}{dx}\overline{H_{i+1}} + \frac{2gW}{dx}\overline{H_i} + gWI_0 + \\
&\quad -\frac{g}{k_{str}^2\cdot W}\left[\frac{4}{3\overline{H_i}}\left(\frac{1}{\overline{H_i}} + \frac{2}{W}\right)^{\frac{1}{3}} + \left(\frac{1}{\overline{H_i}} + \frac{2}{W}\right)^{\frac{4}{3}}\right]\frac{\overline{Q_i}^2}{\overline{H_i}^2}
\end{aligned}
\tag{6.7}
$$

where $\overline{H_i}$ and $\overline{Q_i}$ represent the steady state values of the river height and of the flow rate in each river section i.

Power equations

As the produced and absorbed powers are computed by multiplying levels and flow rates (see equation (3.4)), which are both system variables, all the power equations are nonlinear. Their general linearization is reported in equation (6.8) and an example of how the power Pe_5 is linearized (in the case that the flow rate q_{57} is pumped and not turbined) is shown in (6.9).

$$
\begin{aligned}
P &= q\cdot K\cdot H_h \\
&\Downarrow \\
\Delta P &= \Delta q\cdot K\cdot \overline{H_h} + \Delta H_h\cdot K\cdot \overline{q}
\end{aligned}
\tag{6.8}
$$

$$
\begin{aligned}
Pe_5 &= q_{57}Kp_{57}(dZ_{57} + L_5 - Zout_{57}) + q_{58}Kt_{58}(dZ_{58} + L_5 - Zout_{58}) \\
&\Downarrow \\
\Delta Pe_5 &= (\overline{q_{57}}Kp_{57} + \overline{q_{58}}Kt_{58})\Delta L_5 + \\
&\quad + Kp_{57}(dZ_{57} + \overline{L_5} - Zout_{57})\Delta q_{57} + \\
&\quad + Kt_{58}(dZ_{58} + \overline{L_5} - Zout_{58})\Delta q_{58}
\end{aligned}
\tag{6.9}
$$

Duct equation

The last nonlinear equation computes the flow rate in the duct between Lake 5 and Lake 4 (3.5). As the chosen steady state implies that the water flows form Lake 4 to Lake 5, the sign operator and the modulus operator can be neglected and the equation is linearized as follows.

$$\begin{array}{rcl} q_{54} & = & -Ad_{54}\sqrt{2 \cdot g \cdot (L_4 - L_5)} \\ & \Downarrow & \\ \Delta q_{54} & = & -\frac{g \cdot Ad_{54}}{\sqrt{2g \cdot (\overline{L_4} - \overline{L_5})}}(\Delta L_4 - \Delta L_5) \end{array} \tag{6.10}$$

6.1.3 Comparison of the two approaches

Once the linearized model has been computed, both manually and using the appropriate Simulink tool, it is possible to compare the two approaches.
Of course, it is not sufficient to verify the matching of the two linearized models to assert that they are both reliable. As a matter of fact, it is necessary to verify also the correspondence between the linearized models and the nonlinear one, when it is used around the chosen steady state.
Due to this consideration, the test performed to evaluate the correspondence between the two linear models consists in exciting both with the same scenario of inputs, starting from the same states, and comparing the transients of the system outputs. The more the curves are overlapped, the more the two models are similar (ideally, they are expected to have the same transients). At the same time, also the nonlinear model is run and its outputs are compared with those of the linear models.

Some of the results obtained with the described test are reported in Figures 6.2 and 6.3, where the level of a lake (6.2) and the final level of a reach (6.3) given by the three models are compared. In both figures, the blue curve corresponds to the nonlinear model, the red one coincides with the manually linearized model and the green one is that of the linear model obtained by the *Timed-Based Linearization*.

From these results, it can be noticed that the correspondence between the two linearized models is always well satisfied, but they both behave as the nonlinear one only for the lakes, while they present a little difference in the part describing the reaches.
This is not surprising because, as stated in Section 6.1.2, the equations of the lakes are already linear and do not necessitate to be linearized, while the reach equations are strongly nonlinear, so that a linear approximation leads

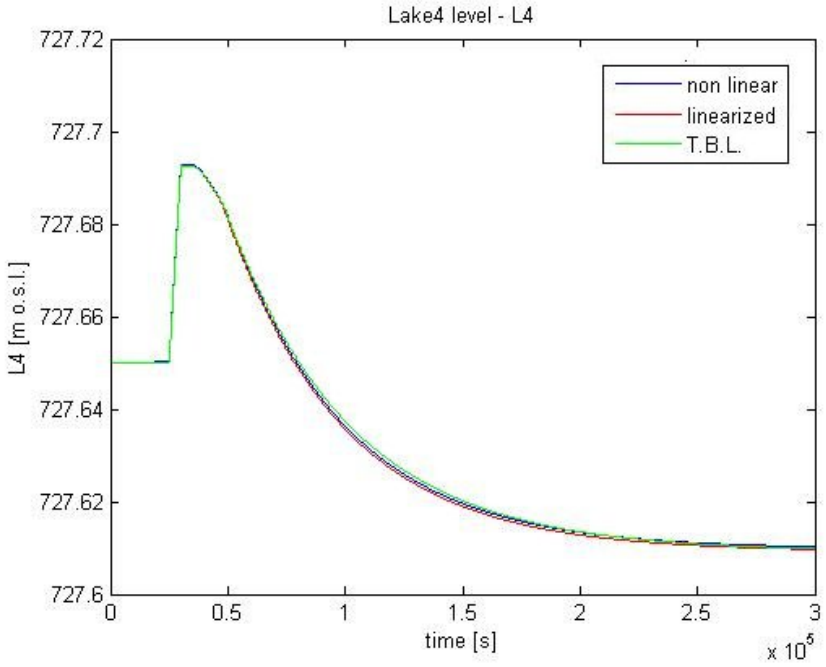

Fig. 6.2: Testing of the linearized models on a lake level.

to a non-perfect matching between the time responses. However, the difference between nonlinear and linearized models is always under reasonable values (referring to the normal range of variation of the levels of the reaches), so it can be concluded that the whole nonlinear system is well described by the linearized ones.

Furthermore, as the matching between the two linearization approaches is good, it is possible to infer that, in this case, the Simulink *Timed-Based Linearization* is reliable and that no mistakes were introduced during the manual linearization. Anyway, a more important conclusion is that it is possible to use the *TBL* for all the possible following linearizations, being reasonably sure that the returned linear system, obtained by a numerical linearization, is very close to that which we would obtain correctly linearizing the system equations. Such a knowledge makes the procedure much faster because it avoids to re-calculate all the equations of a new linear model each time it is necessary to modify a bit the nonlinear one.

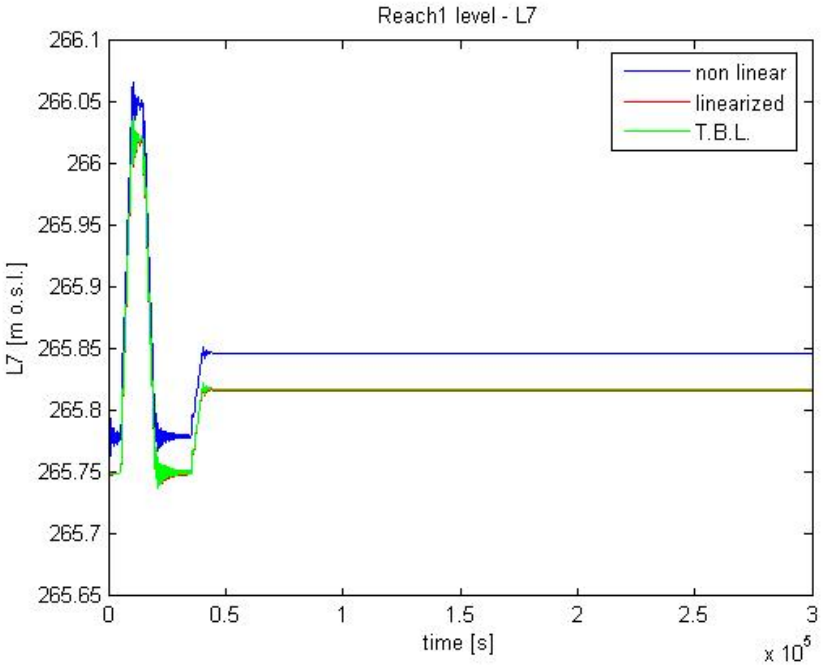

Fig. 6.3: Testing of the linearized models on a reach level.

6.2 Analysis of the Linearized Model

Before developing the control law, it is important to analyze the system to be controlled, in order to better understand its structure and to verify some expected behaviors. In this way, some useful information may be taken into account in the project of the controller. In addiction, a simple and quick analysis might point out some potential errors due to the modeling phase.
Of course, the first analysis of the implemented system was performed during the development of the software described in Chapter 4, by simulating separately each subsystem and verifying its correspondence to the underlying physical phenomena. However, the computed linearized system allows for some more and deeper analysis which imply the use of the system matrices. The first analysis performed is a direct inspection of the system matrices, which allows one to verify the number and the coupling of the variables. The second and the third one are the Relative Gain Array analysis [39] and a Singular Value analysis [28], to better understand the input-output couplings and the variation of the dynamic gain in a "Multiple Input - Multiple Output" (MIMO) system.

6.2.1 Direct Inspection of the System Matrices

The direct inspection consists in looking at the system matrices to verify if their structure and their values are coherent with the described system, pointing out potential interesting aspects.

Outputs Reduction

The first check is done on the dimensions of the matrices, that are verified to agree with the number of inputs, outputs and states of the system.
Subsequently the values of the elements of the matrices are controlled and the first, most noticeable, aspect that turns up is that the matrix D, which expresses the direct couplings between the inputs and the outputs of the system, contains some elements whose value is different from zero.
This means that some controlled variables have algebraic relationships with some inputs. So it is possible to define an instantaneous value for some outputs simply imposing a value to a given input.
This effect, in system modeling, is usually due to some neglected dynamics, which are supposed to be much faster than the nominal band of the system. In this case, since the matrices are calculated numerically by the *TBL*, such neglected dynamics may result both by a part of the system so built on purpose or by a numerical result of the linearization, due to the values given to the parameters.

Recalling the choices reported in Section 2.2, it is clear that all the controlled flow rates q_{ij} (which constitute a subset of the system outputs) are necessarily equal to their references u_i (which are the inputs) due to the hypothesis introduced at the beginning of this work. So, the presence of a set of elements different from zero in the matrix D is not surprising. This observation suggests the opportunity to reduce the dimensions of the system by a more appropriate choice of the outputs. As a matter of fact, the flow rates are not actually system outputs, as they are imposed with their references, so their values are always available and they do not need a dynamic simulation to be performed. According to this consideration, the set of selected outputs is reduced to only the powers and the levels and a further automatic linearization is performed.
The matrices so obtained are smaller than the previous ones and permit an easier treatment of the linear system for both analysis and simulation. Their

dimensions are reported below.

$$\begin{array}{lcccc} A: & (9+2N_6+2N_7+2N_8+2N_9) & \times & (9+2N_6+2N_7+2N_8+2N_9) \\ B: & (9+2N_6+2N_7+2N_8+2N_9) & \times & (12) \\ C: & (16) & \times & (9+2N_6+2N_7+2N_8+2N_9) \\ D: & (16) & \times & (12) \end{array}$$

Powers Dependence

After reducing the system matrices by eliminating the unnecessary variables, another direct inspection can be performed and, looking at the matrix D, it is immediately evident that there are still direct couplings between the inputs (now the actual flow rates) and the power outputs. These are not due to a part of the system structure which deliberately neglects some fast dynamics, but are generated by the numerical linearization and their cause can be attributed to the values assumed by the system parameters.
After an accurate inspection of the system and of the role of its parameters, such as the lake surfaces A_i, it is clear that this happens because, in the linearized model, the powers are computed as the sum of the two contributions of level and flow rate (see equations (6.8) and (6.9)) and the changes in level variations from the steady state (ΔL) are numerically much smaller (by a scale factor of 10^5) than the changes in flow rate variations (Δq) which caused them.
This fact implies that the contribution of ΔL to the variations of power (ΔP) is negligible compared to that provided by Δq.
Such a situation explains the lack of dynamic interaction between inputs (Δq) and power outputs in the system generated by the *TBL*. As a matter of fact, such a dynamic interaction is provided only by the level variations, which are dynamically linked to the flow rate variations.
Then, the transfer functions between inputs and powers are only static gains.

The previous observations lead to an important conclusion, namely that during the development of the control law it is possible to consider the powers as inputs rather than outputs of the system. As the main objective of the MPC control is to follow a power set point over a period while respecting the constraints on levels and flow rates, the controller will then have no problems in following the power set point without constraints, but all its efforts will be focused on finding those values of powers which do not move too far away from the assigned profile and that guarantee the satisfaction of all the constraints at the same time.

6.2.2 Relative Gain Array analysis

One of the simplest and most famous tools used to evaluate the couplings between the inputs and the outputs of a MIMO system is the Relative Gain Array [39]. This is a heuristic method which returns a matrix where each element value is as high as the coupling between the input corresponding to its column and the output corresponding to its row is significant. The values of the RGA matrix are independent by the adopted units of measure of the system variables.
The main advantage of the RGA is that it gives a quick information on the inputs which are suitable to be used to control some outputs and which interactions may affect the control and have to be taken into account.
Actually, as MPC is a centralized control method which makes these choices by itself with an optimization algorithm, the RGA is not very useful in order to design the controller, but it gives important information in order to check the system structure and to better understand its internal interactions.

The RGA matrix is calculated by multiplying element-wise the matrix of the static gains by its transposed inverse matrix, as in equation (6.11).

$$\Lambda = G(0) \odot (G(0)^{-1})^T \tag{6.11}$$

where Λ is the RGA, $G(0)$ is the static gain matrix and $\odot$ means the element by element product.
It is important to notice that the RGA actually gives information only on a system in static conditions, while no information is provided at other frequencies. However, this is often enough to have good suggestions on how to assign the control variables to the controlled ones in a decentralized control scheme. Finally, it has to be noticed that $G(0)$, and so the general transfer matrix $G(s)$, must be square and without integrators, otherwise it is impossible to calculate the inverse matrix.

In order to compute the RGA associated to the HPV linearized model and taking into account the previous considerations on the direct dependence of the power on the input flow rates, it is natural to focus the RGA analysis only on that part of the system which links the flow rates to the levels of lakes and reaches. So, the analyzed system for the RGA has 9 inputs and only 9 outputs, that makes the transfer matrix square.
Nevertheless, it is still not possible to compute the RGA, since the transfer function matrix contains pure integrators, which represent the water accumulation phenomenon in lakes and reaches. To overcome this problem, two main ideas may be followed:

- to multiply the whole transfer matrix by the Laplace operator s and to proceed with the normal RGA calculation;

- to compute a RGA with a gain matrix not computed in static conditions, but at a low frequency with respect to the main system dynamics, for example $\frac{1}{3600}Hz$.

Both solutions have been considered and in both cases it happens that the gain matrix is still singular, even if its elements are no longer infinite. Now the singularity cannot be due to the integrators, but to the interactions between the variables. Therefore, a further inspection on the system and on the new gain matrix has been done.

In particular, the transfer matrix at frequency $\frac{1}{3600}Hz$ has been computed and analyzed. It is reported in Fig.6.4, where the rows represent the level outputs L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8 and L_9 and the columns represent the flow rate inputs q_{12}, q_{23}, q_{34}, q_{57}, q_{58}, q_{67}, q_{78}, q_{89} and q_{out}.

0.0945	0	0	0	0	0	0	0	0
0.3075	0.3075	0	0	0	0	0	0	0
0	0.0110	0.0110	0	0	0	0	0	0
0	0	0.1024	0.0942	0.0942	0	0	0	0
0	0	0.0942	0.7328	0.7328	0	0	0	0
0	0	0	0	0	0.0540	0	0	0
0	0	0	0.1682	0	0.1682	0.1682	0	0
0	0	0	0	0.2149	0	0.2149	0.2149	0
0	0	0	0	0	0	0	0.1213	0.1213

Fig. 6.4: Transfer matrix evaluated at $\frac{1}{3600}Hz$.

This analysis leads to conclude that the reason of the singularity is that the two input flow rates q_{57} and q_{58} act in the same way on the outputs L_4 and L_5, creating a singular sub-matrix in the transfer matrix (elements (4,4), (4,5), (5,4) and (5,5) in Fig.6.4).
In confirmation of the previous statement, it is verified that removing one row and one column associated to one of those variables, makes the condition number of the gain matrix to decrease by a factor of 10^{17}.

Therefore, it is reasonable to not analyze a further reduced RGA, but to look only at the transfer matrix evaluated in the selected frequency and presented in Fig.6.4.
From that matrix, the following statements can be inferred:

- each level is correctly affected in the same way by the two inlet and outlet flow rates;
- for Lake 4 and Lake 5 this does not occur because of the existing uncontrolled connection between them;
- flow rates q_{57} and q_{58} give the same contribution to levels L_4 and L_5 and this fact causes the mentioned singularity of the gain matrix;
- the expected decoupling between the variables which describe a subsystem upstream the valley and those relative to a downstream one is verified as the whole matrix is almost diagonal.

Then, all the conclusions derived by the RGA analysis confirm that the system is reliable and add some further information on the already available knowledge.

6.2.3 Singular Value Analysis

For "Single Input - Single Output" (SISO) systems, the analysis of the dependence of the gain of a given transfer function on the frequency is usually performed through the Bode diagrams, which plot the modulus and the phase of the Fourier transform of the frequency response $G(j\omega)$ of the system versus the frequency.
For MIMO systems this approach cannot be directly applied, since the SISO transfer functions (and their Bode diagrams) between the single input-output pairs do not fully represent the plant behavior. However, some useful information can be provided by the singular value analysis [28], which allows to define and analyze the system gain at different frequencies also in the MIMO case by resorting to the properties of the induced norm of a generic matrix and to the singular value decomposition.

Singular Value Decomposition

It can be proven that each matrix Φ, either square or not, can be decomposed into three different matrices as in equation (6.12) [28] [44]:

$$\Phi = U\Sigma V \tag{6.12}$$

where U and V are unitary matrices and, if Φ is square, Σ is a square diagonal matrix whose main diagonal contains real positive values arranged in a

decreasing order, as shown in equations (6.13) and (6.14). These values are named "singular values" and are univocally associated to the matrix Φ.

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \sigma_n \end{bmatrix} \tag{6.13}$$

$$\overline{\sigma} = \sigma_1 > \sigma_2 > \ldots > \sigma_n = \underline{\sigma} \tag{6.14}$$

It is useful for our purposes to define the condition number of the matrix Φ as the ratio between the maximum singular value and the minimum one, as in equation (6.15). Such a ratio is also important to evaluate the required computational burden in order to perform the numeric inversion of Φ.

$$\gamma = \frac{\overline{\sigma}}{\underline{\sigma}} \tag{6.15}$$

Induced Norm

The induced *p-norm* is only one of the several possible definitions for a norm of a matrix [10] and is defined in the following equation (6.16).

$$\| \Phi \|_{i_p} = \sup_{w \neq 0} \frac{\| \Phi \cdot w \|_p}{\| w \|_p} \tag{6.16}$$

where w is a generic vector whose elements can be either real or complex. One of the most useful properties of this norm is that the induced 2-norm of a matrix Φ corresponds to its maximum singular value and its minimum one can be also obtained by a similar definition:

$$\overline{\sigma}(\Phi) = \| \Phi \|_{i_2} = \sup_{w \neq 0} \frac{\| \Phi \cdot w \|_2}{\| w \|_2} \tag{6.17}$$

$$\underline{\sigma}(\Phi) = \inf_{w \neq 0} \frac{\| \Phi \cdot w \|_2}{\| w \|_2} \tag{6.18}$$

Equations (6.17) and (6.18) lead to (6.19), which is an important result that allows one to perform the singular value analysis.

$$\underline{\sigma}(\Phi) \leq \frac{\| \Phi \cdot w \|_2}{\| w \|_2} \leq \overline{\sigma}(\Phi) \tag{6.19}$$

Singular Value Analysis

The Frequency Response Theorem [39] states that if a linear and asymptotically stable SISO system whose transfer function is $G(j\omega)$ receives as input a generic sinusoidal signal with amplitude U and frequency ω, after an initial transient the system output signal must have the same frequency as the input with a phase shift equal to the phase of $G(j\omega)$ and must have an amplitude equal to the product between U and the modulus at ω of the transfer function (see equation (6.20)).

$$U(j\omega) = Usin(\omega t) \quad \Rightarrow \quad Y(j\omega) = U|G(j\omega)|sin(\omega t + \angle(G(j\omega))) \tag{6.20}$$

Then, the gain in frequency of a SISO system can be written as in the following equation (6.21).

$$|G(j\omega)| = \frac{|Y(j\omega)|}{|U(j\omega)|} \tag{6.21}$$

Similarly, for a MIMO system a sort of gain in frequency can be defined as the ratio between the 2-norm of the output vector and that of the input vector, as shown in equation (6.22).

$$\frac{\parallel Y(j\omega) \parallel_2}{\parallel U(j\omega) \parallel_2} = \frac{\parallel G(j\omega) \cdot U(j\omega) \parallel_2}{\parallel U(j\omega) \parallel_2} \tag{6.22}$$

This equation, recalling (6.19), leads to:

$$\underline{\sigma}(G(j\omega)) \leq \frac{\parallel G(j\omega) \cdot U(j\omega) \parallel_2}{\parallel U(j\omega) \parallel_2} \leq \overline{\sigma}(G(j\omega)) \tag{6.23}$$

Such a relationship allows one to infer two main conclusions:

- the gain of a MIMO system does not depend only on the system, but also on the direction of the input vector U;
- such a gain is upper and lower bounded by the maximum and by the minimum singular values of the frequency-variant transfer matrix.

Then, the plot of the singular values $\underline{\sigma}(G(j\omega))$ and $\overline{\sigma}(G(j\omega))$ versus frequency gives an information which is similar to that provided by a magnitude Bode diagram for SISO systems, that is the range of variation of the gain of the MIMO system with the different possible combinations of the input signals.
This is a practical way to analyze the linear model and to verify if it respects

the potential constraints for the gain at prescribed frequencies (typically a higher gain is required at the working frequencies, while a lower gain is expected at high frequencies in order to minimize the high-band disturbances).

According to the definition presented in equation (6.15), the frequency-dependent condition number can be calculated as follows:

$$\gamma(G(j\omega) = \frac{\overline{\sigma}(G(j\omega)}{\underline{\sigma}(G(j\omega)} \tag{6.24}$$

The more $\gamma(G(j\omega)$, evaluated at one frequency $\overline{\omega}$, is close to 1, the more the gain of the MIMO system at that frequency does not depend on the inputs.

Within the Matlab environment, two useful functions which automatically compute and plot the singular values are provided:

- *svd.m* implements the singular value decomposition; it computes the singular values of a prescribed matrix of defined elements;
- *sigma.m* plots with respect to frequency the diagram of all the singular values of a linear dynamic system expressed in its transfer matrix form.

The singular value analysis of the considered linear system is reported in Fig.6.5. Looking at this figure, two considerations can be done:

- the general slope of the diagram accounts for the presence of pure integrators, which imply that the static gain is $+\infty$ and the initial slope is $-20dB/dec$;
- most of the singular values are close to the greater one, while the lower is very close to zero with respect to the others.

The second observation is very interesting and deserves a deeper analysis. As all the lake surfaces have, more or less, the same order of magnitude and also the imposed variations to the flow rates are quite similar to each other, it were expected to find a gain which does not change considerably with the input distribution and then to have a frequency plot of the singular values described by a narrow beam of curves.
Such a configuration is instead valid only for all except one singular values, while the smallest one seems to be close to zero. This fact implies that the

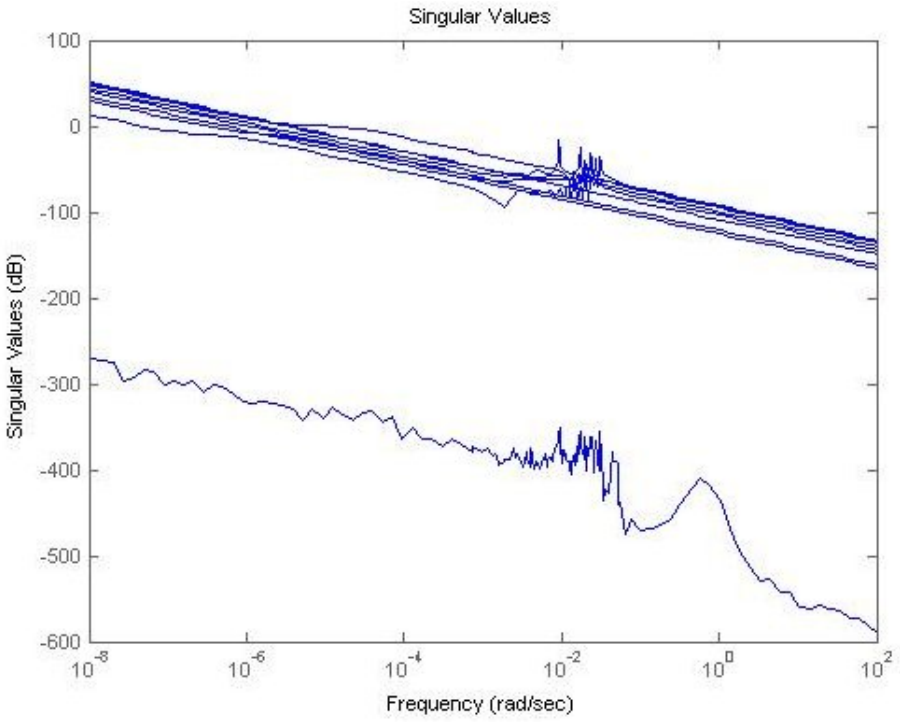

Fig. 6.5: Result of the singular value analysis.

gain of the system normally assumes values which belong to a precise and not-so-wide range, while there are some particular configurations of the input variables which make the gain to be very low, so that the outputs are not affected by them.

In order to understand and check this result, a further inspection of the system has been performed and an example of input vector which does not make the outputs move has been found. As a matter of fact, if a positive quantity Δq is assigned to the system inputs as follows:

$$\Delta q_{57} = -\Delta q \qquad \Delta q_{78} = -\Delta q \qquad \Delta q_{58} = +\Delta q$$

while all the other inputs are null, the whole set of outputs does not change, that is all the variations of the levels are zero.
Then, the behavior of the minimum singular value shown in Fig.6.5 can be explained with the presence of a double route for the water in order to arrive to the third reach from the fifth lake. Such a particularity of the system structure explains the reason why there exists a configuration of the input vector which makes the system gain to be so low.

The system analysis is concluded and no particularly critical problems have been met. However, several interesting information about the system have been found and an overall check has been performed.
Therefore, it is possible to proceed with the development of the MPC controller for the hydro power valley described by the analyzed model.

Chapter 7

Model Predictive Control

This chapter presents the Model Predictive Control (MPC) strategy adopted to control the hydro power valley considered in this Thesis.
Initially, the overall idea of MPC and its most widely used implementation for control of linear systems [28] are presented and commented. Then, a new MPC algorithm for nonlinear systems is discussed, with particular emphasis on the problems related to the choice of a suitable optimization solver.
Afterwards, a few ways to provide an integral action with a MPC controller are presented, discussing both classical and new strategies.
Finally, the application of the developed controller to the system described in the previous chapters is reported and its results are discussed.

7.1 The Model Predictive Control approach

Model Predictive Control is currently one of the most widely used advanced control technologies for process plants [35] [8] [22] [31] [28].
The history of this technology originates from the industrial world, as MPC was first implemented in industry long before a thorough understanding of its theoretical properties was available. As a matter of fact, academic interest in MPC started growing in the mid eighties and the understanding of MPC properties has now built a strong conceptual and practical framework for both practitioners and theoreticians, even if several issues are still open.

The main idea of the MPC strategy is to generate the control inputs for the controlled system as solutions of a real-time optimization problem, which consists in finding the vectors of control inputs over a time horizon which minimize a prescribed cost function.

Assume that the system under control is described by the discrete-time and a priori nonlinear model

$$x(k+1) = f(x(k), u(k)) \tag{7.1}$$

where x and u are the state and the control vectors, respectively.
For system (7.1), assuming that the state x is available, the simplest cost function considered in the development of MPC is quadratic and defined as in equation (7.2).

$$J = \sum_{i=0}^{H-1} (\|x(k+i)\|_Q^2 + \|u(k+i)\|_R^2) + \|x(k+H)\|_S^2 \tag{7.2}$$

At each discrete sampling time k, the value of the function J is computed as the sum of the squared norms of the states x and control inputs u, weighted by the positive definite matrices Q and R, over the prediction horizon H. The state of the terminal state, i.e. of the state at the end of the prediction horizon, is weighted in a different way through a square positive semidefinite matrix S.
In the problem formulation, a set of constraints on the control variables and on the system outputs are often considered, including control limits due to saturations of the actuators or state limits representing physical constraints.

Therefore, the optimization problem is composed by several elements:

- the process model (prediction model), which is used to compute the prediction of the controlled variables over a defined time interval, called "prediction horizon", knowing the current set of system states and inputs;

- the process measurements, which provide the information on the variables of the real system, in particular on the current state of the system, which is used to initialize the simulation of the prediction model;

- the cost function, which contains the future inputs, the states and the outputs computed through simulation of the prediction model, over the whole prediction horizon;

- the constraints on the control variables and on the system states and outputs to be taken into account during the optimization.

Once the optimal set of future inputs over the prediction horizon has been computed, only the vector corresponding to the current time slot k is really applied to the system. Afterwards, in the next (discrete) time instant, the full procedure is re-iterated, according to a *receding horizon* strategy. At each iteration, the prediction horizon is shifted forward by one time instant, so that the prediction is always computed over a prediction horizon of the same length and the control law resulting from the application of MPC becomes time-invariant, although implicitly defined.

In order to reduce the computational burden of the optimization problem to be solved at any sampling time, it is possible to optimize only with respect to a subset of the $u(k+i)$ vectors, by defining the "control horizon" H_c. Then, the algorithm minimizes only with respect to the vectors from $u(k)$ to $u(k + H_c)$, while those from $u(k + H_c + 1)$ to $u(k + H - 1)$ are maintained equal to $u(k + H_c)$. In this way, the number of vectors to be optimized is not longer H but H_c, which is obviously smaller.

Fig.7.1 [35] resumes the procedure described above and shows the structure of a typical MPC implementation.

7.1.1 Pros and cons of the MPC strategy

The main advantages of MPC, that explain its wide use in the process industry, are discussed below [28]:

- in its simplest formulations, the MPC architecture is very easy to implement in a real control application;
- its parameters can often be tuned easily with a few experiments on the system;
- it manages systems with a high number of variables (inputs, outputs, states) and parameters, a typical situation in the process industry;
- it takes automatically into account also the saturations on the control variables and on the system outputs;
- it allows to choose between different cost functions and also allows the final user to define them, according to the various possible objectives of the control;

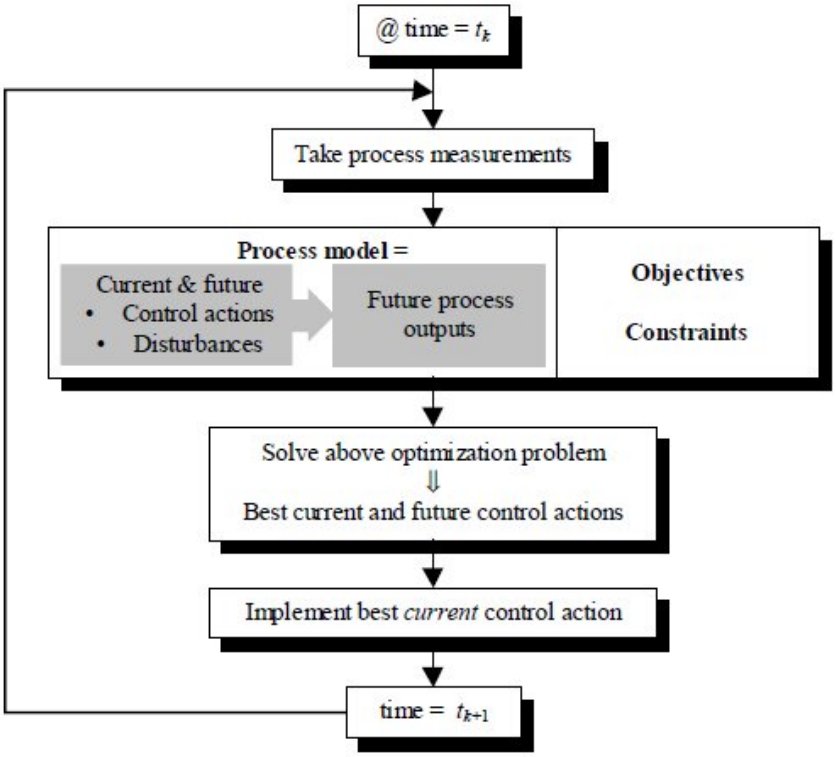

Fig. 7.1: General scheme of MPC.

- also economic cost functions can be minimized, so economic problems (for example the minimization of the cost of the overall production) are considered directly inside the control law, during the on-line choice of the control inputs.

On the other hand, as the optimization procedure is performed at each sampling time, the MPC strategy is not indeed advisable for those applications with fast dynamics requiring short sampling times, such as automotive or robotics, but it is well suitable for slow systems, such as those in the process industry (chemical, petrochemical, hydro, gas, ...).
For these reasons, the MPC strategy seems to be a promising approach for controlling the hydro power valley considered in this Thesis.

7.1.2 MPC for linear systems

If the controlled system can be well described by a linear model, this model can be used for prediction in the MPC structure.

In this case, under the hypotheses that the cost function is quadratic like the one presented in equation (7.2) and that there are no constraints, the optimization problem has an explicit solution and does not need any optimization engine to be solved. This makes the algorithm much faster, because at each sampling time the optimal set of vectors of the control inputs can be calculated in a closed form.
As a matter of fact, if the system is linear (see (7.3)), there are no constraints and the cost function can be expressed in a matrix form as in (7.4), MPC corresponds to a generic problem of linear quadratic optimal control [28].
In particular, let the system be described by:

$$\begin{cases} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{cases} \tag{7.3}$$

and the performance index be:

$$J = \sum_{i=0}^{H-1} (x(k+i)^T Q x(k+i) + u(k+i)^T R u(k+i)) + x(k+H)^T S x(k+H) \tag{7.4}$$

Then, the minimization of (7.4) can be directly executed in two ways:

1) the iterative approach;

2) the matrix-based approach.

The iterative approach consists in computing the coefficient K of the control law $u(k) = -Kx(k)$ by the backward iterative evaluation of the Riccati equation [28] [45]. The algorithm is reported below.

- $P(H) = S$
- *for i=H-1 to 1 do* $\begin{cases} K(i) &= (R + B^T P(i+1) B)^{-1} B^T P(i+1) A \\ P(i) &= A^T P(i+1) A + Q - A^T P(i+1) B K(i) \\ u(k+i) &= -K(i) x(k+1) \end{cases}$
- $K(0) = (R + B^T P(1) B)^{-1} B^T P(1) A$
- $K = K(0) \quad \Rightarrow \quad u(k) = -Kx(k)$

The matrix-based approach, instead, allows for a one-shot calculation of all the vectors of the optimal control variables over the prediction horizon with a matrix calculus.

This is possible if the system matrices are properly arranged into bigger matrices, which multiply the input and the state vectors enlarged to cover all the prediction horizon, according to the discrete Lagrange equation (7.5) [39].

$$x(k+1) = A^i x(k) + \sum_{j=0}^{i-1} A^{i-j-1} B u(k+j), \qquad i > 0 \tag{7.5}$$

The equation which describes the behavior of the system in the prediction horizon is presented in (7.6), that can be easier written as in (7.7).

$$\begin{bmatrix} x(k+1) \\ x(k+2) \\ x(k+3) \\ \dots \\ x(k+H) \end{bmatrix} = \begin{bmatrix} A \\ A^2 \\ A^3 \\ \dots \\ A^H \end{bmatrix} x(k) + \begin{bmatrix} B & 0 & 0 & \dots & 0 \\ AB & B & 0 & \dots & 0 \\ A^2B & AB & B & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ A^{H-1}B & A^{H-2}B & A^{H-3}B & \dots & B \end{bmatrix} \begin{bmatrix} u(k) \\ u(k+1) \\ u(k+2) \\ \dots \\ u(k+H-1) \end{bmatrix} \tag{7.6}$$

$$X(k) = \mathbf{A}x(k) + \mathbf{B}U(k) \tag{7.7}$$

where $X(k)$ and $U(k)$ are the extended column vectors which contain all the future states and inputs, while $\mathbf{A}$ and $\mathbf{B}$ are the extended matrices built as shown in (7.6).
Similarly, also the equation of the cost function can be written in an extended matricial form. Defining the matrices $\mathbf{Q}$ and $\mathbf{R}$ as in (7.8), it is possible to obtain the expression (7.9).

$$\mathbf{Q} = \begin{bmatrix} Q & 0 & 0 & \dots & 0 \\ 0 & Q & 0 & \dots & 0 \\ 0 & 0 & Q & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & S \end{bmatrix} \qquad \mathbf{R} = \begin{bmatrix} R & 0 & 0 & \dots & 0 \\ 0 & R & 0 & \dots & 0 \\ 0 & 0 & R & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & R \end{bmatrix} \tag{7.8}$$

$$\begin{aligned} J &= X(k)^T \mathbf{Q} X(k) + U(k)^T \mathbf{R} U(k) \\ &= (\mathbf{A}x(k) + \mathbf{B}U(k))^T \mathbf{Q} (\mathbf{A}x(k) + \mathbf{B}U(k)) + U(k)^T \mathbf{R} U(k) \\ &= x(k)^T \mathbf{A}^T \mathbf{Q} \mathbf{A} x(k) + 2x(k)^T \mathbf{A}^T \mathbf{Q} \mathbf{B} U(k) + U(k)^T (\mathbf{B}^T \mathbf{Q} \mathbf{B} + \mathbf{R}) U(k) \end{aligned} \tag{7.9}$$

The performance index (7.9) has a quadratic form, then its minimum can be easily found by solving the equation obtained setting to zero its derivative with respect to the vector U(k), see (7.10).

$$\frac{\partial J}{\partial U} = 2U(k)^T(\mathbf{B}^T\mathbf{Q}\mathbf{B} + \mathbf{R}) + 2x(k)^T\mathbf{A}^T\mathbf{Q}\mathbf{B} = 0 \tag{7.10}$$

Then, for linear systems and under the hypotheses listed above, the control law can be directly obtained as shown in (7.11).

$$\begin{array}{rcl} U(k) & = & -(\mathbf{B}^T\mathbf{Q}\mathbf{B} + \mathbf{R})^{-1}\mathbf{B}^T\mathbf{Q}\mathbf{A}^T x(k) \\ & = & -\mathbf{K}x(k) \\ & \Downarrow & \\ \begin{bmatrix} u(k) \\ u(k+1) \\ \dots \\ u(k+H-1) \end{bmatrix} & = & -\begin{bmatrix} \mathbf{K}(0) \\ \mathbf{K}(1) \\ \dots \\ \mathbf{K}(H-1) \end{bmatrix} x(k) \end{array} \tag{7.11}$$

According to the receding horizon principle, only the first element of the vector $U(k)$, is really used, see (7.12).

$$u(k) = -\mathbf{K}(0)x(k) \tag{7.12}$$

When constraints are present, such as those in (7.13), the problem becomes a finite-horizon optimization problem with constraints, which can no longer be directly solved in a closed form but needs to be solved numerically.

$$U_{min} < U(k) < U_{max} \tag{7.13}$$

7.1.3 MPC for nonlinear systems

If the model of the controlled system is nonlinear and if its linearized model does not provide a satisfactory prediction of the real process behavior, it is necessary to use a nonlinear prediction model. The corresponding optimization problem is no longer linear and there is not any explicit solution to be used as in the previous case. Therefore, an appropriate optimization solver is required.
The choice of the solver can be difficult, since it depends on the problem at hand and on the required performance of the control. To this regard, various licensed solvers are available and also the Matlab environment provides some of them in the *Optimization Toolbox*.

While the main advantage of implementing a nonlinear MPC is that the prediction of the system variables is more accurate, its worst disadvantage is that the solver must be able to find acceptable solutions of a non convex, and often very difficult due to nonlinearities, optimization problem.

7.2 Definition and implementation of the MPC controller

In Chapter 6 a linearized model of the hydro power valley at a given steady state has been derived and its correspondence with the nonlinear model has been analyzed and verified. Therefore, we could assume that a linear model is available for the synthesis of MPC.
However, one of the objectives of this Thesis is to develop a quite general MPC controller with the following properties:

- since the considered hydro power valley is only a case study, the controller should not be strictly related to the case at hand, but it should be possible to use it also with different systems, or with the same system but in different working conditions;
- it must allow to change the cost function according to the current control objectives, because the required behavior of the system may change after the controller has been built; for example it may be asked to follow both local or global power references while respecting the constraints on levels and flow rates;
- it must be able to manage both linear and nonlinear constraints;
- its structure must be enough flexible to permit a further modification in order to implement a distributed control architecture.

These properties can be achieved only with a nonlinear MPC structure, which has been chosen to develop a nonlinear MPC controller.

This choice is significant because, while a linear MPC controller already built is available in the Matlab *MPC Toolbox*, no software for nonlinear MPC is available for free and the only way to get the required controller is to entirely build it from scratch.
As the model has been built in Simulink, the Matlab-Simulink environment has been chosen also for the development of the controller.
Even if the optimization solver, which is the core of a nonlinear MPC architecture, can be chosen between those provided by the Matlab *Optimization*

Toolbox, a different software (Tomlab [3] [25]) has been chosen for this purpose, for the reasons discussed in the following Section 7.2.3.

Therefore, a full MPC controller has been created, following the guidelines for the development of classical MPC algorithms presented in Section 7.1. The structure of the controller and the chosen control objective functions are presented in the following paragraphs.

7.2.1 Choice of the objective functions

Even if the developed controller is able to work with any cost function that the final user may define, two main optimization criteria are expected to be used for the hydro power valley [17]:

- to follow several local set points of power, one for each power plant of the valley, while respecting the assigned constraints;
- to follow only one global set point of power, which defines the desired trajectory of the total electric power produced in the valley, while respecting the assigned constraints.

In both cases, the classic objective function presented in (7.2) has to be changed into the "trajectory tracking" form, which does not minimize the system state vectors $x(k+i)$, but the difference between the controlled output vectors and the imposed set point $(y(k+i) - SP(k+i))$.
The trajectory tracking form corresponding to (7.2) is presented in (7.14).

$$J = \sum_{i=0}^{H-1} (\|y(k+i) - SP(k+i)\|_Q^2 + \|u(k+i)\|_R^2) + \|x(k+H)\|_S^2 \quad (7.14)$$

Local set points

When this reference is chosen, a matrix containing the required values for each power in each time slot of the prediction horizon has to be defined during the initialization of the controller.
Then, the corresponding cost function J to be minimized is defined as follows.

$$J = \sum_{i=0}^{H} \left((y_P(k+i) - SP_{loc}(k+i))^T Q_P (y_P(k+i) - SP_{loc}(k+i)) + u(k+i)^T R u(k+i) \right) \quad (7.15)$$

where:

- $y_P(k+i)$ is the column vector of the predicted power outputs of the sampling time $k+i$;
- $SP_{loc}(k+i)$ is the column vector of local set points at time $k+i$;
- Q_P is the diagonal matrix of the weights of the predicted power errors;
- R is the diagonal matrix of the weights of the future inputs.

Global set point

If the reference is expressed as a global power request, the set point is a vector whose values represent the time variation of the reference.
Then, the corresponding cost function J to be minimized is defined in (7.16).

$$J = \sum_{i=0}^{H}((\sum_{j=1}^{npower} y_P(k+i,j) - SP_{glob}(k+i))^2 q_P + u(k+i)^T R u(k+i)) \quad (7.16)$$

where:

- *npower* is the number of power outputs;
- $y_P(k+i,j)$ is the generic element of the vector of predicted power outputs $y_P(k+i)$;
- $SP_{glob}(k+i)$ is the global scalar set point at time $k+i$;
- q_P is the scalar weight of the global power error.

Constraints

During the minimization procedure of the selected cost function J, the solver must also verify that the solution fulfills the imposed constraints, which define the upper and lower limits for some system variables.
In the considered case [17], it is required to bound the flow rates u_i, their derivatives $\frac{du_i}{dt}$ and the levels of lakes and reaches L_i. Then, the constraints to be included in the optimization problem are:

$$\begin{aligned} \underline{L_i} &\leq L_i \leq \overline{L_i} \\ \underline{u_i} &\leq u_i \leq \overline{u_i} \\ \underline{v_i} &\leq \tfrac{du_i}{dt} \leq \overline{v_i} \end{aligned} \quad (7.17)$$

where:

- $\underline{L_i}$ and $\overline{L_i}$ are the lower and upper bounds for the levels L_i;
- $\underline{u_i}$ and $\overline{u_i}$ are the lower and upper bounds for the flow rates u_i;
- $\underline{v_i}$ and $\overline{v_i}$ are the lower and upper bounds for the flow rate variations $\frac{du_i}{dt}$.

7.2.2 Controller Structure

In the development of a new controller using a software environment, the first step is to divide the control algorithm into several sections and to assign each one of them to a specific function or subfunction of the software.
Thus, a conceptual scheme of the implementation of the procedure discussed in Section 7.1 in the Matlab-Simulink environment is presented in Fig.7.2.

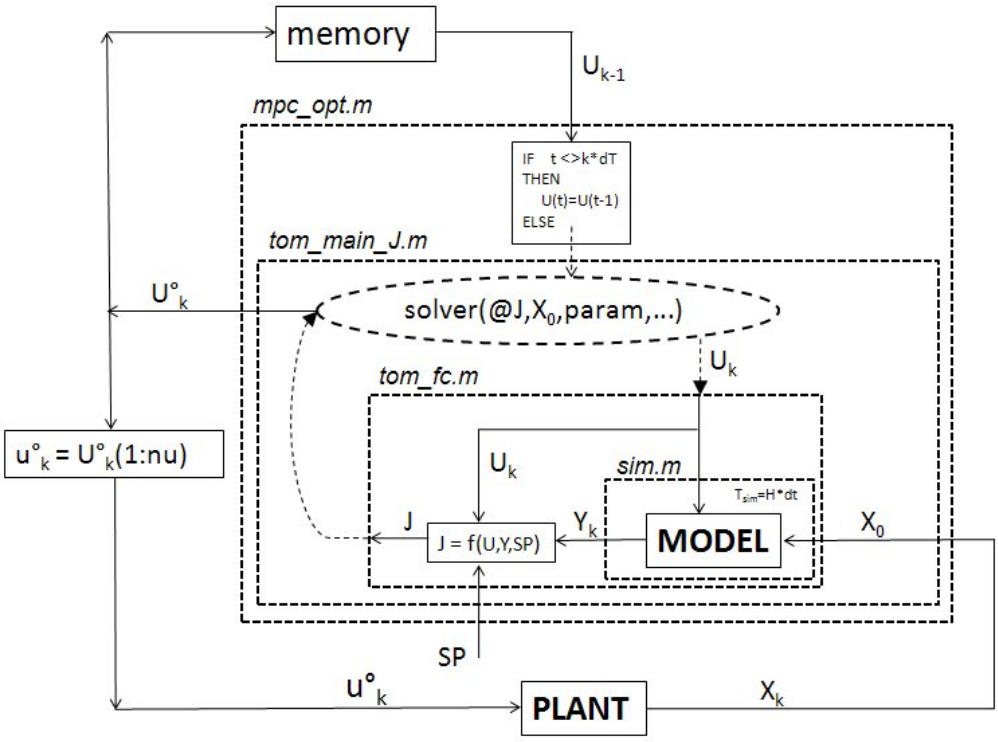

Fig. 7.2: Conceptual scheme of the developed MPC controller.

The Matlab functions which compose the controller are listed below in hierarchical order, from the higher to the lower one:

1) *hpvdata.m* initializes the parameters of the plant;

2) *mpc_data.m* initializes the parameters of the controller;

3) *mpc_opt.m* manages the interface of the discrete-time controller with the continuous-time system;

4) *tom_main_J.m* sets and solves the optimization problem;

5) *tom_fc.m* evaluates the cost function each time it is started by *tom_main_J.m*.

Initialization

The MPC algorithm described in Section 7.1 presents some parameters, to be chosen by the final user, which define the overall performances of the controller. These parameters are:

- dT is the sampling time of the controller measured in seconds (control interval);
- H is the prediction horizon, measured in multiples of dT;
- Hc is the control horizon, measured in multiples of dT;
- nu is the number of the control variables;
- $nlevel$ is the number of the level outputs of the system;
- $npower$ is the number of the power outputs of the system;
- ny is the total number of the system outputs.

Moreover, the local or global set point have to be defined before starting the simulations.
The Matlab function *mpc_data.m* contains all the required definitions and has to be called before running the simulations together with *hpvdata.m*, which defines the parameters of the system.
The main parts of the script *mpc_data.m* are reported below as an example of configuration.

```
% MPC parameters definition
dT=7200;        % [s]control interval
H=5;        % [dT intervals] prediction (receding) horizon
Hc=5;        % [dT intervals] control horizon
nu=9;        % plant inputs
nlevel=9;        % level outputs
npower=7;        % power outputs
ny=nlevel+npower;        % total plant outputs

nY=ny*H;
nU=nu*H;

% system inputs definition (differential values from steady state)
U0=zeros(1,nU);        % initial inputs vector
dist=zeros(1,3);        % constant disturbs: d1 d6 d9

% constant global set point definition (value from steady state)
SPgl=50e6;

% constant local set points definition (values from steady state)
% definition order:L1,L2,L3,L4,L5,L6,L7,L8,L9,Pe2,Pe3,Pe5,Pe6,Pe7,Pe8,Pe9
SP1=[0 0 0 0 0 0 0 0 0 0 0 50 0 0 0 0]*1e6; % set point at the initial time
SP=zeros(1,nY);
for i=1:H
SP(1,(i-1)*ny+1:i*ny)=SP1;        % set point at time k+i
end
```

Continuous-Discrete Interface

As the controller is expected to be as flexible as possible, in order to allow its use in different contexts, it is worth to include in the software an interface layer which makes the choice of the discrete sampling time of the controller independent from that of the controlled system.
Such a decoupling has been obtained by an event-check strategy.
An opportune set of Simulink blocks (Fig.7.3) generates an event *ev* each time the simulation time is equal to a multiple of the MPC sampling time dT. The function *mpc_opt.m* at each simulation step checks the event *ev*: if it is "FALSE ", the previous value of the input vector $U(k-1)$ is maintained, otherwise the function *tom_main_J.m* is called to compute the new optimal vector of $U^o(k)$.
The result is that, according to the MPC algorithm, the controller changes

its action on the controlled process only at the end of any control interval dT, and maintains it constant during the interval.

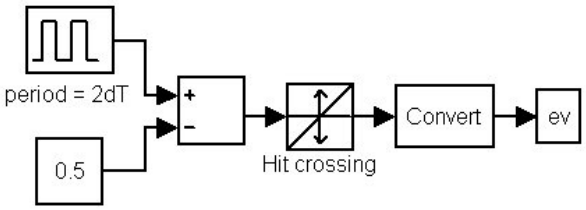

Fig. 7.3: Event generator.

The main parts of the function *mpc_opt.m* are reported below:

```
function out = mpc_opt(u,dT,H,Hc,nU,nX,nY)

Up=u(1:nU);     % previous output (manipulated variables)
cnt=u(nU+1);    % counter
ev=u(nU+2);     % event: if ev=1 compute new output (manipulated variables)
SP=u((nU+2)+1:(nU+2)+nY);        % horizon set point
Xp=u((nU+2+nY)+1:(nU+2+nY)+nX); % plant states of the current sample time

if ev==1
U_opt=tom_main_J(Up,Xp,H,Hc,dT,nU,nX,nY,SP);
U=U_opt;
cnt=cnt+1        % plots the optimization sequential number
else
U=Up;
end

out(1:nU) = U;        % manipulated variables
out(nU+1) = cnt;      % time of the previous optimization
```

Solution of the optimization problem

The function *tom_main_J.m* has as inputs the previous control vector $U(k-1)$, the current state of the system $X(k)$ and the set point SP, while it returns the new optimal vector $U^o(k)$.
The function first sets the optimization problem, its objectives, its constraints and its parameters using the appropriate Tomlab syntax [25]. Then, it solves the problem using one of the various available Tomlab solvers [6] [3].

The solver works iteratively, evaluating at any iteration the value of the cost function J through the Matlab function *tom_fc.m*, corresponding to a different tentative value of $U(k)$. The result of the optimization ($U^o(k)$) is returned by *tom_main_J.m* to the upper level of the software and it is routed to a Simulink *demultiplexer* which extrapolates only the first vector $u(k)$ and routes it to the controlled system.
An instance of the main sections which compose the function *tom_main_J.m* is reported below.

```
function out=tom_main_J(Up,Xp,H,Hc,dT,nU,nX,nY,nlevel,SPgl,SP)

%problem setting
x_0 = Up';          % Starting values for the optimization
x_L = zeros(1,nU)'-25;          % Lower bounds for x
x_U = zeros(1,nU)'+25;          % Upper bounds for x
fLB = 0;          %Lower bound on function
f_opt=0;          % Optimal value for the objective function
c_L_L = -1*ones(1,H*nlevel);          % Lower bounds on levels
c_U_L = +1*ones(1,H*nlevel);          % Upper bounds on levels
c_L_dU = -10*ones(1,H*nu);          % Lower bounds on flow rates
c_U_dU = +10*ones(1,H*nu);          % Upper bounds on flow rates
c_L = [c_L_L, c_L_dU ]';          % Lower bound on constraints
c_U = [c_U_L, c_U_dU ]';          % Upper bound on constraints

Prob=simAssign('tom_fc',[],[],[],x_L,x_U,[],x_0,fLB,[],[],[],[],c_L,c_U,[],[],f_opt);

Prob.optParam.eps_f=10;          % Tolerance on objective function values
Prob.optParam.MaxIter=5;          % Max iterations
Prob.optParam.MaxFunc=10;          % Max function evaluations
Prob.user=[H Hc dT nU nX nY Xp SP nlevel SPgl]; % Prob user parameters

%problem solving
Result = tomRun('knitro', Prob, 3);          % The solver 'knitro' is chosen

out=Result.x_k;
```

Computation of the cost function

Each time the optimization solver requires to evaluate the objective function and to verify the constraints, the Matlab function *tom_fc.m* is called with

the current tentative value of $U(k)$. This function computes the selected objective function J (7.15) or (7.16) and gets all the variables required to calculate it.
Since the control input vector $U(k)$ and the set point $SP(k)$ are passed from the calling function *tom_main_J.m*, the only information to be generated is the vector of the system outputs $Y(k)$, whose power elements are necessary to calculate J, while its level elements constitute a subset of the constraints to be satisfied.
Then, it is clear the need to use the prediction model in the MPC algorithm, since the cost function (which expresses the control purposes) depends on the future system outputs, which need a model for prediction.
In *tom_fc.m* the prediction of the vector $Y(k)$ is generated by the Matlab command *sim()*, which simulates the prediction model (either linear or not) defined separately in a Simulink file (for example *hpv_prediction_model.mdl*).

This simulation must start from the current state of the controlled system and must end one prediction horizon later, therefore the whole state of the controlled system must be measurable, or at least observable. This is a real problem when the controller is applied to the real system, because each reach model introduces $2N + 1$ states, while in the real system only 3 variables are really measured (the final level and the two flow rates at the boundaries of the reach) and the implementation of a Kalman Filter to predict the unmeasured states [28] [45] [44] is indeed not trivial.
However, according to the purposes of this Thesis, the application of this controller to a real system will be done only in a later phase and, at the moment, it is reasonable to assume that the states of the controlled system are all measurable.

After the simulation, the objective function and the constraints can be computed and returned to the solver, which evaluates them and decides the new value of $U(k)$ for the next iteration.
The most significant parts of a possible implementation for *tom_fc.m* are presented below.

```
function [f c] = tom_fc(x, Prob)

% prediction model simulation
opt=simset('InitialState',Xp);
Ye=sim('hpv_prediction_model.mdl',H*dT,opt,[tsim,Usim]);

% system outputs division
Ye=resample_horiz(Ye,H);        % resampling of the Ye vectors in H samples
Y_L=Ye(:,1:nlevel);         % levels
Y_P=Ye(:,nlevel+1:ny);          % powers

% local set points division
SP=reshape(SP,ny,H)';         %reorder SP in a H*ny matrix
SP_L=SP(:,1:nlevel);          %levels SP
SP_P=SP(:,nlevel+1:ny);         %powers SP

% weights definition
maxL=10;
maxP=10e6;
Q_L=[1 1 1 1 1 1 1 1 1]/maxL^2;
Q_P=[1 1 1 1 1 1 1]/maxP^2;
Q=[Q_L Q_P];
R =[1 1 1 1 1 1 1 1 1];

% goal functions computation
fPref = sum((SP_P - Y_P).^2*Q_P');
fPtot = sum((ones(H,1)*SPgl - ones(npower,1)*(Y_P)).^2);
f = fPref;          % fPtot;          % choice of the objective function

% constraints computation
Yc=reshape(Y_L,nlevel*H,1);
dUsimc=reshape(dUsim,nu*H,1);
c=[Yc; dUsimc];         %constraints
```

Simulink implementation

In the previous paragraphs, the conceptual scheme of the developed controller (Fig.7.2) and the Matlab scripts which implement its subfunctions have been presented. Now, the Simulink structure implementing this conceptual scheme is shown.

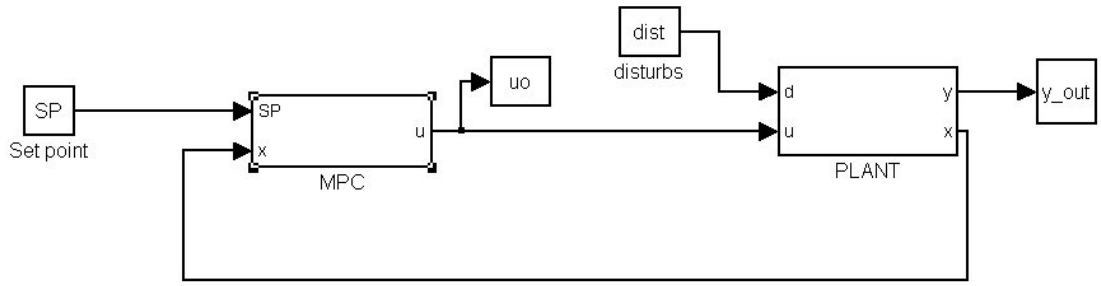

Fig. 7.4: Simulink control scheme.

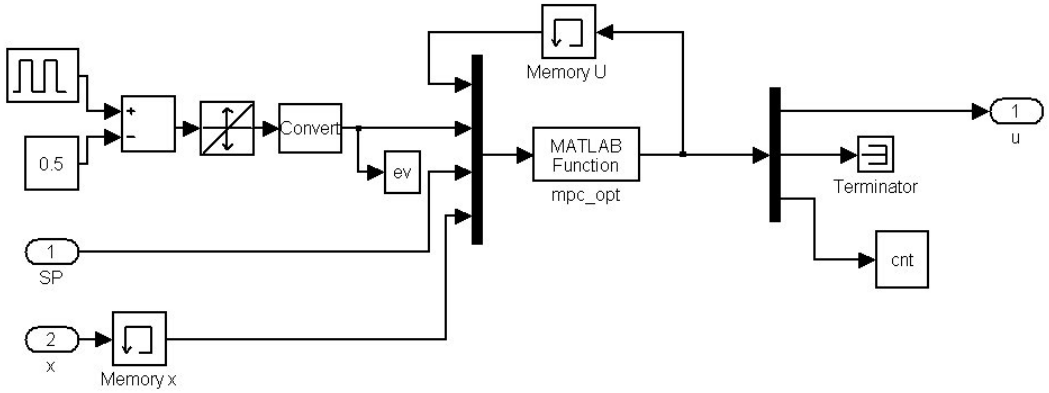

Fig. 7.5: Simulink MPC controller.

Fig.7.4 displays the general Simulink layout of the control scheme, where the subsystems of the MPC controller and of the plant with their interface variables are clearly visible. In particular, the subsystem which implements the MPC structure explained above is shown in Fig.7.5, where it is possible to notice all the correspondences with the conceptual scheme of Fig.7.2:

- the "MATLAB Function" block calls the function *mpc_opt.m*, which receives as inputs:
  - the previous control variable vector;
  - the event ev;
  - the set point SP;
  - the state of the plant;

- the generator of the event ev, as shown in Fig.7.3;
- the "Memory U" block, which makes the control variable vector of the previous simulation step time available;
- the demultiplexer, which selects the first vector $u(k)$ from the larger one $U(k)$ and routes it towards the plant;
- the "Memory x" block, which only has the purpose to break the algebraic loop and to overcome numeric problems during the simulation.

The developed MPC controller has been entirely described, but no mention has already been made to the choice of the optimization solver among those provided by Tomlab. This choice is a very important step in order guarantee satisfactory control performances and deserves a separate explanation.

7.2.3 Optimization Tool

Why Tomlab

Since the optimization problem underlying the MPC controller is indeed not trivial and involves a lot of variables, it is advisable to look for a good solver, testing different solutions and choosing between them. Moreover, since the solver will be employed in an on-line optimization, it must be provided with a good interface with the Matlab environment, as it has to be automatically called during the simulation, and, later, during the real-time control. Tomlab [25] [6] [23] seems to satisfy these requests.

Tomlab is a general purpose environment in Matlab for the numerical solution of many different optimization problems, which provides robust and reliable tools to be used in the development of algorithms and software.
There are many good solvers available in the area of numerical analysis, operational research and optimization, but because of the different languages and systems, as well as the lack of standardization, it is not indeed simple to use them. As a matter of fact, if it is necessary to test different solvers, it is often required to rewrite the problem formulation and the function specifications, or to develop some new interface routines.

The strength of Tomlab is that it allows to define the optimization problem once and then to run the many available solvers, because it takes care of all the interface problems, whether between languages or due to different problem specifications. Furthermore, it is entirely developed in the Matlab

environment. Thus, it employs the concept of structure arrays and the ability to execute Matlab code defined as string expressions. This makes Tomlab very powerful and, most important, completely compatible with the software where our controller has been defined, so that no interfaces are necessary.

For these reasons, Tomlab has been chosen as the platform for the solution of the optimization problem associated to the MPC controller.

Optimization problem and available solvers

In order to choose the tool which best solves our optimization problem, it is important to figure out which problem we are considering.
As a matter of fact, each solver is designed for a particular problem and, even if it might work in other cases, it gives its best performances when properly applied.
Then, the first step to choose a solver is to find out, among the available solutions, those matching with the problem considered.
In the field of operations research, many families of problems have been defined, such as:

- linear or nonlinear;
- quadratic or not quadratic;
- constrained or unconstrained;
- local or global;
- integer or continuous;
- ...

The combination of these characteristics defines the typology of the problem: LP, MILP, QP, MIQP, MIQQ, NLP, LPCON, QPCON, LLS, MILLS, NLLS, GLB, GLC, SDP, BMI, ...
For each one of these families Tomlab provides:

- one function which mathematically sets the problem (ex. conAssign);
- one suitable solver for the specific case (ex. conSolve).

The problem setting is performed by the specific function, which creates one standard structure named *Prob* that contains all the information on the problem to be solved. An example of the Tomlab syntax is:

Prob = conAssign('tom_f','tom_c', parameters)

conAssign is the Tomlab function which creates a proper *Prob* structure in which the problem to be solved is defined as a generic constrained nonlinear problem. It receives as input the problem parameters and the user defined functions *tom_f* and *tom_c*, which compute the cost function and the constraints.

The problem solving is always performed by the Tomlab function *tomRun*, which receives as inputs the selected solver and the previously created problem structure *Prob*. For example, using the solver *conSolve* the syntax is:

Result = tomRun('conSolve', Prob)

In this way, once the problem has been defined, it is possible to test different solvers simply by changing the name of the solver in the function *tomRun*, without modifying the problem definition.

In our case, the optimization problem introduced by the MPC algorithm does not belong to the set of standard cases because the cost function J can not be directly expressed in terms of a static equation, while it requires the simulation of a dynamic system.
This implies that most of the available tools can not be used, since they require a defined structure for the cost function. Rather, one can use only those which allow the user to freely define the minimization criterion.
With the help of the staff of the company which releases the Tomlab software, a few optimization solvers have been selected: *glcSolve, glcDirect, glcFast, glcCluster, oqnlp, lgo, multiMIN, snopt, npsol*. In particular, it has been suggested to test *snopt* and *npsol*, which perform a local optimization, after using one between *glcSolve* and *glcDirect*, which are global solvers, in order to refine the solution.
Therefore, a simple test to evaluate their performance in a simple situation has been run.

Solver testing

In order to test the behavior of the selected solvers and to finally take a decision, they have been tested in the design of the MPC controller with a particular control objective and in a defined scenario. To make the procedure

easier and faster, the linear model has been employed for both the controlled plant and the prediction model.
In the adopted scenario, the controller must maintain the initial value of the all the levels, when a step disturbance of $40m^3/s$ acts on the inlet flow rate of the first lake. Thus, the cost function is:

$$J_{test} = \sum_{i=0}^{H} (y_L(k+i) - SP_L(k+i))^T (y_L(k+i) - SP_L(k+i)) \qquad (7.18)$$

where:

- y_L is the column vector of the levels;
- SP_L is the column vector of the level set points, which coincides with their initial state.

Actually, the optimal solution is obvious to understand and consists in immediately giving to all the downstream flow rates a step equal to the disturbance. However, it is not easy to find numerically this solution, as it involves a lot of interacting variables. For this reason, it is a good test for the optimization solver.

Initially, in order to have a benchmark to be compared with the test results, an open-loop simulation of the uncontrolled system in the previously described scenario has been done. The simulation time has been chosen sufficiently long with respect to the involved dynamics, in order to show what happens also after a long period (8 hours).
All the results are reported in the following as variations of the considered variables with respect to their steady state.
Fig.7.6 reports the expected result: when, at time $3000s$ the step disturbance occurs, only the level $L1$ increases, due to the pure integrator describing the dynamics of the lake.

Subsequently, the controller has been inserted and all the selected solvers have been used.
The results achieved by each solver are listed below:

1) *glcSolve*: the simulation is completed in 1 minute, the results are reported in Fig.7.7;

2) *glcDirect*: the simulation is completed in 1 minute 45", no difference is noticed from the results obtained with *glcSolve*;

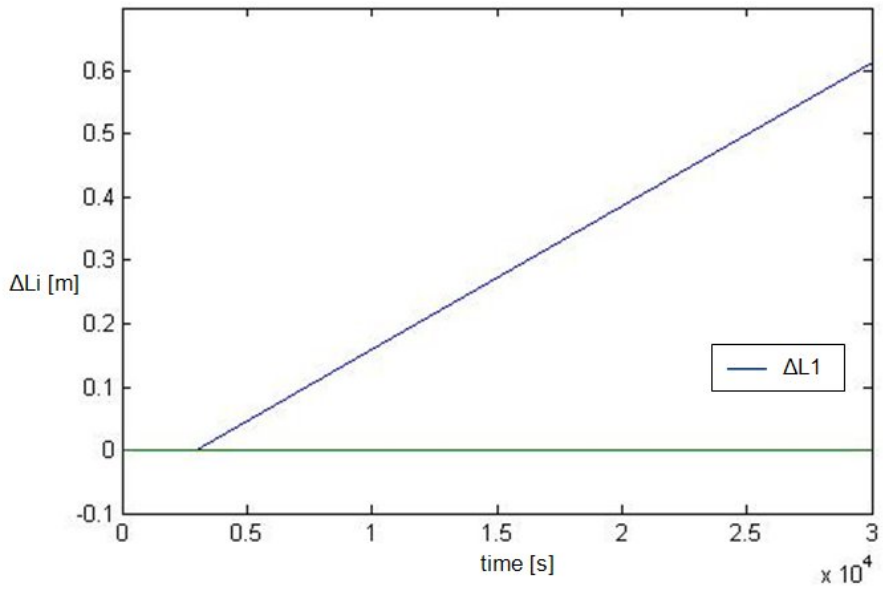

Fig. 7.6: Open loop simulation.

3) *glcFast*: the simulation is completed in 1 minute 30", no difference is noticed from the results obtained with *glcSolve*;

4) *glcCluster*: the simulation stops (crashes) in the initial instants;

5) *oqnlp*: the simulation is completed in about 15 minutes, the results are reported in Fig.7.8;

6) *lgo*: the simulation stops (crashes) in the initial instants;

7) *multiMIN*: the simulation stops (crashes) in the initial instants;

8) *glcSolve + snopt*: the simulation stops (crashes) in correspondence to the instant where the disturbance occurs;

9) *glcSolve + npsol*: the simulation stops (crashes) in correspondence to the instant where the disturbance occurs.

It can be noticed that none of the solvers achieves the expected optimal result and some of them are clearly inefficient, as they cannot bear the computational burden or their structure is not compatible with our problem. However, four solvers achieve a solution and three of them reach exactly the same result, even if they require different computational times. These solutions (Fig.7.7 and Fig.7.8), even if not optimal, maintain all the levels

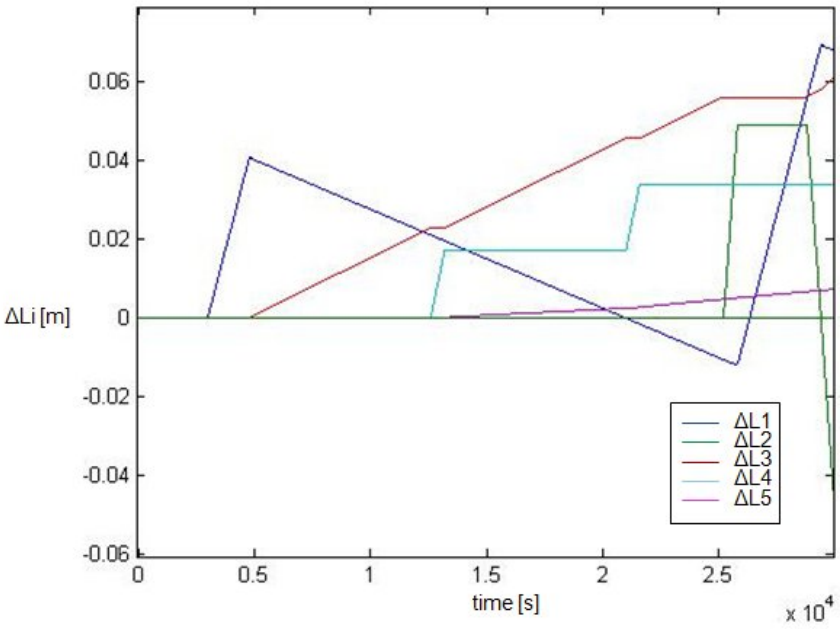

Fig. 7.7: Test with glcSolve.

under a satisfactory value, which is one tenth of the value reached in the open-loop simulation (Fig.7.6). In particular, the responses obtained with *glcSolve*, *glcDirect* and *glcFast* (Fig.7.7) seem to maintain the level better limited, while that obtained with *oqnlp* (Fig.7.8) seems to diverge.
Therefore, initially the solver *glcSolve* has been chosen for the development of the MPC controller.

Once the solver has been selected, it is possible to choose the Tomlab function for the problem setting, which generates the structure *Prob*, among those compatible with that solver.
The most recommended function [6] for *glcSolve* is *glcAssign*. However, another interesting function compatible with *glcSolve* is *simAssign*, which, unlike *glcAssign*, allows to define only one function which computes both the cost function and the constraints (as *tom_fc.m*). This, in our case, is a great advantage because it allows to simulate the prediction model only once (and not twice) at each optimization step of the solver.
Therefore, *simAssign* is chosen as the problem setting function.

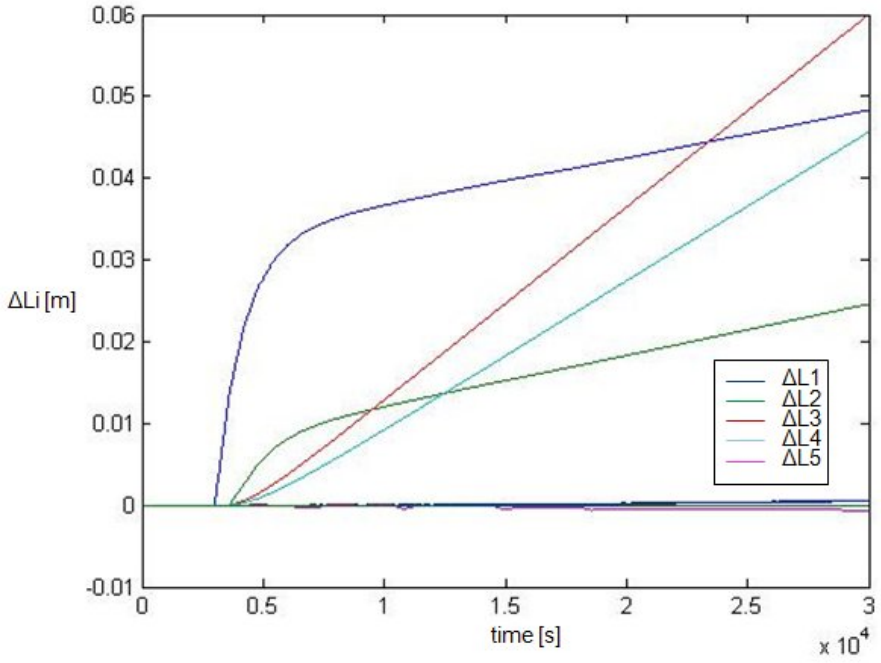

Fig. 7.8: Test with oqnlp.

Advanced solver testing

After choosing the solver, which has been tested in a simple situation (see Section 7.2.3), it can be finally used in a likely working condition, that is when the MPC controller is asked to follow a power set point while respecting some constraints on levels and flow rates.
In this way, it is possible to evaluate the performances of the whole MPC controller with respect to its parameters.

The considered working condition is defined as follows:

- the initial state is equal to the reference steady state defined in Chapter 6;
- the control objective is expressed by a set of local power set points, therefore J is defined by (7.15);
- all the local set points are equal to the steady state while only one, which corresponds to Pe_5, differs from its steady state value by $+50MW$;
- the bounds on all the levels differ from the steady state by $\pm 1m$;
- the bounds on all the flow rates differ from the steady state by $\pm 25m^3/s$;

- no bounds are imposed on the flow rate variations;
- the discrete sampling time dT of the controller has been set to $600s$;
- the prediction horizon H has been set equal to 15 sampling times;
- the whole simulation time is $5 \cdot 10^4 s$, about $14h$.

The results obtained are shown in Figures 7.9, 7.10 and 7.11, which display the controlled powers, the levels to be bounded and the control variables (flow rates). All the variables are plotted as variations with respect to the steady state.

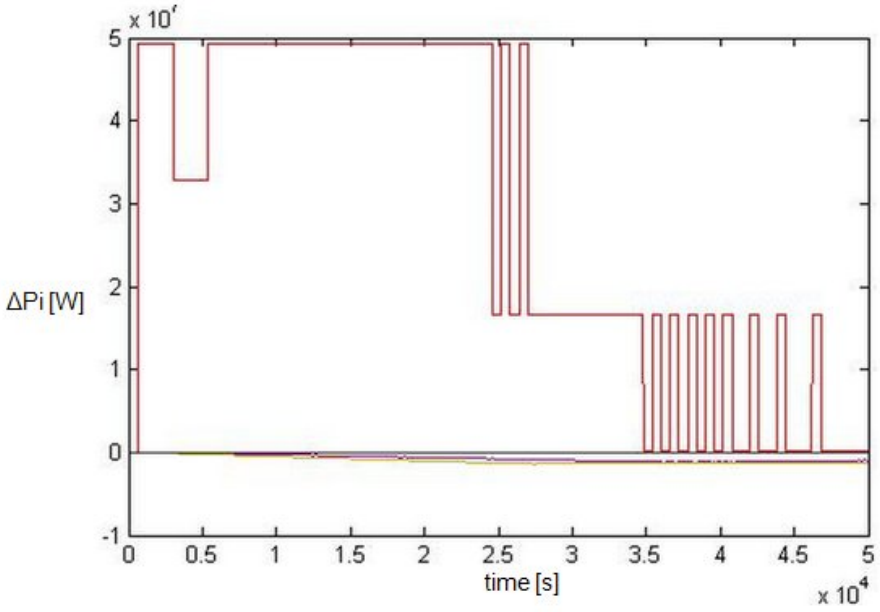

Fig. 7.9: Powers with glcSolve.

It can be noticed from Fig.7.9 and Fig.7.10 that the power Pe_5 quickly reaches its set point, with a considerable undershoot, and maintains the correct value with a small static error till the levels are not close to their bounds. Then, to avoid the constraints violation, the set point tracking is sacrificed and the power Pe_5 seems to return to the initial value. Fig.7.11, instead, shows that the flow rates stay always far away from their bounds.

The transient of the flow rates shows an unexpected behavior of the controlled variables, which seem to change only by defined discrete values. The

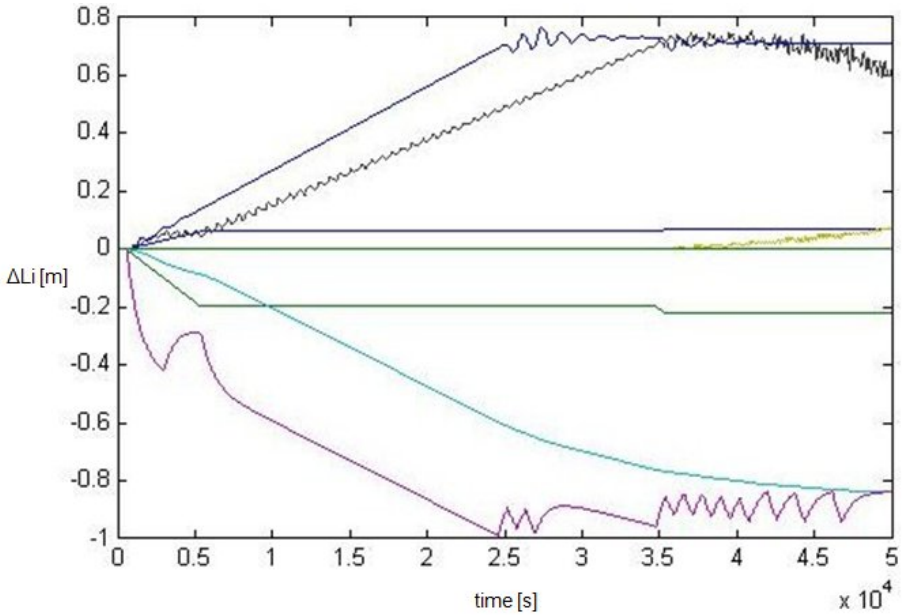

Fig. 7.10: Levels with glcSolve.

same behavior, due to the existing direct relationship (see Section 6.2.1), is also observable in the powers transients.
In order to better understand this result, several simulations with different parametrizations have been run plotting in real-time the flow rates computed during each optimization. The final observation is that *glcSolve* tries to find the optimal point by incrementing the vector to be optimized always by a constant value, whose amplitude depends on that of the set point.

This may explain the static error: if only discrete values are available for the flow rates, and then for the powers, the solution can guarantee only that the error is smaller than the discrete variation of the powers.
However, this fact is indeed bizarre and no satisfactory explanation has been found in the Tomlab documentation [3] [25] [6] [23]. The only reasonable, but incomplete, explanation is that *glcSolve*, a solver built to manage global optimization problems, uses large discrete variations in order to arrive quickly close to the solution, but in this case it is no longer able to reduce these variations for a fine tuning of the optimization process.

For these reasons, a local solver is tested instead of the global *glcSolve*. Looking at the Tomlab documentation [25] [6] [24], the solver *knitro* has been chosen, because it is recommended in association to the used problem

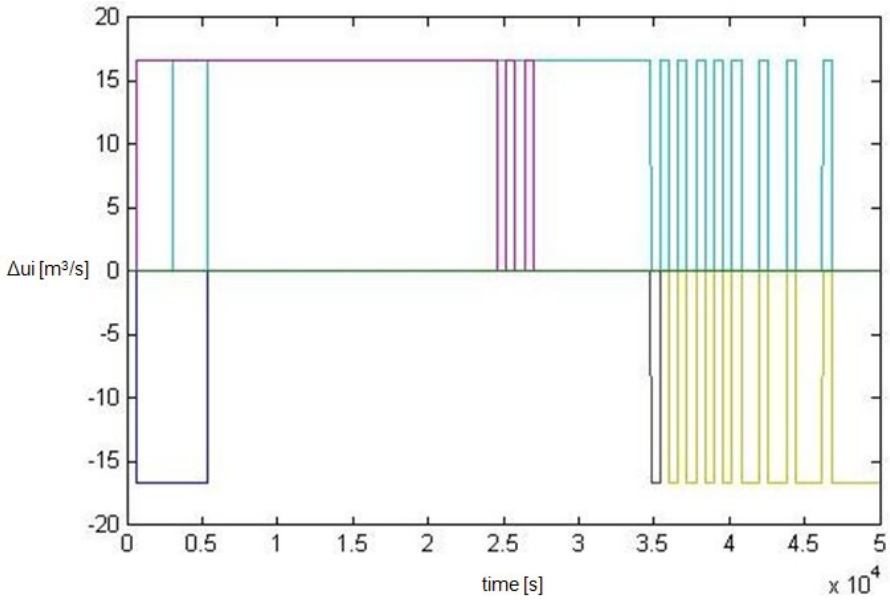

Fig. 7.11: Flow rates with glcSolve.

setting function *simAssign*. The new solver has been tested within the MPC controller and with the same scenario and parametrization presented above. The initial results show that the problem detected with the previous solver is disappeared, but the overall performances are much worse. However, it is sufficient to reduce the prediction horizon H from 15 to 5, in order to have less variables to be optimized, to achieve some very interesting results, which are reported in Figures 7.12, 7.13 and 7.14.

Looking at these figures, it is clear that in the first part of the simulation the results are much better than in the previous case:

- Pe_5 quickly reaches its set point;
- there are neither overshoots nor undershoots at the beginning of the transient;
- the static error quickly goes to zero;
- all the variables vary with continuity.

On the other hand, the constraints are not always fulfilled, as Fig.7.13 shows that three levels go beyond their bounds. This behavior can be explained looking at Fig.7.14, where it is possible to notice that the flow rates arrive

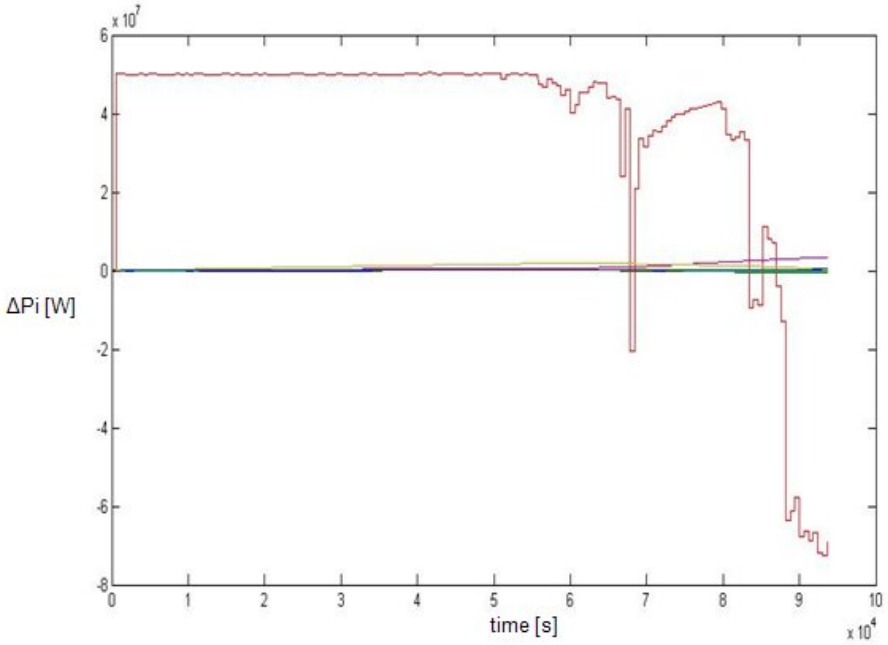

Fig. 7.12: Powers with knitro.

close to their bounds and, as they constitute the variables respect to which the objective function is minimized, their constraint violation avoidance takes a higher priority in the optimization routine and penalizes both the trajectory tracking and the other constraints satisfaction (which are necessarily soft constraints).
However, the overall performances of the controller have been judged satisfactory, also because the real bounds on the lakes levels are usually larger than $1m$, about $5m$, and this guarantees a perfect tracking of the set point for 5 times the actual period, that is about 3 days and half, much more than the usual period of variation of the required power.
Therefore, the final decision has been to use *knitro* for the implementation of the MPC controller.

7.2.4 Integral action

In the design of a controller, it is common practice to include also an integral action in order to guarantee that the static error vanishes in presence of constant references or additive disturbances [39].
In this section, the classical approach used to add the integral action to a MPC controller is presented. Then, an alternative solution is proposed and

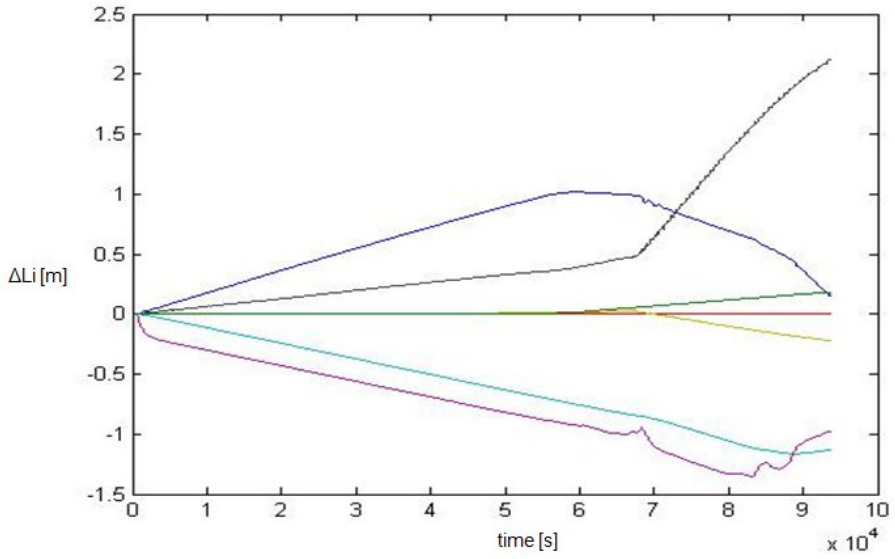

Fig. 7.13: Levels with knitro.

discussed.

Integral action in MPC

The most common way to include an integral action in a MPC controller consists in inserting one integrator on each control variable, as in Fig.7.15, so that the vector $u(k)$ is obtained as the output of the dynamic system described in equations (7.19) and (7.20), where $v(k)$ represents the state vector of the integrators.

$$u(k) = u(k-1) + \delta u(k) \tag{7.19}$$

$$\begin{cases} v(k+1) &= v(k) + \delta u(k) \\ u(k) &= v(k) + \delta u(k) \end{cases} \tag{7.20}$$

Then, it suffices to include in the cost function to be minimized the vectors $\delta u(k+i)$ in place of the vectors $u(k+i)$. In this way, the cost functions presented in Section 7.2.1 have to be modified as follows.

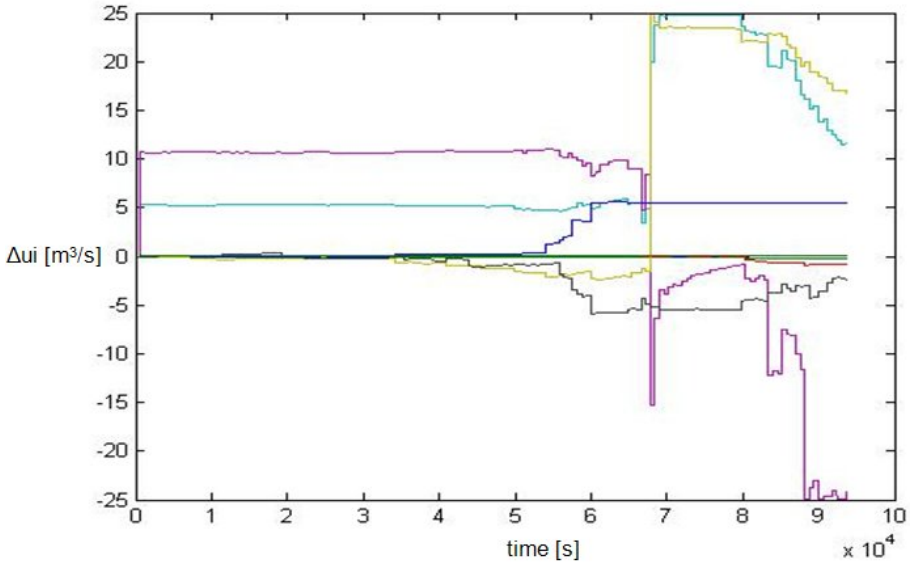

Fig. 7.14: Flow rates with knitro.

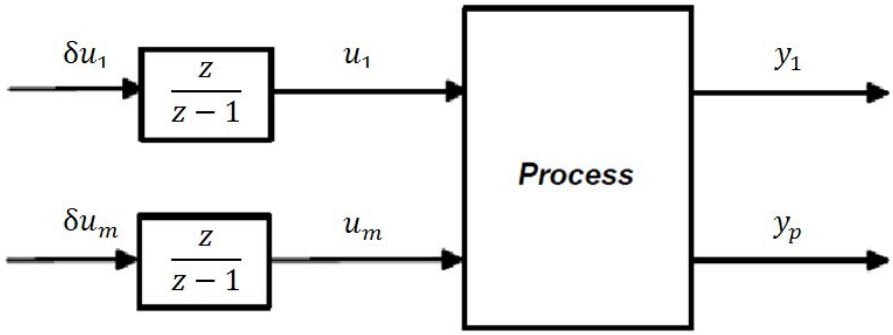

Fig. 7.15: Scheme of the classical integral action for MPC.

The general trajectory tracking form presented in (7.14), becomes:

$$J = \sum_{i=0}^{H-1} (\|y(k+i) - SP(k+i)\|_Q^2 + \|\delta u(k+i)\|_R^2) + \|x(k+H)\|_S^2 \quad (7.21)$$

Then, the two objective functions chosen for our control purposes and presented in equations (7.15) and (7.16) must be redefined as in (7.22) (local set points) and (7.23) (global set point).

$$J = \sum_{i=0}^{H}((y_P(k+i)-SP_{loc}(k+i))^T Q_P(y_P(k+i)-SP_{loc}(k+i))+\delta u(k+i)^T R\delta u(k+i)) \tag{7.22}$$

$$J = \sum_{i=0}^{H}((\sum_{j=1}^{npower} y_P(k+i,j)-SP_{glob}(k+i))^2 q_P+\delta u(k+i)^T R\delta u(k+i)) \tag{7.23}$$

The Simulink implementation of this strategy is easy and can be performed by substituting the vector $\delta U(k)$ to the vector $U(k)$ in the objective function computation, within the Matlab function *tom_fc.m*.

The construction of the vector $\delta U(k)$ can follow two equivalent methods:

1) the vector passed by the solver to the function *tom_fc.m* at each optimization step is still $U(k)$, then $\delta U(k)$ is obtained by the discrete derivation $\delta U(k) = U(k) - U(k-1)$;

2) the vector passed by the solver to the function *tom_fc.m* at each optimization step is directly $\delta U(k)$; then, at the end of each optimization, the solver returns to the upper level of the MPC algorithm the optimal vectors $\delta U^o(k)$, which have to be integrated by the discrete integration $U^o(k) = U(k-1) + \delta U^o(k)$ to obtain the control vector acting on the system.

An example of implementation of the first method is shown below:

```
function [f c] = tom_fc(x, Prob)
% x: current guess for the optimal input Uopt. Size(x)=[1,nU]:

% Usim and dUsim matrices construction
Usim=zeros(lt,nu);        % simulation inputs matrix
for i=1:Hc
for j=1:nt
Usim((i-1)*nt+j,:)=x((i-1)*nu+1:i*nu)';
end
end

dUsim=zeros(lt,nu);       % simulation inputs increment matrix
for i=1:lt-1
dUsim(i,:)=Usim(i+1,:)-Usim(i,:);
end

% prediction model simulation
opt=simset('InitialState',Xp);
Y=sim('hpv_prediction_model',H*dT,opt,[tsim,Usim]);

% objective function calculation
fPref = sum((SP_P - Y_P).^2*Q_P') + sum(dUsim.^2*Rd');
f = fPref;
```

The described strategy has been used to include an integral action in our controller and several simulations have been performed. However, the integral action does not affect the control performances in terms of steady state error and the obtained results are pretty equal to those reported in Figures 7.12, 7.13 and 7.14. The reason can be found in a simple theoretical analysis [22] [9], which demonstrates the existence of a tricky problem when using the described classical strategy associated to a state feedback control law.
Actually, a proof is easy to develop only for linear unconstrained problems, but it is representative of what happens also in a more general case. Looking at the overall control scheme represented in Fig.7.16, it is clear that the control variable u is computed on the basis of the contributes of both the reference Y^o and the enlarged state vector $[x\ v]^T$. More specifically, there is a feedback of the integrator state v, through the regulator block K_v, which

makes the integral action disappear in practice.
The theoretical solution is to include in the control loop a state observer which estimates both the process state x and the state of the integrator v. In this way, as shown in Fig.7.17, the integrator is not canceled by the internal feedback.

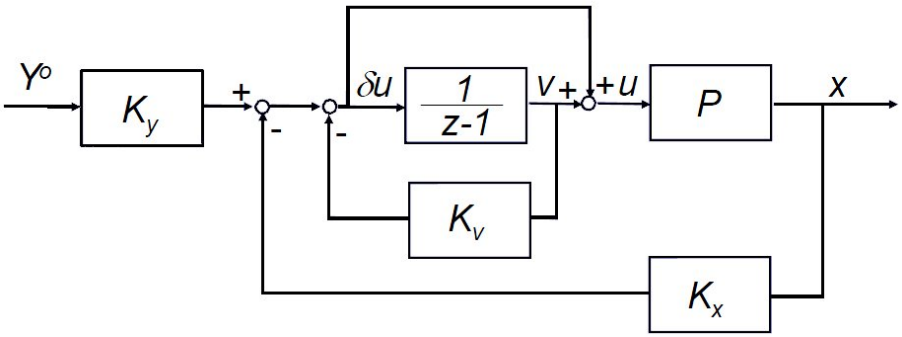

Fig. 7.16: Control with state feedback.

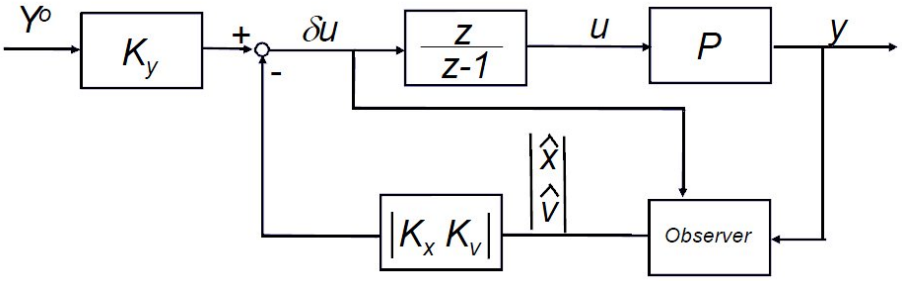

Fig. 7.17: Control with estimate state feedback.

Unfortunately, as stated in Section 7.2.2, we assumed that all the state variables are available, so that no observers have been placed into the control loop. Therefore, a different method to provide an integral action has been used, that is the inclusion of an integral action directly acting on the error variables, as it is common practice in the design of regulators for MIMO systems.

Integral action for MIMO systems

The classical approach to design a controller with integral action for a MIMO system consists of the following steps:

1) the reference errors are created by subtracting the controlled variables to their set points;

2) each error is passed through an integrator;

3) the remaining part of the MIMO controller is synthesized for the dynamic system composed by both the controlled process and the set of the integrators.

Fig.7.18 shows the structure of this solution, where every signal i is expressed by its Laplace transform [39] [10]. Specifically, $E_i(s)$ is the reference error, $V_i(s)$ is the integrator state and $U_i(s)$ is the control variable, while $R'(s)$ is the stabilizing part of the controller and $G(s)$ is the transfer function of the controlled system.

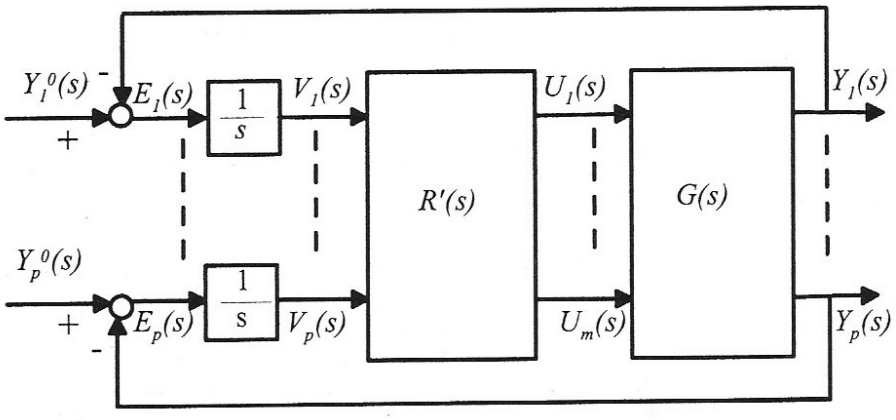

Fig. 7.18: Control scheme with integral action for MIMO systems.

The conditions which have to be satisfied in order to use this control scheme, both verified in our case, are:

- the number of control inputs must be greater or equal to the number of controlled variables;
- the controlled system must not have invariant zeros in $s = 0$, that is pure derivative actions.

The main difference of this approach with respect to the previous one is that in this case the integrators do not act on the control variables generated by the controller, but directly on the error signals, avoiding the problems caused by the use of a state feedback.

Then, it is worth testing this method in our problem, with an equivalent formulation suitable for a MPC control law. The algorithm which has been developed is discussed below and the corresponding Simulink diagram is presented in Figures 7.19, 7.20 and 7.21.

1) within the controller block (Fig.7.20), the error e is computed;

2) the error is integrated generating the signal v, which is the state of the integrator;

3) the integrator state vector v is passed, together with the measured state of the controlled system x, to the prediction model, in order to initialize its states; the prediction model now contains also the integrators to be initialized, because it is necessary to simulate the behavior of the whole system including the integrators;

4) when called by the function *tom_fc.m*, the prediction model (Fig.7.21), generates the prediction of the system outputs;

5) the predicted outputs *ypr* are subtracted to the set points SP, generating the vector of the predicted error *epr*;

6) the predicted error *epr* is integrated by the integrators of the prediction model starting from the previously initialized state vector v, generating the signal *vpr*;

7) the signal *vpr* is returned together with the predicted output vector *ypr* to the function *tom_fc.m*, which includes both in the objective function to be minimized:

```
function [f c] = tom_fc(x, Prob)

...

SP_L=SP(:,1:nlevel);          %level set points
SP_P=SP(:,nlevel+1:ny);       %power set points

% prediction model simulation
opt=simset('InitialState',Xp);
Ye=sim('hpv_prediction_model',H*dT,opt,[tsim,Usim]);

Y_L=Ye(:,1:nlevel);           % levels
Y_P=Ye(:,nlevel+1:ny);        % powers
v_L=Ye(:,ny+1:ny+nlevel);     % integrated level errors
v_P=Ye(:,ny+nlevel+1:2*ny);   % integrated power errors

% objective function calculation
fPi = sum((v_P).^2*Q_Pi');    % integral action contribute
fPref = sum((SP_P - Y_P).^2*Q_P');   % trajectory tracking contribute
f = fPref + fPi;              % total objective function
```

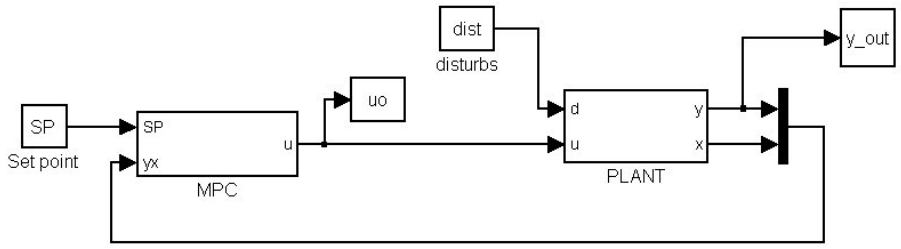

Fig. 7.19: Control system with integrators.

In this way, the implemented cost functions are presented in (7.24) (local set points) and (7.25) (global set point).

$$J = \sum_{i=0}^{H}((y_P(k+i)-SP_{loc}(k+i))^T Q_P (y_P(k+i)-SP_{loc}(k+i))+v(k+i)^T Q_{Pi} v(k+i)) \tag{7.24}$$

$$J = \sum_{i=0}^{H}((\sum_{j=1}^{npower} y_P(k+i,j) - SP_{glob}(k+i))^2 q_P + v(k+i)^T Q_{Pi} v(k+i)) \tag{7.25}$$

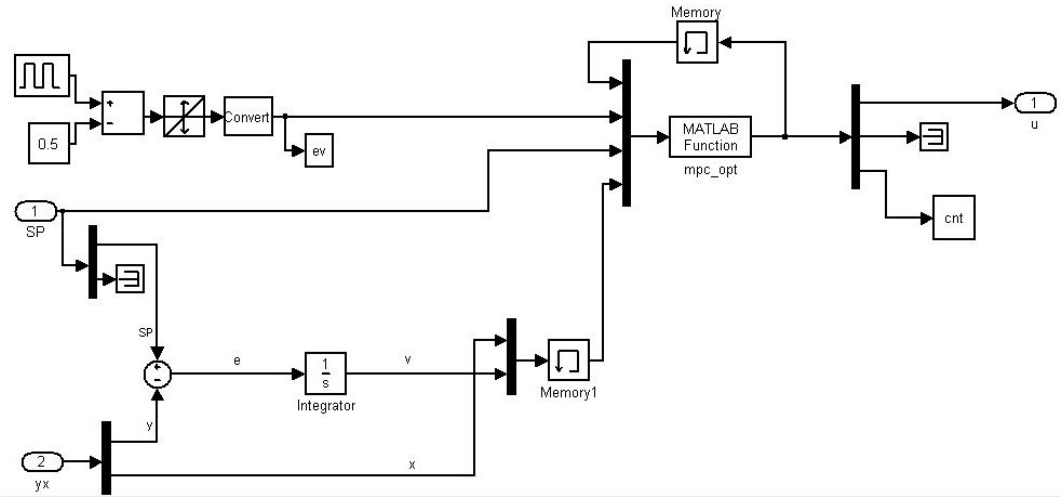

Fig. 7.20: MPC controller with integral action.

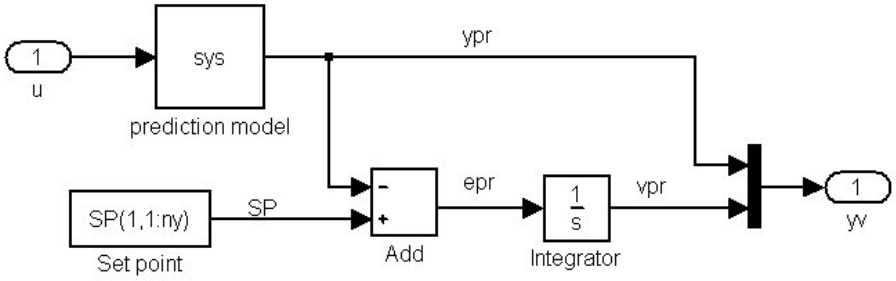

Fig. 7.21: Prediction model with integral action.

where:

- $v(k+i)$ is the column vector of the predicted integrated errors of time $k+i$;
- Q_{Pi} is the diagonal matrix of the weights of the integrated errors.

Therefore, not only the error $(Y(k)-SP(k))$ is minimized over the prediction horizon, but also its integral, reproducing the effect of the classical integral action and leading to an asymptotically null static error.

However, simulating the resulting control system with integral action, it is possible to observe that the obtained results are much worse than in the case without integrators. This may be due to the added complexity caused by the inclusion of the integrators. Therefore, in order to reduce the number

of variables to be optimized and to lighten the computational burden, the discrete sampling time dT of the controller has been changed from $600s$ to $2700s$, which is however a reasonable value in view of the dynamics of the controlled system.
In this case, the results presented in Figures 7.22, 7.23 and 7.24 are very satisfactory, since a significant contribution of the integral action is evident in the last part of the simulation, where the controlled power Pe_5 stays near the set point, instead to completely loose its reference when the levels are close to their bounds (like in Fig.7.12). At the same time, the levels excursion is not compromised, even if their trend is not as regular as in Fig.7.13, which represents the corresponding case without integrators.

Therefore, a pretty good solution in order to provide an integral action to a MPC control system without state observer has been found. It is important to underline anyway that the integral effect on the overall control performances is very sensitive to the integral weight Q_{Pi} (see (7.24) and (7.25)), which is an additional parameter of the MPC controller and has to be chosen carefully.

7.3 MPC Control of the Hydro Power Valley

Once the controller has been completely developed, it is interesting to evaluate its performances when it is applied in its expected working conditions. Thus, this section illustrates the results achieved by using the presented MPC controller in several situations.
As a good nonlinear model of the considered hydro power valley is available, it is natural to suppose that, in a further real implementation, it may play the role of the prediction model, while the controller acts on the real system. However, the high computational time required to simulate it with the available resources makes it very hard to use this model for prediction, as it has to be called and simulated at each iteration of the optimization algorithm.
Then, in view of the purposes of this work, it has been decided to use a linear prediction model within the controller, which makes the optimizations much faster, and to use the nonlinear one as the controlled system. This configuration makes also possible to highlight the loss of performance due to the non-exact correspondence between prediction model and controlled system, which is what happens in a real implementation. To this regard, in order to have a benchmark where the prediction model perfectly fits the system

dynamics, also the case in which both the predictor and the system are described by the same linear model has been considered.
Besides, according to what inferred in Section 7.2.1, in each case, both the prescribed local (7.15) and global (7.16) goal functions have been used and the results achieved in the two cases have been commented.
In the simulations, a variation of $50MW$ is initially required to only one local power (Pe_5), then the same amount is required to all the valley, leaving the controller decide the optimal distribution of the production. In this second case, in order to replicate a likely working condition, a further test with a variable global set point has been performed.

7.3.1 MPC of the linear system

When the linear model is used for both prediction and system simulation, the solver takes advantage from this correspondence because its prediction is exactly what the system is going to do in the near future. Then, any problem in the computation of the control action depends only on the difficulty of the optimization problem.
In the following, all the results reported are expressed as differences from the steady states.

Local Set Point

In this case, all the powers of the system are provided by a local set point, which has to be followed while fulfilling the constraints on levels and flow rates.
The best results, as already shown in Figures 7.22, 7.23 and 7.24, have been obtained by including the integral action in the control law.

Global Constant Set Point

In this case, only one global constant set point is defined, which represents the variation of the total power required to the whole valley.
The associated optimization problem is more critical than in the previous situation, because it theoretically has infinite optimal solutions. This may lead to unexpected variations of the controlled variables between different optimal solutions at each sample time of the controller.
Thus, in order to stabilize the final behavior, the flow rates variations have been bounded between $\pm 10 m^3/s^2$ so that, once one optimal solution is found,

it is not advantageous to change it any more.

From Figures 7.25, 7.26 and 7.27 it is possible to notice that the solver decides to act only on the most significant power (Pe_5) and, when the levels come close to their bounds, it distributes the power request to the other power plants, managing to satisfy the constraints.
The overall result is quite good, as the total power production (see Fig.7.26) seems to maintain the required set point, even if with some oscillations when the level constraints start to influence the optimization.

7.3.2 MPC of the nonlinear system

When the prediction model, in order to save computational time, is simpler than that describing the controlled system, the optimization procedure is affected by model errors that are not taken into account by the solver and may degrade the controller performances. Therefore, it is important to understand which variables are affected by a significative prediction error and, if necessary, to compensate this uncertainty with an appropriate choice of the weighting matrix within the cost function.
Moreover, due to the possible presence of constant additional errors on the predicted outputs, which do not depend on the real deviation of the controlled variables from their references, the integral action may result counterproductive and it has to be used carefully.

Simulating under these conditions, it comes evident the loss of performances with respect to the control of the linear system. However, some good results can be achieved by simply extending the level bounds to $\pm 2m$ from the steady states, which is anyway a reasonable value, as stated in Section 7.2.3. In order to make the results comparable with those obtained with the linear model, they have been all depolarized and they are reported as differences from the steady states.

Local Set Point

Figures 7.28 and 7.29 show the results obtained by requiring the nonlinear system to follow a set of local constant references, as described above.
It is possible to notice how the maintenance of the constant references is worse than that observed in Fig.7.22, as the controlled variables loose their set point many times, even in the first part of the simulation, when the level

constraints are still satisfied.
The low penalization of the worse predicted variable (like $L9$) makes it possible to guarantee a quite good behavior for the other variables, but it necessarily degrades the control performances over the interested ones. As a matter of fact, looking at Fig.7.29, the controller does not seem to guarantee the respect of the constraint on $L9$.

Global Constant Set Point

Analogous considerations can be made when a global constant power reference is imposed. As a matter of fact, comparing Fig.7.30 with Fig.7.26 it is evident that in the nonlinear case (Fig.7.30) the constant set point is maintained with more difficulty.

Global Variable Set Point

In the previous sections, it has been presented how the MPC controller behaves when operating with global or local constant set points.
Nevertheless, in the real application in which this controller is expected to operate, it receives daily a global power reference from the transmission system operator. Such a reference is not expected to be constant at all, as the request of electric power changes a lot during the day, being greater during the working hours and lower at night.

Therefore, to conclude this work of Thesis, it is important to test the developed controller in its natural working conditions, namely while operating on the nonlinear system with a global sine-shaped power set point.
In view of these reasons, two tests have been performed, each one with a different kind of reference:

- a square wave with a period of one day ($86400s$), simulating for one day;
- a sine wave with a period of one day, simulating for two days.

The first one, even if not realistic, makes it possible to evaluate the controller behavior when a sudden variation of set point occurs. Furthermore, the power reference is often expressed as discontinue function composed by little gradual steps, then, if the controller manages to follow one big step, it is reasonable to suppose that it will have no problems to handle a series of smaller ones.

In this situation, the optimization problem may become more difficult and, in order to simplify it, it is worth to reduce the total number of variables to be optimized, without changing the total prediction horizon. This has been performed by incrementing the controller sample time dT from $2700s$ to $7200s$ and reducing the prediction horizon H (which influences the number of optimization variables) expressed in multiples of dT from 5 to 2, so that the total time horizon $H_t[s] = dT \cdot H = 14400s$ is almost the same.

Figures 7.31 and 7.32 show how the local powers contribute to follow the global reference and it is possible to assert that there is a good trajectory tracking. In particular, when the reference changes abruptly, the sum of powers immediately adapts to the new required value.

The second kind of set point reproduces the most common function used to describe the overall electric power request submitted to the transmission system operator by the distributed networked loads, namely a sinusoidal wave. In order to verify the behavior of the control system on a longer period, the simulation have been prolonged to about two days ($2 \cdot 10^5 s$) and the results achieved can be defined very satisfactory.

As a matter of fact, Fig.7.34 shows that, after an initial settlement, the sinusoidal reference is tracked by the global power in an excellent way. Moreover, looking at Fig.7.35, it is interesting to underline that a periodic motion of the power reference allows the level constraints be more easily respected than in the constant case, because it involves a charge-discharge phenomenon on the reservoirs, which corresponds to the normal working condition of a hydro power valley. This situation also affects the quality of the set point tracking, because the solver does not have to overcome frequent constraint violations and can concentrate on the minimization of the objective function.

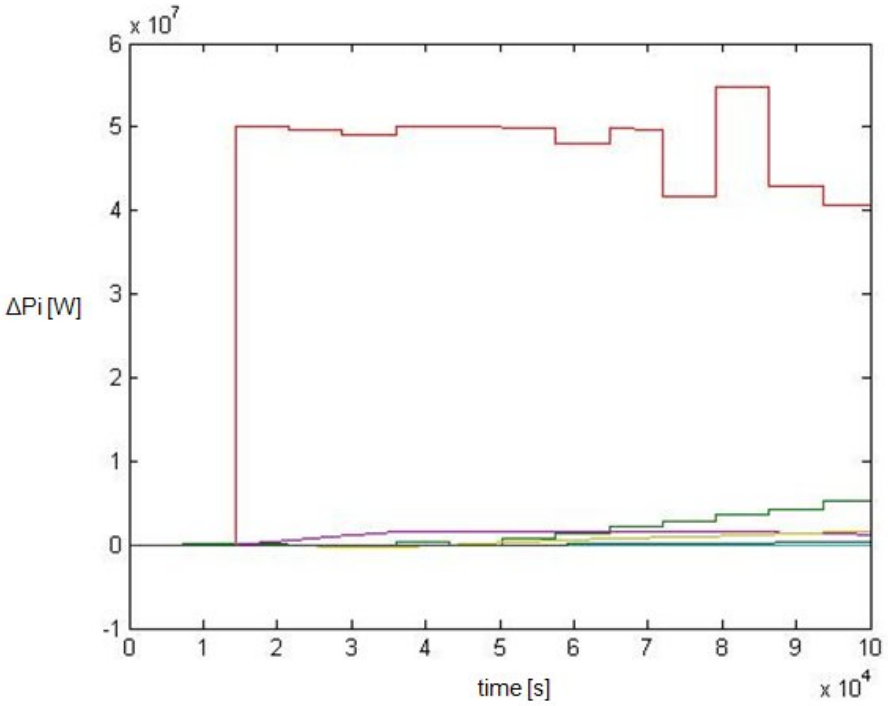

Fig. 7.22: Powers with integral action.

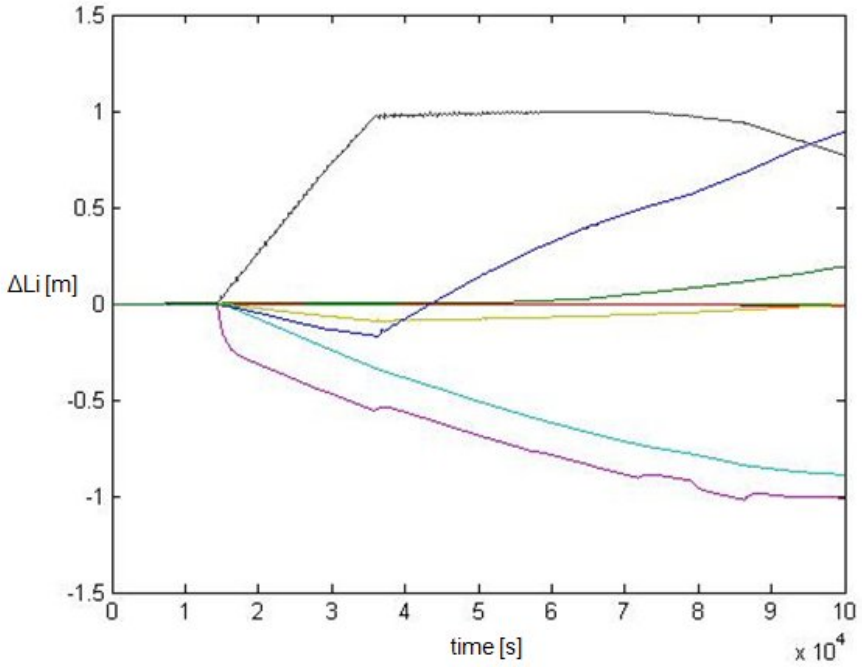

Fig. 7.23: Levels with integral action.

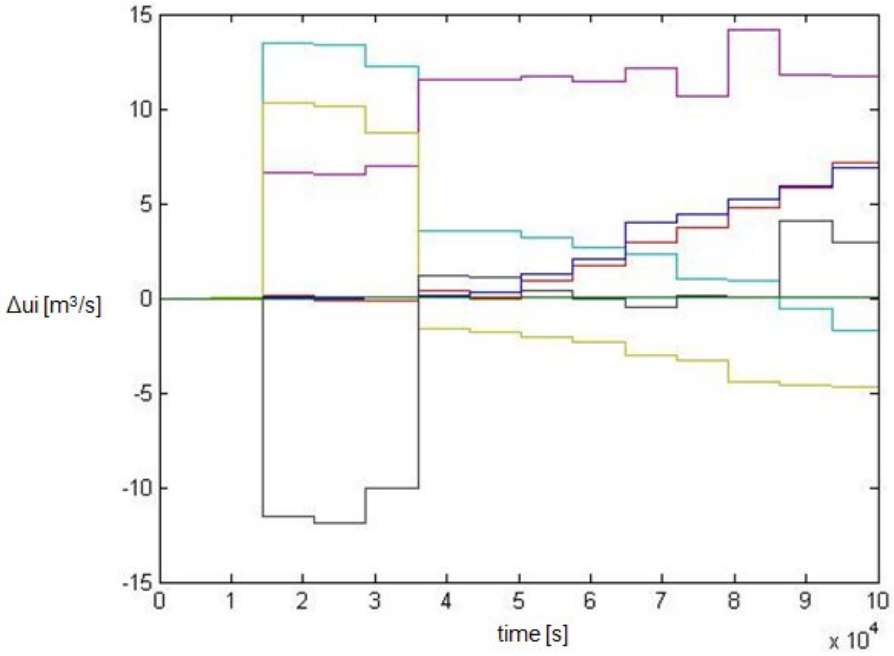

Fig. 7.24: Flow rates with integral action.

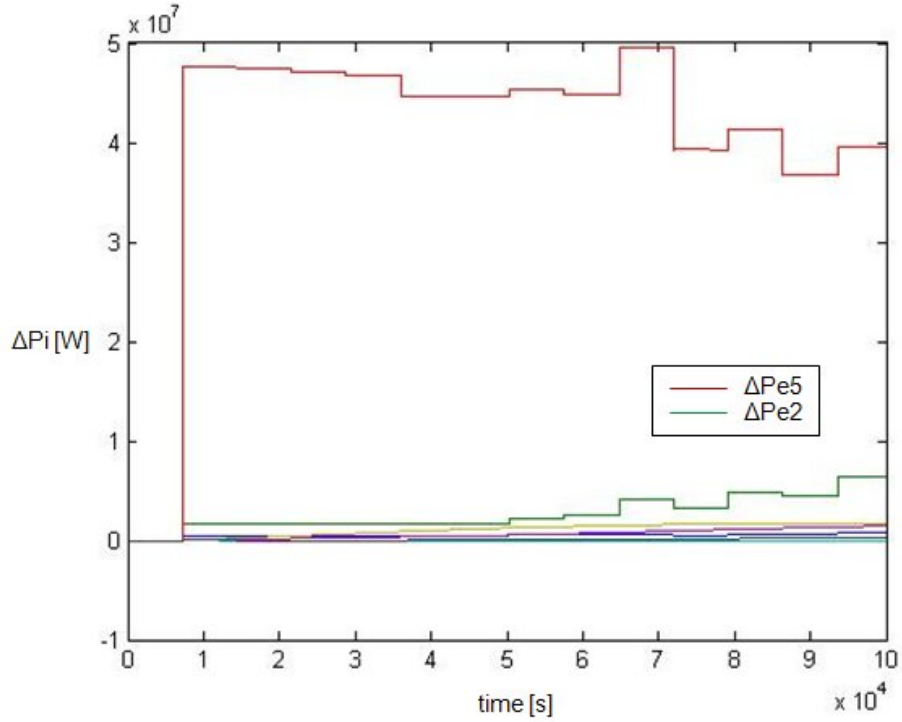

Fig. 7.25: Local powers of the linear model with global set point.

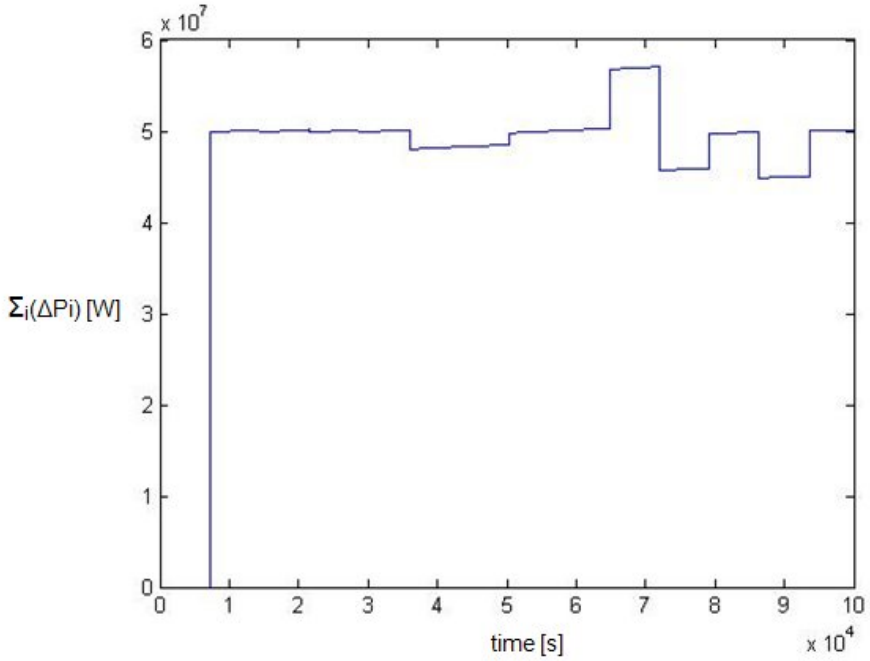

Fig. 7.26: Sum of powers of the linear model with global set point.

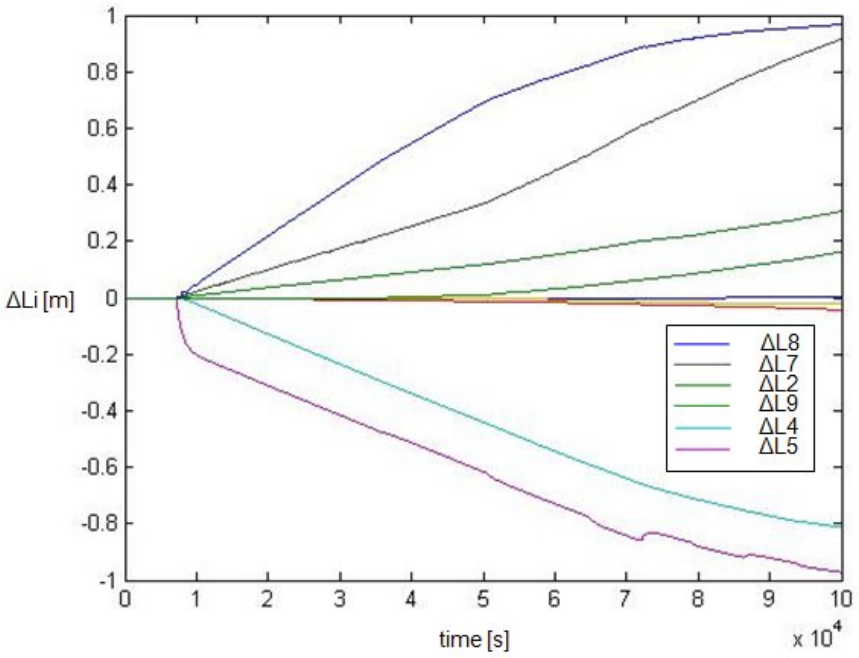

Fig. 7.27: Levels of the linear model with global set point.

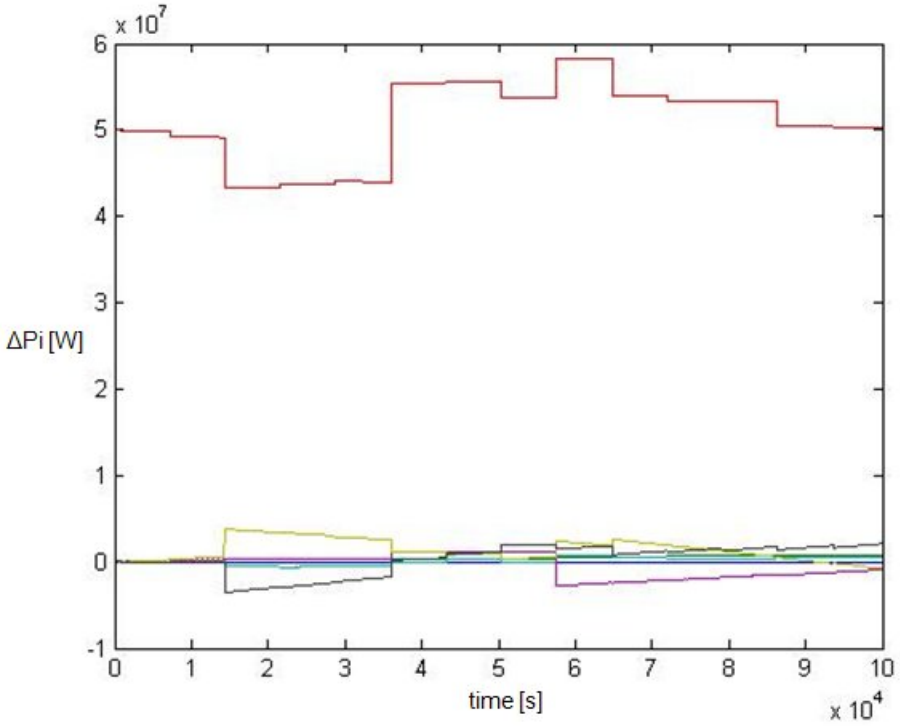

Fig. 7.28: Powers of the nonlinear model with local set point.

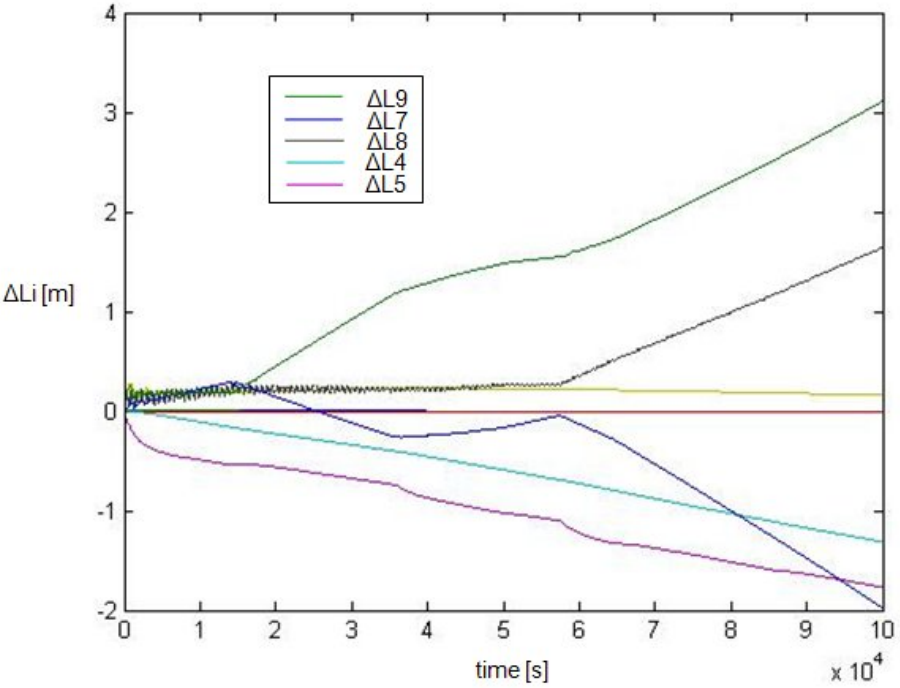

Fig. 7.29: Levels of the nonlinear model with local set point.

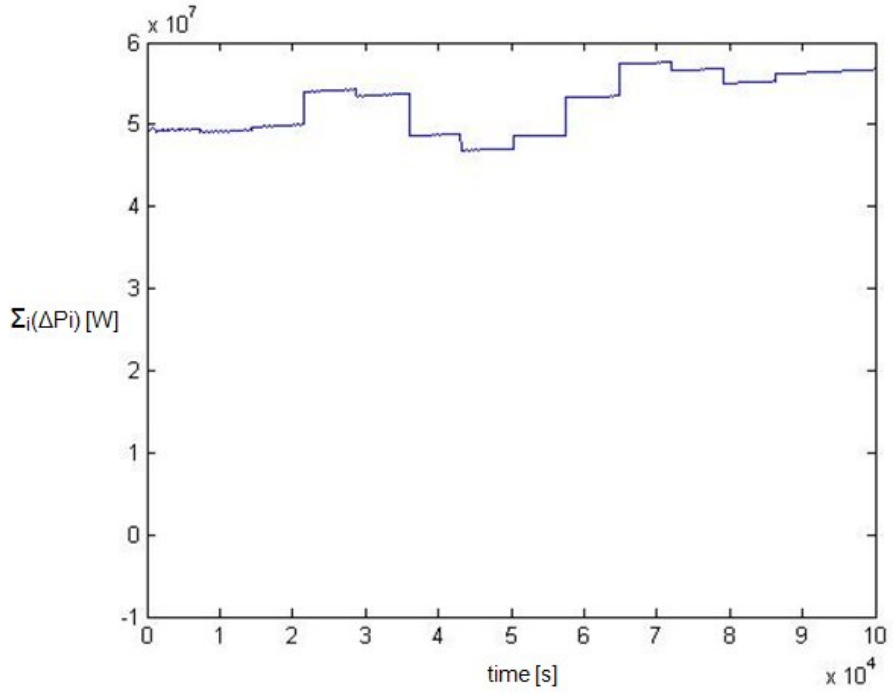

Fig. 7.30: Sum of powers of the nonlinear model with global set point.

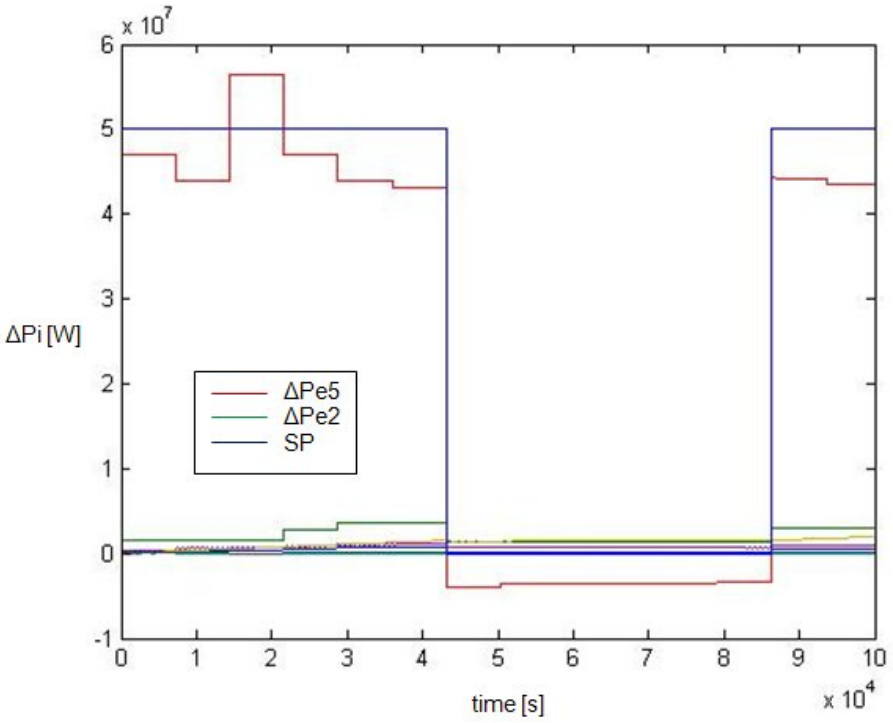

Fig. 7.31: Powers of the nonlinear model with global square-wave-shaped set point.

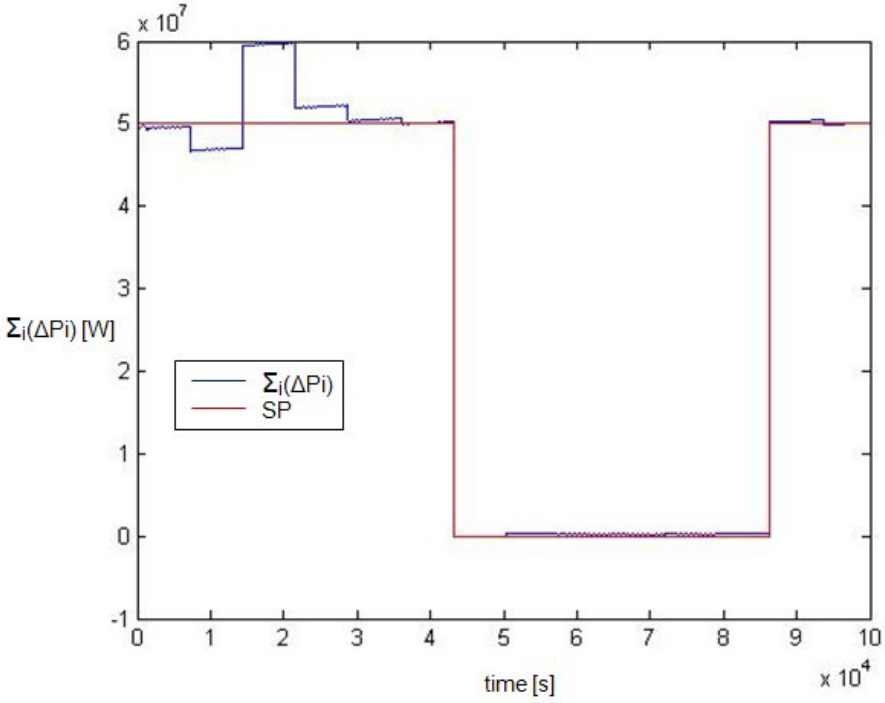

Fig. 7.32: Sum of powers of the nonlinear model with global square-wave-shaped set point.

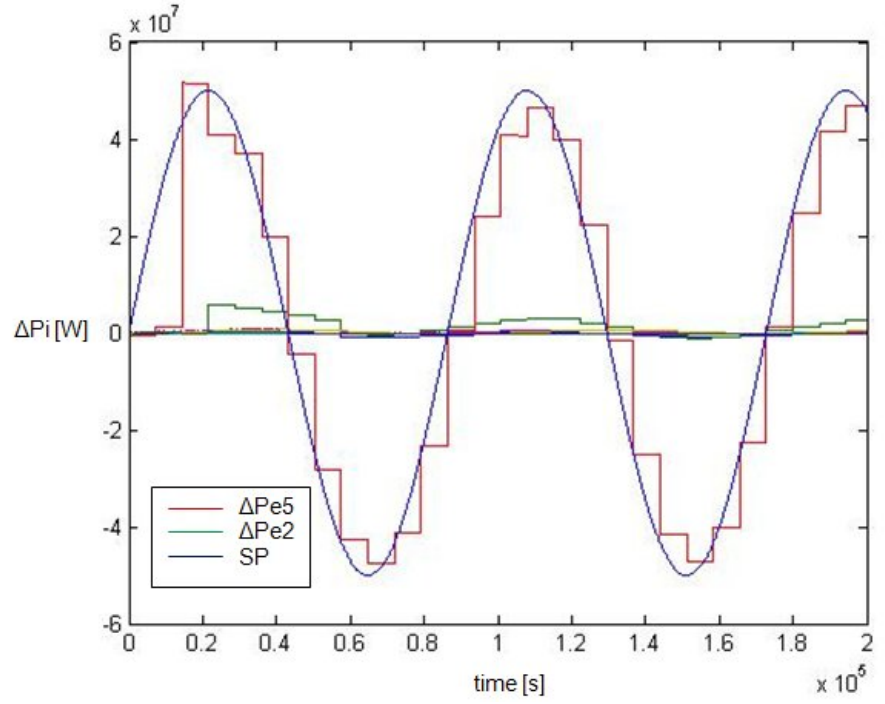

Fig. 7.33: Powers of the nonlinear model with global sinusoidal set point.

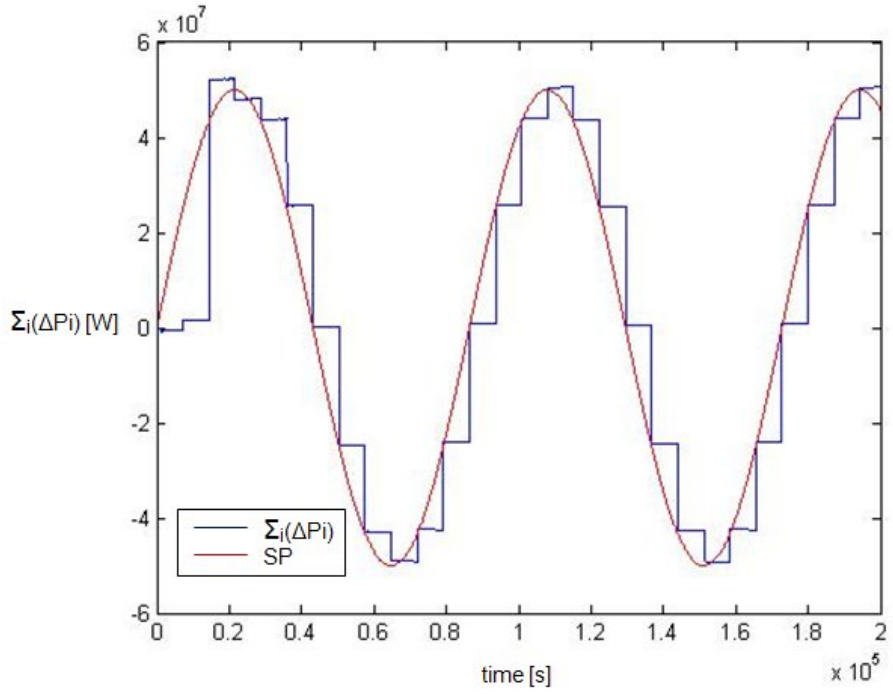

Fig. 7.34: Sum of powers of the nonlinear model with global sinusoidal set point.

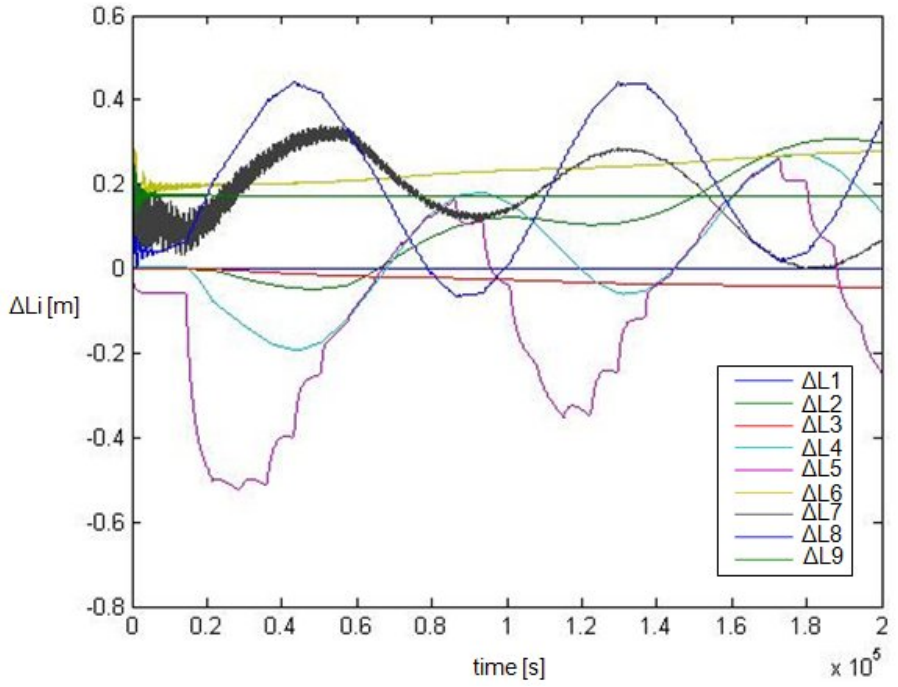

Fig. 7.35: Levels of the nonlinear model with global sinusoidal set point.

Chapter 8

Conclusions and future developments

8.1 Conclusions

In the first part of this Thesis, the model and the simulator of a hydro power valley have been developed. The whole system has been studied and the dynamic equations that describe its model have been derived. In particular, the de Saint Venant equations for the modeling of open-channel hydraulic systems in their different formulations have been considered and their finite-difference approximation has been obtained.
Then, the mathematical model of the hydro power valley has been implemented and simulated in the Matlab-Simulink software environment, paying attention to the numerical aspects guaranteeing the stability of the solution. Subsequently, a linearization of the model has been obtained both with a numerical approach and by means of a symbolic procedure. An accurate analysis of the obtained linear model has been performed, and some interesting conclusions on its structure have been drawn.

Afterwards, a complete MPC controller has been developed, starting from the basic theory of MPC and generalizing it to support both linear or non-linear systems used as prediction models and to manage constraints on the control variables and on the system states and outputs. The problems related to the insertion of an integral action when using a MPC strategy without state observer have been studied and an efficient approach has been proposed.
Finally, the controller has been tested on the hydro power valley model in several working conditions and its performances have been commented.

Although the results achieved can be considered as complete by themselves, this work can also be seen as the basis for future developments. Among them, it is possible to recall the need of include in the problem solution also a state observer, since the MPC control law here derived assumes that the plant states are all measurable.
Moreover, new distributed MPC schemes can be designed to reduce and subdivide the computational task associated to a centralized solution. Indeed, a distributed approach would be more similar to the currently adopted solutions for hydro power valleys, based on decentralized control structures made by simple PID regulators, and would be more robust with respect to faults.
Some guidelines on the development of a distributed controller are presented in the next section, with some advices on the overall architecture to be designed and on the possible ways to perform the decomposition of the global cost function.

8.2 Hierarchical Distributed MPC control

The control of large and networked systems, such as the considered hydro power valley, is usually based on a decentralized control scheme, made by local controllers which ignore the interactions between the different subsystems. Nevertheless, this approach may result in poor performances if the subsystems interact significantly [4].
On the other hand, a centralized MPC, as any other centralized solution, is impractical for the control of large-scale and geographically distributed systems due to the high computational burden associated to the controller and to the high risk of communication losses.

A distributed MPC architecture seems to be a good intermediary step between decentralized and centralized control, as each local subsystem is provided by a local MPC controller which acts on only one (or a few) system variable but, at the same time, takes into account also the interactions with the other subsystems by including some information about their optimization process into its goal function and considering an added set of constraints, which describe its relationship with the variables of the interacting subsystems [4].
Therefore, a feasible distributed architecture for the studied hydro power valley and different advices on how to organize the control algorithm are proposed [17].

8.2.1 Expected hierarchical and distributed architecture

A possible hierarchical and distributed MPC architecture for the hydro power valley is shown in Fig.8.1.
It consists of 2 layers:

- the global optimization layer, named "HPV Optimization";
- local MPC controllers R_i layer.

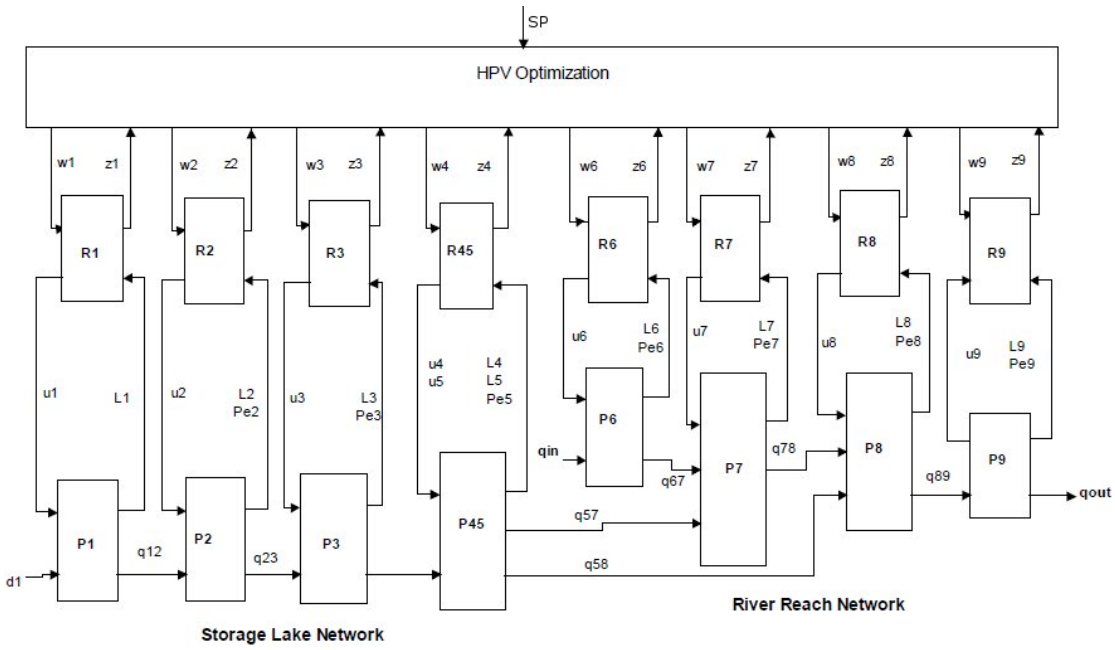

Fig. 8.1: HD-MPC architecture.

Each R_i controller acts directly on one of the local subsystems P_i of the valley, described and modeled in Chapters 2, 3 and 4, and communicates with the other controllers through the higher optimization layer.
The kind of information exchanged between the two layers depends on the method used to decompose the global optimization problem.
Some examples about the decomposition which can be used are presented below [11] [32] [41]:

- *Price decomposition*: the higher layer sends a price vector to the plants, and each of these subsystems minimizes a cost function that depends also on those prices;

- *Quantity decomposition*: the higher layer manages the constraints and sends the set points to the subsystems. Each subsystem minimizes its cost function and sends a price to the coordination. This approach is dual of the previous one;

- *Prediction decomposition*: each subsystem deals with only a subset of the coupling constraints and sends to the other ones a price associated with these constraints, and every subsystem takes into account the prices associated with the constraints that it does not consider;

- *Cascade decomposition*: two loop levels can be considered, a fast loop that regulates the variables around the set-points and manages the physical constraints, and an optimization loop that computes the set-points.

Price, Quantity and Prediction decomposition correspond to a spatial partition, while Cascade decomposition refers to temporal decomposition.
In the following section, the decomposition procedure, its necessary assumptions and the explanation of the mentioned methods are presented.

8.2.2 Objective function decomposition

Assumptions

We assume that the global control problem is formulated as the following constrained optimization problem:

$$u^o = \arg\min_{u,q} J(q,u) \tag{8.1}$$

$$g(q,u) = 0 \tag{8.2}$$

where q are the input flow-rates for each subsystem and u is the discharge reference, which is assumed to be equal to the real discharge (see Section 2.2).
The constraint equation (8.2) considers only the coupling constraints due to the mass conservation between two contiguous subsystems, expressed by equation (8.3).

$$u_i - q_{ij} = 0 \tag{8.3}$$

where u_i is the already defined discharge from the upstream subsystem Pi, while q_{ij} is the inlet flow rate of the downstream subsystem Pj.
In the previous chapters, this relationship has been automatically taken into

account by the links between the subsystems within the Simulink model. In this case, we underline the conceptual difference between a subsystem discharge, named as and equal to its reference u_i, and the inlet flow rate of the next system, named q_{ij}.
For what concerns the other constraints which bound the values of levels, flow rates and flow rate variations (presented in equation (7.17)) and which have been considered in this work, for simplicity it is now worth to not consider them during the presentation of the objective decomposition, as they do not affect the conceptual coordination mechanism.

The primal problem presented in equations (8.1) and (8.2) can be associated to the dual one by applying a lagrangian relaxation, that is the maximization of the dual function $\Psi(\lambda)$.

$$\max_{\lambda} \Psi(\lambda) \tag{8.4}$$

where:

$$\Psi(\lambda) = \min_{u,q}(J(q,u) + \lambda g(q,u)) \tag{8.5}$$

and λ and $g(q,u)$ are column and row vectors.

In the hierarchical an distributed architecture, such a dual problem is decomposed into several local optimization problems, following one of the possible decomposition procedures listed above.

The basic hypothesis to be respected in order to apply the mentioned decomposition solutions is that the objective and the constraint functions must be additive. That is, it must be possible to write the objective and constraint functions as follows:

$$J(q,u) = \sum_{i=1}^{M} J_i(q_{ji}, u_i) \qquad j \neq i \tag{8.6}$$

$$g(q,u) = \sum_{i=1}^{M} g_i(q_{ij}, u_i) \qquad j \neq i \tag{8.7}$$

where M is the number of subsystems.

Under these assumptions, the indicated ways to perform the objective function decomposition are illustrated in detail.

Price decomposition

The price decomposition aims to solve the dual problem ((8.4), (8.5), (8.6), (8.7)) rewritten in equation (8.8).

$$J(q^o, u^o) = \max_{\lambda}(\min_{u_i,q}(\sum_{i=1}^{M} J_i(q_{ji}, u_i) + \lambda \sum_{i=1}^{M} g_i(q_{ij}, u_i))) \tag{8.8}$$

were q is a vector which contains q_{ij} and q_{ji}.

For a price λ, each local controller solves the local problem:

$$[u_i^o, q^o] = \arg\min_{u_i,q}(J_i(q_{ji}, u_i) + \lambda g_i(q_{ij}, u_i)) \tag{8.9}$$

The local problem is in fact a constrained problem itself, but local constraints are ignored for simplicity.
With the new solution for the local problem, the price is adapted with the following equation:

$$\lambda^o = \lambda + \epsilon \sum_{i=1}^{M} g_i(q_{ij}^o, u_i^o) \tag{8.10}$$

The optimal value u_i and λ are not obtained one-shot. So u_i and q_{ij} have to be recomputed with the new λ, and so on until the new value of λ is equal to the previous one. In that case, the equality constraint is satisfied. The solution is admissible (fulfills the constraints) only at convergence.

Quantity decomposition

In the quantity decomposition approach, the terms of the constraint equations (8.7) are considered as quantities θ_i. The upper level imposes values for the quantities such that the constraint equality is fulfilled.

$$g(q, u) = \sum_{i=1}^{M} g_i(q_{ij}, u_i) = \sum_{i=1}^{M} \theta_i = 0 \tag{8.11}$$

For given quantities, the local constrained problems are solved:

$$\min_{u_i,q_{ij}}(\sum_{i=1}^{M} J_i(q_{ji}, u_i)) \tag{8.12}$$

$$g_i(q_{ij}, u_i) = \theta_i \tag{8.13}$$

The local problems can be solved by lagrangian relaxation with the maximization of the dual function:

$$[u_i^o, q^o, \lambda_i^o] = \arg \max_{\lambda_i} \min_{u_i,q} (J_i(q_{ji}, u_i) + \lambda_i(g_i(q_{ij}, u_i) - \theta_i)) \tag{8.14}$$

The coordinator updates the quantity with the price given by the local optimization:

$$\theta_i^o = \theta_i + \epsilon(\lambda_i - \frac{1}{M}\sum_{j=1}^{M} \lambda_j^o) \tag{8.15}$$

As for price decomposition, the update does not immediately give the optimal solution and, according to an iterative procedure, new quantities have to be imposed to the local problem until the convergence is reached.
If the local optimizations succeed, then the solutions meet the constraints even if the optimum is not reached. This could be an advantage over price decomposition where the constraints are satisfied at optimum, but the local optimization is a constrained and more difficult problem, without solution in some cases (the plant is not able to give the quantity by satisfying its local constraints).

Prediction decomposition

Unlike price and quantity decomposition, the prediction decomposition is a method that was originally developed for control purposes.
Each subsystem is responsible of a subset of the constraints. This subset can include for instance the constraint equations with its neighbors given by the equation (8.16), restricted to the variables of the subsystem i (u_i,q_{ji},q_{ij}).

$$g_i(u_i, q_{ij}) = u_i - q_{ij} = 0 \tag{8.16}$$

Then, $g_i(u_i, q_{ij})$ are the constraint equations that are taken into account by subsystem i.

We define μ_j as the price that the other subsystems are willing to pay to the system i to fulfill their own constraints, while the contributions of subsystem i to the constraints of the subsystem j will be denoted by $g_j(u_i, q_{ij})$. We also suppose that we know the interaction variables q_{ij} and the price μ_j given by the subsystem j to the subsystem i.
Then, for subsystem i, the local problem corresponds to the equations (8.17) and (8.18), where a term is added to the objective function to take into account the effect of the local control on the constraints for the subsystem j.

$$\min_{u_i}(J_i(q_{ji}, u_i) + \sum_{j \neq i} \mu_j g_j(q_{ij}, u_i)) \tag{8.17}$$

$$g_i(q_{ji}, u_i) = 0 \tag{8.18}$$

Under certain conditions, this problem is equivalent to the lagrangian augmented problem described in equation (8.19):

$$[u_i^o, q_{ji}^o, \mu_i^o] = \arg\max_{\mu_i} \min_{u_i, q_{ji}}((J_i(q_{ji}, u_i) + \sum_{j \neq i} \mu_j g_j(q_{ij}, u_i)) + \mu_i g_i(q_{ji}, u_i)) \tag{8.19}$$

The solution of problem (8.19) provides the updates for the prices μ_i and the interaction variables q_{ji} that are sent to the subsystem j. In parallel, the j^{th} minimization will compute the new price μ_j and the interaction variable q_{ji} to be sent to the i^{th} subsystem. At convergence the prices must be equal. With this coordination, the exchanges are done on a one-on-one basis. Then, no decision are taken outside the local optimization and the upper level described in Fig.8.1 is just a communication level.

Temporal decomposition

If it is necessary to consider an objective function which is not additive, so it cannot be decomposed over the M subsystems, a cascade decomposition may be implemented.
In this case, the coordination level solves a centralized problem with a simplified model that considers only the slow dynamics. This optimization computes the power and level references for each subsystem, while a lower (centralized) MPC controller applies these set-points on the real system.
However, in order to apply this decomposition method, it is required to split the dynamics of the system in fast and slow dynamics and this may result not trivial. Then, this method should be used only if strictly necessary.

Appendix A

De Saint Venant equations demonstration

In Section 3.2.1 the de Saint Venant equations which describe the one-dimensional hydraulics of a river have been presented in two different forms (see (3.6) and (3.7)), which consist of a mass balance equation and a momentum balance equation.

The mass equation is the same for both formulations and is recalled in (A.1):

$$\frac{\partial Q(x,t)}{\partial x} = \frac{\partial S(x,t)}{\partial t} \tag{A.1}$$

In the sequel, the equivalence of the two forms of the momentum equation is demonstrated.
Initially, the first formulation (3.6) is derived from the general expression of the momentum conservation of a hydraulic system. Subsequently, the second formulation (3.7) is drawn starting from the first one.

A.1 Definitions

Referring to a generic cross section of a river (see Fig.3.2), some useful variables are defined or recalled below:

- t is the time;
- x is the main spatial coordinate of the river;

- dx is a generic infinitesimal section of the main spatial coordinate x;
- $M(x,t)$ is the generic momentum;
- $dM(x,t)$ is the momentum variation;
- $M_i(x,t)$ is the momentum at the beginning of the section;
- $M_o(x+dx,t)$ is the momentum at the end of the section;
- $dm(x,t)$ is the mass of the water in the infinitesimal section;
- $v(x,t)$ is the water speed;
- ρ is the water density;
- $S(x,t)$ is the surface of the wetted cross section;
- $p(x,t)$ is the pressure;
- $P(x,t)$ is the generic impulse;
- $P_i(x,t)$ is the impulse at the beginning of the section;
- $P_o(x+dx,t)$ is the impulse at the end of the section;
- $P_g(x,t)$ is the gravitational impulse;
- $P_f(x,t)$ is the friction impulse;
- $h(x,t)$ is the absolute altitude of the water surface;
- $H(x,t)$ is the altitude of the water surface from the river bed;
- $Q(x,t)$ is the flow rate through the cross section;
- I_f is the friction coefficient;
- I_0 is the river bed slope.

In addiction, some useful equations which connect the previous variables are defined:

$$M(x,t) = dm(x,t)v(x,t) \tag{A.2}$$

$$dm(x,t) = \rho S(x,t)dx \tag{A.3}$$

$$dx = v(x,t)dt \tag{A.4}$$

$$v(x,t) = \frac{Q(x,t)}{S(x,t)} \tag{A.5}$$

$$\frac{\partial h(x,t)}{\partial x} = \frac{\partial H(x,t)}{\partial x} - I_0 \tag{A.6}$$

From (A.2), (A.3) and (A.4), some further formulations of (A.2) can be drawn:

$$\begin{aligned} M(x,t) &= dm(x,t)v(x,t) \\ &= \rho S(x,t)v(x,t)dx \\ &= \rho S(x,t)v^2(x,t)dt \end{aligned} \tag{A.7}$$

A.2 First form of the momentum equation

The general expression of the momentum conservation for a river section is:

$$\begin{aligned} dM(x,t) &= M_i(x,t) - M_o(x+dt,t) + \\ &+ P_i(x,t) - P_o(x+dx,t) + \\ &+ P_g(x,t) + P_a(x,t) \end{aligned} \tag{A.8}$$

where:

$$M_i(x,t) = \rho S(x,t)v^2(x,t)dt \tag{A.9}$$

$$M_o(x+dx,t) = M_i(x,t) + \frac{\partial M_i(x,t)}{\partial x}dx \tag{A.10}$$

$$P_i(x,t) = p(x,t)S(x,t)dt \tag{A.11}$$

$$P_o(x+dx,t) = P_i(x,t) + \frac{\partial P_i(x,t)}{\partial x}dx \tag{A.12}$$

$$P_g(x,t) = -\rho g S(x,t) dh(x,t) dt \tag{A.13}$$

$$P_a(x,t) = -\rho g S(x,t) I_f dx dt \tag{A.14}$$

Substituting equations (A.9), (A.10), (A.11), (A.12), (A.13) and (A.14) in (A.8), leads to (A.15):

$$\begin{array}{rcll} d(\rho S(x,t)v(x,t)dx) & = & -\rho\frac{\partial S(x,t)v^2(x,t)}{\partial x}dtdx & -\frac{\partial p(x,t)S(x,t)}{\partial x}dtdx+ \\ & & -\rho g S(x,t)dh(x,t)dt & -\rho g S(x,t) I_f dtdx \end{array} \tag{A.15}$$

Replacing $v(x,t)$ with (A.5), dividing both members by $\rho \cdot dx \cdot dt$ and separating the second partial derivative, equation (A.16) is obtained:

$$\begin{array}{rcll} \frac{\partial Q(x,t)}{\partial t} & = & -\frac{\partial}{\partial x}(\frac{Q^2(x,t)}{S(x,t)}) & -\frac{S(x,t)}{\rho}\frac{\partial p(x,t)}{\partial x} \quad -\frac{p(x,t)}{\rho}\frac{\partial S(x,t)}{\partial x}+ \\ & & -gS(x,t)\frac{dh(x,t)}{dx} & -gS(x,t)I_f \end{array} \tag{A.16}$$

Since the considered system is supposed to be an non-deep open channel, the pressure along the river can be considered constant and always equal to the atmospheric pressure. Then, the pressure space derivative is null and the contribution of the pressure term to the total momentum variation is negligible with respect to the inertial, gravitational and friction ones.
Therefore, equation (A.16) can be rewritten as:

$$\frac{\partial Q(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left(\frac{Q^2(x,t)}{S(x,t)}\right) - gS(x,t)\frac{dh(x,t)}{dx} - gS(x,t)I_f \tag{A.17}$$

Finally, moving all the terms to the first member and recalling (A.6), the first form of the de Saint Venant momentum equation, presented in (3.6), is found.

$$\frac{\partial Q(x,t)}{\partial t} + \frac{\partial}{\partial x}\left(\frac{Q^2(x,t)}{S(x,t)}\right) + gS(x,t)\frac{dH(x,t)}{dx} + gS(x,t)(I_f - I_0) = 0 \tag{A.18}$$

A.3 Second form of the momentum equation

Once the first form of the de Saint Venant momentum equation has been drawn from the general formulation, the computation of the second form can be performed starting from this result (A.18).
In the sequel, the notation (x, t) will be omitted for simplicity.

Initially, equation (A.18) is divided by S, obtaining (A.19).

$$\frac{1}{S}\frac{\partial Q}{\partial t} + \frac{1}{S}\frac{\partial}{\partial x}\left(\frac{Q^2}{S}\right) + g\frac{dH}{dx} + g(I_f - I_0) = 0 \tag{A.19}$$

Let now consider the partial derivatives $\frac{\partial}{\partial t}\left(\frac{Q}{S}\right)$ and $\frac{\partial}{\partial x}\left(\frac{Q^2}{S^2}\right)$, which are computed in equations (A.20), (A.21) and lead to a new expression for the first and second term of (A.19):

$$\begin{aligned} \tfrac{\partial}{\partial t}\left(\tfrac{Q}{S}\right) &= \tfrac{1}{S}\tfrac{\partial Q}{\partial t} + Q\tfrac{\partial}{\partial t}\left(\tfrac{1}{S}\right) \\ &\Downarrow \\ \tfrac{1}{S}\tfrac{\partial Q}{\partial t} &= \tfrac{\partial}{\partial t}\left(\tfrac{Q}{S}\right) - Q\tfrac{\partial}{\partial t}\left(\tfrac{1}{S}\right) \end{aligned} \tag{A.20}$$

$$\begin{aligned} \tfrac{\partial}{\partial x}\left(\tfrac{Q^2}{S^2}\right) &= \tfrac{1}{S}\tfrac{\partial}{\partial x}\left(\tfrac{Q^2}{S}\right) + \tfrac{Q^2}{S}\tfrac{\partial}{\partial x}\left(\tfrac{1}{S}\right) \\ &\Downarrow \\ \tfrac{1}{S}\tfrac{\partial}{\partial x}\left(\tfrac{Q^2}{S}\right) &= \tfrac{\partial}{\partial x}\left(\tfrac{Q^2}{S^2}\right) - \tfrac{Q^2}{S}\tfrac{\partial}{\partial x}\left(\tfrac{1}{S}\right) \end{aligned} \tag{A.21}$$

Thus, equation (A.19) becomes:

$$\frac{\partial}{\partial t}\left(\frac{Q}{S}\right) - Q\frac{\partial}{\partial t}\left(\frac{1}{S}\right) + \frac{\partial}{\partial x}\left(\frac{Q^2}{S^2}\right) - \frac{Q^2}{S}\frac{\partial}{\partial x}\left(\frac{1}{S}\right) + g\frac{dH}{dx} + g(I_f - I_0) = 0 \tag{A.22}$$

Recalling the de Saint Venant mass equation (A.1), the second term of (A.22) can be rewritten as in (A.23), leading to (A.24).

$$\begin{aligned} -Q\tfrac{\partial}{\partial t}\left(\tfrac{1}{S}\right) &= +\tfrac{Q}{S^2}\tfrac{\partial S}{\partial t} \\ &= -\tfrac{Q}{S^2}\tfrac{\partial Q}{\partial x} \end{aligned} \tag{A.23}$$

$$\frac{\partial}{\partial t}\left(\frac{Q}{S}\right)-\frac{Q}{S^2}\frac{\partial Q}{\partial x}+\frac{\partial}{\partial x}\left(\frac{Q^2}{S^2}\right)-\frac{Q^2}{S}\frac{\partial}{\partial x}\left(\frac{1}{S}\right)+g\frac{dH}{dx}+g(I_f-I_0)=0 \quad \text{(A.24)}$$

Let now consider the expression $\frac{1}{2}\frac{\partial}{\partial x}\left(\frac{Q}{S}\right)^2$, which can replace the sum of the second and the fourth term in (A.24), as proved in (A.25), leading to (A.26).

$$\begin{aligned} -\tfrac{1}{2}\tfrac{\partial}{\partial x}\left(\tfrac{Q}{S}\right)^2 &= -\tfrac{Q}{S}\tfrac{\partial}{\partial x}\left(\tfrac{Q}{S}\right) \\ &= -\tfrac{Q}{S^2}\tfrac{\partial Q}{\partial x}-\tfrac{Q^2}{S}\tfrac{\partial}{\partial x}\left(\tfrac{1}{S}\right) \end{aligned} \quad \text{(A.25)}$$

$$\frac{\partial}{\partial t}\left(\frac{Q}{S}\right)+\frac{\partial}{\partial x}\left(\frac{Q^2}{S^2}\right)-\frac{1}{2}\frac{\partial}{\partial x}\left(\frac{Q^2}{S^2}\right)+g\frac{dH}{dx}+g(I_f-I_0)=0 \quad \text{(A.26)}$$

Finally, adding the second to the third term of (A.26) and dividing by g, the second form of the de Saint Venant momentum equation, presented in (3.7), is found.

$$\frac{1}{g}\frac{\partial}{\partial t}\left(\frac{Q}{S}\right)+\frac{1}{2g}\frac{\partial}{\partial x}\left(\frac{Q^2}{S^2}\right)+\frac{dH}{dx}+(I_f-I_0)=0 \quad \text{(A.27)}$$

Then, the equivalence of the two formulations of the de Saint Venant equations and their correspondence with the generic momentum conservation equation have been demonstrated.

Bibliography

[1] www.edf.com. Web site of EDF.

[2] www.matworks.com. Web site of Matworks.

[3] www.tomopt.com. Web site of Tomlab.

[4] J.B.Rawlings S.J.Wright A.N.Venkat, I.A.Hiskens. Distributed mpc strategies with application to power system automatic generation control. *IEEE Transactions on Control Systems Technology*, 16(6):1192–1206, 2008.

[5] S.J.Wright A.N.Venkat, J.B.Rawlings. *Distributed Model Predictive Control of Large-Scale Systems*. 2007.

[6] M.M.Edvall A.O.Goran. *Tomlab Quick Start Guide*. 2009.

[7] W.W.Symes A.Sei. Gradient calculation of the travel time cost function without ray-tracing. *Center for Research on Parallel Computation Transactions, Rice University, Houston*, 94(15):1–15, 1994.

[8] E. Camacho and C. Bordons. *Model Predictive Control in The Process Industry*. Springer, 1995.

[9] E. Camacho and C. Bordons. *Model Predictive Control*. Springer, 2004.

[10] S.Salsa C.D.Pagani. *Analisi Matematica*. Zanichelli, 2005.

[11] G. Cohen. Optimization with an auxiliary constraint and decomposition. *SIAM Journal on control and optimization*, 28(1):137–157, 1990.

[12] K.Musiake D.Dutta, S.Herath. Flood inundation simulation in a river basin using a physically based distributed hydrologic model. *Hydrological Processes*, 14:497–519, 2000.

[13] Centre d'études techniques maritimes et fluviales. *Guide de prise en main MASCARET*. 2007.

[14] R.Scattolini D.W.Clarke. Constrained receding-horizon predictive control. *Control Theory and Applications, IEEE Proceedings D*, 138:347–354, 1991.

[15] B.H.Krogh S.Talukdar E.Camponogara, D.Jia. Distributed model predictive control. *Control Systems Magazine, IEEE*, 22:44–52, 2002.

[16] E.Gullhamn. *Control of Water Content and Retention in Hydropower Plant Cascades, Master Thesis Project.* PhD thesis, KTH Computer Science and Communication, Stockholm, Sweden, 2004.

[17] D. Faille. Control specification for hydropower valleys. *Seventh Framework Programme, HD-MPC, deliverable D.7.2.1.*, 2009.

[18] G.Glanzmann. *Supervisory Water Level Control for Cascaded Power Plants, PhD thesis.* PhD thesis, IFA - Automatic Control Laboratory, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 2004.

[19] G.Hauke. A stabilized finite element method for the saint-venant equations with application to irrigation. *International Journal for Numerical Methods in Fluids*, 38:963–984, 2002.

[20] M.Wegner J.A.Cunge. Intégration numérique des équations d'écoulement de barré de saint-venant par un schéma implicite de différences finies. *La Houille Blanche*, 1:33–39, 1964.

[21] J.Dubois. *Comportement Hydraulique et Modélisation des Écoulements de Surface.* PhD thesis, École Politechnique Fédérale De Lausanne, 1998.

[22] J.Maciejowski. *Predictive Control with Constraints.* Prentice-Hall, 2001.

[23] K.Holmstrom. *The TOMLAB Optimization Environment in Matlab. Advanced Modeling and Optimization.* 1999.

[24] M.M.Edvall K.Holmstrom, A.O.Goran. *User's Guide for Tomlab Knitro v6.* 2009.

[25] M.M.Edvall K.Holmstrom, A.O.Goran. *User's Guide for Tomlab 7.* 2010.

[26] J.B.Rawlings K.R.Muske. Model predictive control with linear models. *AIChE Journal*, 39:262–287, 1993.

[27] X. Litrico and V. Formon. Infinite dimensional modelling of open-channel hydraulic systems for control purposes. *Proceedings of the Conference on Decision and Control, Las Vegas, Neveda, USA, December 2002.IEEE.*, 2:1681–1686, 2002.

[28] R.Scattolini L.Magni. *Complementi di Controlli Automatici*. Pitagora Editrice Bologna, 2006.

[29] G.Leugering M. Gugat. Global boundary controllability of the de st. venant equations between steady states. *Annales de l'Institut Henri Poincaré (C) Non Linear Analysis*, 20:1–11, 2003.

[30] R.Mose M.Zoaeter M.Abdallah, J.Vazquez. Traitement des conditions aux limites intérieures et extérieures pour la simulation numérique unidimensionnelle de l'écoulement de l'eau dans les canaux à surface libre. *J. Phys. IV France*, 124:207–212, 2005.

[31] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.

[32] M.Jamshidi. *Large-Scale Systems: Modeling and Control*. Elsevier Science Ltd, 1983.

[33] G.Besancon X.Litrico M.Thomassin, D.Georges. Modélisation et identification d'un bief d'irrigation par une méthode de collocation. *Journées Identification et Modélisation Expérimentale JIME 2006, Poitiers, France*, 2006.

[34] M.Kurth V.Hagenmeyer M.Treuer, T.Weissbach. Flatness-based control of a pumped storage power station. *Proceedings of the 17th IFAC World Congress, 2008*, 17:11080–11085, 2008.

[35] M. Nikolaou. Model predictive controllers: A critical synthesis of theory and industrial needs. *Advances in Chemical Engineering*, 26:131–204, 2001.

[36] N.L.Ricker. Model predictive control with state estimation. *Industrial Engineering Chemistry Research*, 29:374–382, 1990.

[37] B.Sayyar-Rodsari N.Motee. Optimal partitioning in distributed model predictive control. *Proceedings of the American Control Conference, 2003*, 6:5300–5305, 2003.

[38] A. Heinrich P. Rostalski G. Papafotiou C. Setz and M. Morari. Application of model predictive control to a cascade of river power plants. *Proceedings of the 17th World Congress The International Federation of Automatic Control, Seoul, Corea, July. 6-11 2008.IFAC*, pages 11978–11983, 2008.

[39] N.Schiavoni P.Bolzern, R.Scattolini. *Fondamenti di Controlli Automatici*. McGraw-Hill, 2nd edition, 2004.

[40] J.P.Baume P.O.Malaterre. Modeling and regulation of irrigation canals: existing applications and ongoing researches. *Proceedings of the International Conference on Systems, Man, and Cybernetics.*, 4:3850–3855, 1998.

[41] A. Rantzer. Dynamic dual decomposition for distributed control. *American Control Conference*, pages 884–888, 2009.

[42] K.Musiake R.Jha, S.Herath. River network solution for a distributed hydrological model and applications. *Hydrological Processes*, 14:575–592, 2000.

[43] R.Xia. Impact of coefficients in momentum equation on selection of inertial models. *Journal of hydraulic research*, 32:615–621, 1994.

[44] S.Bittanti. *Identificazione dei Modelli e Sistemi Adattativi*. Pitagora Editrice Bologna, 2004.

[45] S.Bittanti. *Teoria della Predizione e del Filtraggio*. Pitagora Editrice Bologna, 2004.

[46] T.A.Badgwell S.J.Qin. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11:733–764, 2001.

[47] Q.Zhu S.Li, Y.Zhanga. Nash-optimization enhanced distributed model predictive control applied to the shell benchmark problem. *Information Sciences*, 329-349:137–157, 2005.

[48] M.von Siebenthal T.Geyer G.Papafotiou M.Morari G.Glanzmann. Supervisory water level control for cascaded river power plants. *Technical Report CH-8092 Zurich, Automatic Control Laboratory, Swiss Federal Institute of Technology (ETH), Switzerland*, pages 1–10, 2005.

Ringraziamenti

Un doveroso e sentito ringraziamento va al Prof.Riccardo Scattolini, che con infinita pazienza ha accompagnato e supervisionato questa esperienza, sapendomi dare con professionalità le dritte giuste nei momenti di difficoltà e dimostrandomi una fiducia costante che ha giovato non poco alla buona riuscita del lavoro.

Warm thanks to Dott.Damien Faille, who was my guide during my stay in Chatou and shared his office and a lot of his precious time with me, adding his knowledge and experience to my work. Thanks also for his delicious pastries.
Thanks to all the people who helped me with their competence, such as Frans Davelaar, Rudy Negenborn and Marcus Edvall.

Je ne peux pas oublier de remercier touts les stagiaires de le groupe P1B de le département STEP e touts ces qui ont partagé avec moi en amitié cette si belle expérience à Chatou. Merci à Pierre-Alban, qui a partagé avec moi les premiers jours de travail et m'a fait sentir moins seul dans un si grand site. Merci à Adrian, pour sa amitié et aussi pour sa folie... mais surtout pour sa amitié. Merci à David, pour toutes les conversations de bon matin avec le département encore vide et pour les précieux conseils sur les fromages Français. Merci à Jennifer, pour la gentillesse que m'a toujours démontré. Merci à Fanny, Paula, Brice et Sidath, pour notres pic-nics sous la pluie et pour ces avec le soleil, mais surtout merci parce-que votre présence m'a fait arriver au travail avec le sourire chaque matin.

Un grosso grazie va al mio compagno di avventura, Fabio, con cui ho condiviso aspettative, preoccupazioni, idee e, soprattutto, bei momenti. Grazie per l'intraprendenza, la compagnia e l'amicizia, che sono state il valore aggiunto di questa esperienza.

Ma forse l'avventura più grande sono stati i 5 anni tra i banchi del Poli,

dove di persone ne son passate tante ed è capitato che qualcuno di loro, guardacaso, incrociasse la mia strada. Grazie a Cala, Giagia, Wolf, Flavio, Stefano, Alessio, Mario, Marco, Endri, Michelino, Cerro, Tose, Max, Pala, Ale, Dome, Beppe, Diego, Marco, Colz, Barce, Davide, Marco, Gio, Davide, Beppe il Matto, Canta e il mitico Abe che da otto anni ormai mi sopporta e che, esame dopo esame, è diventando il mio compagno di classe doc. Grazie a tutti.
Un grazie speciale a Gabriele, per tutte le "colazioni abbondanti" prima di ogni esame e per l'amicizia che ci ha accompagnato durante questa piccola, ma lunga, battaglia.

E poi ci sono loro. Unici, irripetibili.
Ciossa, Anna, Vicky, Vale, Albe, Casa, Asto, Lamas.
Semplicemente, grazie di esserci.

E che senso avrebbero, queste pagine, questi grafici, questi simboli strani, questo pezzo di carta e tutto ciò che ci gira attorno, se non ci fosse qualcuno che ti ama da cui tornare e su cui posare la testa? Grazie Dani.

L'ultimo grosso grazie va alla mia famiglia ed ai miei genitori, per tutto il bene che mi volete e per avermi permesso di arrivare fin qui. Grazie a mamma Angela, per l'amore che mette in tutto quello che fa. Grazie a papà Icilio, che ha sempre creduto in me e a cui dedico questo traguardo. Grazie alla nonna Tina, che è sempre lì a dire con i gesti che ci vuole bene. Grazie a Chiara e Anna, che restano sempre le mie sorelline preferite.
E un grazie alla persona che per prima mi disse: "da grand te farem fà l'ingegner". Grazie nonna Maria.