DYNAMIC PROGRAMMING PREDICTIVE CONTROL

A. P. de Madrid, S. Dormido, F. Morilla, L. Grau

Dpto. de Informática y Automática, UNED. Senda del Rey, s/n
28040 MADRID (SPAIN)
Phone: +34 (1) 398 71 60 Fax: +34 (1) 398 66 97 E-mail: angel.perez@uned.es

Abstract: This paper deals with the development and characterization of Dynamic Programming Predictive Controllers (DPPC), advanced predictive controllers that make use of dynamic programming to solve highly nonlinear / nonquadratic constrained control problems. Some computational techniques to reduce the computational load are proposed, and an application example, *p*H control, is analyzed using this new methodology.

Keywords: Predictive control, Dynamic programming, Numerical algorithms, Optimization problems, Computational methods, pH control

1. INTRODUCTION

Model Predictive Control (MPC) was developed during the late 70's. Since then, it has increased its popularity due to the important advantages that offers over other control methods (De Keyser, 1992).

From a mathematical point of view, MPC is not but a constrained optimization problem that can be usually solved as a quadratic programming (QP) problem: when the process model is linear, the objective function is quadratic and the constraints are linear.

Dynamic Programming (DP), based on Bellman's Principle of Optimality (Bellman, 1957), is a very general optimization tool, more powerful than QP. Making use of DP, it is possible to develop new types of predictive controllers with more general and highly nonlinear models and constrains, and nonquadratic objective functions (the latter is important, as Chow, *et al.* (1995) have shown that the use of multiple models in predictive control leads to a cost function that is not quadratic).

The main aim of this paper is to introduce a predictive controller based on DP. In section 2 predictive controllers are introduced, paying special attention to their formulation as constrained optimization problems. Section 3 deals with dynamic programming. The main drawback of DP is its extremely high computational complexity; so, section 4 studies some important methods for the reduction of the complexity. In section 5 the predictive controller based on dynamic programming is developed. Finally, in section 6 a highly nonlinear control problem is solved: the *p*H control.

2. MODEL PREDICTIVE CONTROL

All predictive controllers share a common key characteristic: *they make use of a model of the process to obtain the control signal by minimizing a given cost*

function. The MPC-methodology is defined by the following strategy (Clarke, *et al.*, 1987; De Keyser, 1992; Camacho and Bordóns, 1995): the optimal future control signal, $\Delta \boldsymbol{u}^*$, is obtained in such a way it minimizes a given *cost function* of the predicted future errors, $[\boldsymbol{r}(t+k|t) - \boldsymbol{y}(t+k|t)]$, over a long time prediction horizon N_2:

$$\begin{aligned} &J\left(\Delta \boldsymbol{u}(t|t),\ldots,\Delta \boldsymbol{u}(t+N_u-1|t),t\right)= \\ &= E\left\{\sum_{j=N_1}^{N_2}\gamma(j)\left[\boldsymbol{r}(t+j|t)-\boldsymbol{y}(t+j|t)\right]^2+\right. \\ &\left.+\sum_{j-1}^{N_u}\lambda(j)\left[\Delta \boldsymbol{u}(t+j-1|t)\right]^2\right\} \end{aligned} \tag{1}$$

subject to $\Delta \boldsymbol{u}(t+j-1|t) = 0$, $N_u+1 \le j \le N_2$, where $E(\cdot)$ stands for *expected value*, as predictions are made based on the information available at the instant t. $\boldsymbol{y}(t+k|t)$, $k = 1, 2, \ldots N_2$, are the predicted outputs of the process, based on the process model. $\boldsymbol{r}(t+k|t)$ is a reference trajectory, describing how the current process output, $\boldsymbol{y}(t)$, should tend to the —predicted— future setpoint, $\boldsymbol{w}(t+k|t)$. The first element of the optimal control vector, $\Delta \boldsymbol{u}^*(t|t)$, is actually applied to the process. At the next sampling instant a new output measurement $\boldsymbol{y}(t+1)$ is obtained, which is generally different to the predicted value $\boldsymbol{y}(t+1|t)$. Therefore all the other elements of $\boldsymbol{u}^*$ are forgotten, all the sequences are shifted and the whole procedure is repeated. This is called a *receding horizon* strategy.

In the absence of constraints, it is possible to obtain the solution of the minimization problem in closed form. But inequality constraints appear in real control problems and the solution has to be obtained numerically.

Most predictive controllers are based on linear models (including the dynamics of the disturbances), quadratic cost functions and linear constraints. In this case, the optimization problem has the general expression

$$\begin{aligned} &\boldsymbol{x}^* = \operatorname{argmin}\left(\frac{1}{2}\boldsymbol{x}^T\mathbf{H}\boldsymbol{x}+\mathbf{G}^T\boldsymbol{x}\right) \\ &\text{subject to} \quad \mathbf{A}\boldsymbol{x} \le \boldsymbol{b} \end{aligned} \tag{2}$$

where $\boldsymbol{x}$ is the manipulated variable ($\boldsymbol{u}$ or $\Delta \boldsymbol{u}$), $\boldsymbol{x}^*$ is the optimal solution, $\mathbf{H}$ is the Hessian matrix and $\mathbf{G}$ is the gradient vector. This problem is known as *quadratic programming*, for which there exists a deep theoretical and practical background.

Whenever the models are nonlinear (a more realistic situation), the cost function is non-quadratic, or the constraints are nonlinear, the optimization cannot be solved as a quadratic programming problem. It does not exist a general theory for such nonlinear / non-quadratic predictive controllers. For them, *ad hoc* numerical methods have to be used, or the problem has to be linearized.

3. DYNAMIC PROGRAMMING

3.1. Backward Dynamic Programming

Dynamic Programming (DP) is an analytic tool created to solve a set of optimization problems known as *multistage decision processes*. One of the most important ones is as follows (Larson and Casti, 1978):

Minimize (maximize) the separable cost (performance) function

$$J = \sum_{k=0}^{N} L\left(\boldsymbol{x}(k),\boldsymbol{u}(k),k\right) \tag{3}$$

where

$$\boldsymbol{x}(k+1) = g\left(\boldsymbol{x}(k),\boldsymbol{u}(k),k\right) \tag{4}$$

subject to

$$\boldsymbol{x} \in X(k) \subset \Re^n, \quad \boldsymbol{u} \in U\left(\boldsymbol{x}(k),k\right) \subset \Re^m \tag{5}$$

◊◊◊

In this problem, $\boldsymbol{x}$ is the *state variable*, $\boldsymbol{u}$ is the *decision* or *control variable* and k is the *stage*.

Defining:

$$I(\boldsymbol{x},k) = \min_{\boldsymbol{u}(k),\boldsymbol{u}(k+1),\ldots,\boldsymbol{u}(N)}\left\{\sum_{j=k}^{N} L\left(\boldsymbol{x}(j),\boldsymbol{u}(j),j\right)\right\} \tag{6}$$

and based on Bellman´s Principle of Optimality (Bellman, 1957; Larson and Casti, 1978), it is possible to prove that:

$$\begin{aligned} &I(\boldsymbol{x},k) = \min_{\boldsymbol{u}}\left\{L(\boldsymbol{x},\boldsymbol{u},k)+I\left[g(\boldsymbol{x},\boldsymbol{u},k),k+1\right]\right\} \\ &I(\boldsymbol{x},N) = \min_{\boldsymbol{u}(N)}\left\{L\left(\boldsymbol{x},\boldsymbol{u}(N),N\right)\right\} \end{aligned} \tag{7}$$

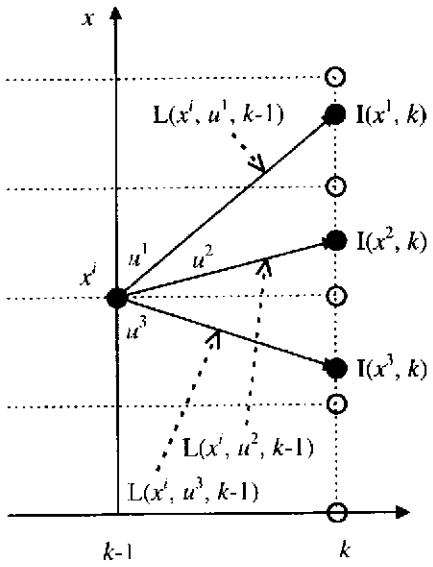

Fig. 1. BDP computational procedure

This iterative functional equation can be solved numerically. To do so, the sets of admissible states, X, and decisions, U, have to be quantified to a finite number of values

$$\mathrm{X} = \left\{ \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^{M_X} \right\}$$
$$\mathrm{U} = \left\{ \boldsymbol{u}^1, \boldsymbol{u}^2, \ldots, \boldsymbol{u}^{M_U} \right\} \qquad (8)$$

defining a *computational grid*; M_X and M_U can be different at each stage k. For all the stages k, and for all the quantified states $\boldsymbol{x}^j(k)$, the quantity inside the braces in (7) is evaluated for each $\boldsymbol{u} \in \mathrm{U}$, and these values are compared to determine the smallest one (fig. 1). The control $\boldsymbol{u}$ that produces the smallest cost is the optimum control $\boldsymbol{u}^*$ for $\boldsymbol{x}(k)$ at stage k. In order to evaluate $\mathrm{I}[\mathrm{g}(\boldsymbol{x}, \boldsymbol{u}, k), k+1]$, when $\mathrm{g}(\boldsymbol{x}, \boldsymbol{u}, k)$ is not a quantified state, an interpolation algorithm can be used. Starting at stage N, this procedure is applied backwardly, towards stage 1. Thus, this method is called *Backward Dynamic Programming* (BDP).

One of the most important properties of this method is that constraints of a very general nature can be handled, and they actually *reduce* the computational effort rather than increase it, as they reduce the number of admissible controls and states to be evaluated.

BDP always determines an absolute minimum, within the accuracy of the computational grid. The optimal initial and final states are determined by BDP, if they are not defined in advance.

The main drawback of this method is a very high computational complexity: although the constraints actually reduce the number of trajectories that must be considered, the number of computations is still too high. Furthermore, it can be proved (Guignabodet, 1961, 1963) that the only way to obtain an accurate solution is to increase the number of quantified controls and states, not just arbitrarily increase the order of the interpolation procedure.

3.2. *Forward Dynamic Programming*

It is possible to obtain a different functional equation that can be solved forwardly. The minimum cost function I' (Larson and Casti, 1978) is defined as

$$\mathrm{I}'(\boldsymbol{x}, k) = \min_{\boldsymbol{u}(0), \boldsymbol{u}(1), \boldsymbol{u}(k-1)} \left\{ \sum_{j=0}^{k-1} \mathrm{L}(\boldsymbol{x}(j), \boldsymbol{u}(j), j) \right\} \qquad (9)$$

Then, the iterative functional equation becomes:

$$\mathrm{I}'(\boldsymbol{x}, k) = \min_{\boldsymbol{u}} \left\{ \mathrm{L}\left[\mathrm{g}^{-1}(\boldsymbol{x}, \boldsymbol{u}, k-1), \boldsymbol{u}, k-1 \right] + \right.$$
$$\left. + \mathrm{I}'\left[\mathrm{g}^{-1}(\boldsymbol{x}, \boldsymbol{u}, k-1), k-1 \right] \right\} \qquad (10)$$

where $\mathrm{g}[\mathrm{g}^{-1}[\boldsymbol{x}(k+1), \boldsymbol{u}(k), k], \boldsymbol{u}(k), k] = \boldsymbol{x}(k+1)$.

This method, called *forward dynamic programming* (FDP), has the same properties and computational complexity that BDP, and is specially useful when the initial state is defined in advance.

4. REDUCTION OF DIMENSIONALITY

In order to obtain an accurate solution, the step of the computational grid has to be small, but then the computational complexity of dynamic programming is extremely high. There exist some methods to reduce it, making possible to apply dynamic programming to some interesting problems of the real world.

4.1. *Special partitions*

Nonuniform partitions. The step size is smaller in those regions of the state space and/or the control space where more precision is needed (Moreno, *et al.*, 1992; Moreno, *et al.*, 1994; P. de Madrid, *et al.*, 1994; P. de Madrid, 1995).

Coarse partitions. A coarse partition of the state space is initially used (Bellman and Dreyfus, 1962; P. de Madrid, 1995). Once the solution has been obtained, it is defined a band around it, where a new smaller partition is defined. Then a new DP problem is solved in this band, and all the procedure is repeated until the desired grid size is reached. The main drawback of this method is that a local optimum can be obtained if the initial grid is too coarse.

4.2. *Nominal trajectories*

If a good nominal initial solution is available, it can be used in two different ways to reduce the computational complexity:

Direct iteration. If the initial state $\boldsymbol{x}_0$ is defined, a nominal solution, starting from $\boldsymbol{x}_0$, is postulated. The DP problem is only solved in a band around the postulated solution. This method can be combined with the coarse partitions method.

Successive approximations in the state space. Let consider a problem with as many state variables as control variables ($n = m$) (Bellman and Dreyfus, 1962; Larson, 1968; Korsak and Larson, 1970; Larson and Korsak, 1970; Cooper and Cooper, 1981; Larson and Casti, 1982). Let $\{\boldsymbol{x}^{(0)}(k)\}$ be the nominal solution. One of the n state variables is selected, x_{i1}, and a DP problem is solved, in such a way that $\{x_i^{(0)}(k), i \neq i_1\}$ does not change. It means that there exist $(n - 1)$ equality constraints in the control variables, and therefore the problem has only one degree of freedom. The solution of this problem generates a new nominal trajectory, $\{\boldsymbol{x}^{(1)}(k)\}$. Then a new state variable is selected, x_{i2}, and the method is repeated. This is done for all the n state variables. Then the method is repeated, until it converges to the same solution for all the state variables.

This method reduces the complexity to linear, and can be generalized for the general case ($n \neq m$). But convergence to the true optimum (even to a local one) can be only guaranteed for a few cases.

4.3. *System dynamics*

Convexity of the set of admissible controls. If the system equations are linear and the set of admissible states for each stage is convex, then the set of admissible controls for each state for each stage is convex (Ruipérez, 1977; P. de Madrid, 1995). This can be used to reduce the number of admissible controls to test: if $\boldsymbol{u}^1$ and $\boldsymbol{u}^2$ are the minimum and maximum admissible controls, respectively, and the system equations are linear, then the optimal control $\boldsymbol{u}^*$ must verify $\boldsymbol{u}^1 \leq \boldsymbol{u}^* \leq \boldsymbol{u}^2$, and thus the number of quantified controls to test for each state is reduced.

Limit trajectories and reachable states. If the initial state $\boldsymbol{x}(0)$ is defined, it is possible to bound a region in the state space by means of the *limit trajectories*: those state space trajectories obtained applying the extreme controls to the system equations. The only states to explore, the *reachable* states, are confined in the region defined by these trajectories (Dorrnido, *et al.*, 1970). If the final state $\boldsymbol{x}(N)$ is defined, this region can be reduced even more.

5. DYNAMIC PROGRAMMING PREDICTIVE CONTROL

The application of DP to process control is not a new subject. In fact, some optimum control problems can be expressed in terms of a Bellman's equation, that can be solved *analytically* when the system is linear, the objective function is linear or quadratic, the stochastic variables are Gaussian and there are no constraints (Larson and Casti, 1982; Pierre, 1986; Mosca, 1995). But most problems can be only solved *numerically*.

Current MPCs make use of linear systems and constraints and quadratic cost functions, due to the limitations of QP algorithms. But QP can be considered a subset of DP problems. So, it is possible to develop a predictive controller based on DP.

P. de Madrid (1995) proposes a general DPPC algorithm. This is a hybrid one —combining BDP and FDP— with no interpolation, optimized for MPC. It makes use of the techniques described in section 4, in such a way the computational cost is dramatically reduced, and can be easily generalized to include interpolation.

6. APPLICATION EXAMPLE: pH CONTROL

*p*H control presents difficulties due to large variations in process dynamics. *p*H is a measure of hydrogen ions H^+ in a solution, and is defined by $pH = -\log[H^+]$, where $[\cdot]$ stands for concentration. Let consider the neutralization of a strong acid-base pair, as it is proposed in (Åström and Wittenmark, 1989). A strong acid effluent is fed to a tank with volume V (l); the acid concentration in the influent stream is c_A (mol/l) and its flow is q (l/s). The acid is neutralized with a strong base of concentration c_B (mol/l) and flow u (l/s). Let x_A and x_B be the

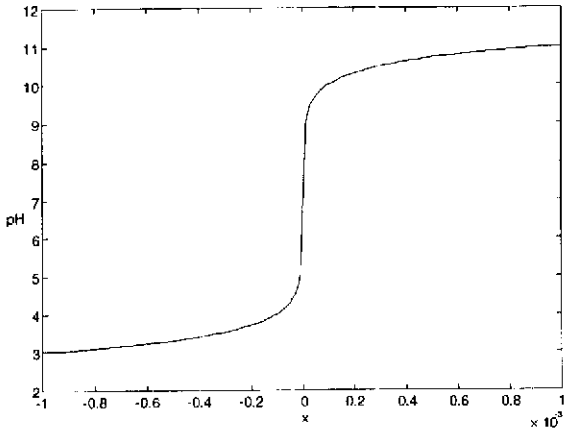

Fig. 2. Titration curve for (12)

concentrations of acid and base in the tank. The system dynamics is then given by

$$\begin{cases} \dfrac{dx_A}{dt} = \dfrac{q}{V}(c_A - x_A) \\ \dfrac{dx_B}{dt} = \dfrac{u}{V}c_B - \dfrac{q}{V}x_B \end{cases} \qquad (11)$$

and the pH is given by

$$p\mathrm{H}(x) = -\log\left(\sqrt{\frac{x^2}{4} + K_w} - \frac{x}{2}\right) \qquad (12)$$

where $x = x_B - x_A$ and $K_w = 10^{-14}$ (mol/l)^2 at 25°C.

The dynamics of the pH sensor and the pump will be modeled as first order transfer functions with time constant τ.

Åström and Wittenmark (1989) show that, assuming proportional control, the critical gain varies drastically — several orders of magnitude, see fig. 2— for different values of the reference pH. The difficulties are compounded by the presence of time delays and flow variations.

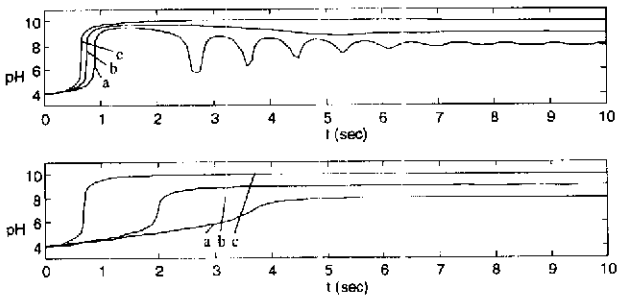

Fig. 3. PI performance: pH control (top) and concentration control (bottom). $p\mathrm{H}_{\mathrm{ref}}$ is 8 (a), 9 (b) and 10 (c)

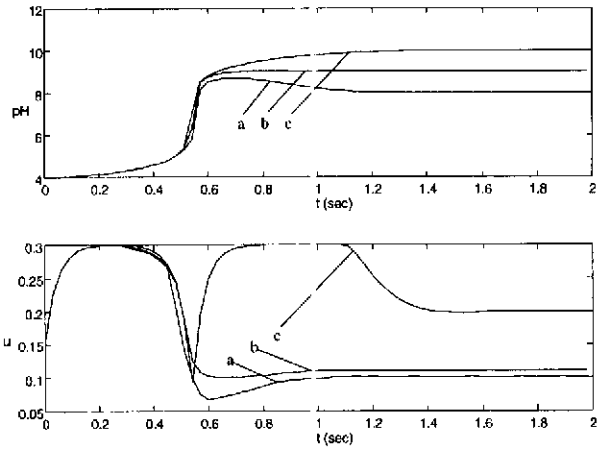

Fig. 4. DPPC performance. $p\mathrm{H}_{\mathrm{ref}}$ is 8 (a), 9 (b) and 10 (c)

Figure 3 (top) shows this situation: assuming $V = 1000$ l, $q = 1000$ l/s, $c_A = 10^{-4}$ mol/l, $c_B = 1$ mol/l, $\tau = 0.1$ s and a sampling period of $T = 0.03$ s, a PI (or PID) controller that has been tuned for a given setpoint (10) has a degraded performance for a different one, and can become unstable. Equation (11) shows that the system depends linearly on the concentrations, so performance is better if concentration is controlled instead of pH. In this case (fig. 3, bottom), the same PI can control the three different setpoints, but the system response degrades quickly.

DPPCs can handle this situation in a natural way. Figure 4 shows the DPPC performance under the same operation conditions assumed in fig. 3. N_1, N_u, N_2 and λ have been set to 1, 1, 10 and 0, respectively, with a control grid of 0.0025 and $u_{max} = 0.3$ mol/l. The control signal has been filtered to eliminate the possible oscillations induced by the quantified values of the variables. The main advantages are:

- The system has not to be linearized. The real system dynamics is taken into account, as a nonlinear model is considered.
- DPPC can control several setpoints with a single tuning. For all of them the system output is good and stable, and much faster than the PI response.

7. CONCLUSION

This paper has introduced DPPCs, advanced predictive controllers that make use of dynamic programming to solve highly nonlinear / nonquadratic constrained control problems. Computational techniques have been proposed to reduce the computational load. Nowadays, real time applications are only available for slow processes, with sampling periods between several seconds to several minutes (typical industrial processes). Future computational advances (faster computers, multiprocessors and parallel algorithms) will make new applications come true.

The application example, *p*H control, has shown how it is possible to work with the exact nonlinear equations. As the nonlinear model takes into account all the complex system dynamics, the performance of the DPPC controller is much better.

8. ACKNOWLEDGMENTS

The authors wish to acknowledge the economical support of the Spanish CICYT, under grant Ref. TAP 95-0790.

9. REFERENCES

Åström, K. J. and B. Wittenmark (1989). *Adaptive control*. Addison-Wesley, Reading, MA.

Bellman, R. E. (1957). *Dynamic programming*. Princeton University Press, New Jersey.

Bellman, R. E. and S. E. Dreyfus (1962). *Applied dynamic programming*. Princeton University Press, New Jersey.

Camacho, E. F. and C. Bordóns (1995). *Model predictive control in the process industry*. Springer-Verlag, New York.

Chow, C.-M., A. G. Kuznetsow and D. W. Clarke (1995). Using multiple models in predictive control. *Proc. of 3rd European Control Conference*, 1732-1737. Rome.

Clarke, D. W., C. Mohtadi and P. S. Tuffs (1987). Generalized predictive control. Part I. The basic algorithm. *Automatica*, **23**, 137-148.

Cooper, L. and M. W. Cooper (1981). *Introduction to dynamic programming*. Pergamon Press, Oxford.

De Keyser, R. M. C. (1992). The MPC Methodology. *Intensive Training Course on Model Based Predictive Control*. U.N.E.D., Madrid.

Dormido, S., M. Mellado and J. M. Guillen (1970). Consideraciones sobre la regulación de sistemas mediante técnicas de programación dinámica. *Revista de Automática*, **10**, 29-34. Madrid. (In Spanish)

Guignabodet, J. J. G. (1961). *Analysis of some process control aspects of dynamic programming*. Ph. D. Thesis. Washington University.

Guignabodet, J. J. G. (1963). Dynamic programming: cumulative errors in the evaluation of an optimal policy. *J. Basic Eng.*, June, 151-156.

Korsak, A. J. and R. E. Larson (1970). A dynamic programming successive approximations technique with convergence proofs. Part II. *Automatica*, **6**, 253-260.

Larson, R. E. (1968). *State increment dynamic programming*. American Elsevier Publishing Company, Inc., New York.

Larson, R. E. and A. J. Korsak (1970). A dynamic programming successive approximations technique with convergence proofs. Part I. *Automatica*, **6**, 245-252.

Larson, R. E. and J. L. Casti (1978). *Principles of dynamic programming. Part I: Basic analytic and computational methods*. Marcel Dekker, Inc., New York.

Larson, R. E. and J. L. Casti (1982). *Principles of dynamic programming. Part II: Advanced theory and applications*. Marcel Dekker, Inc., New York.

Moreno, L., L. Acosta and J. L. Sánchez (1992). Design of algorithms for spacial-time reduction complexity of dynamic programming. *IEE Proc.-D*, **139**, 172-180.

Moreno, L., L. Acosta, A. Hamilton, J. A. Méndez, J. L. Sánchez and J. D. Piñeiro (1994). Dynamic programming approach for nonlinear systems. *IEE Proc.-Cont. Th. Appl.*, **141**, 409-417.

Mosca, E. (1995). *Optimal, predictive and adaptive control*. Prentice Hall, New Jersey.

P. de Madrid, A., M. Santos, S. Dormido and F. Morilla (1994). Constrained generalized predictive control with dynamic programming. In: *Advances in Model-Based Predictive Control* (D. W. Clarke, Ed.), 276-290. Oxford University Press, Oxford.

P. de Madrid, A. (1995). *Aplicación de técnicas de programación dinámica a control predictivo basado en modelos*. Ph. D. Thesis. U.N.E.D., Madrid. (In Spanish)

Pierre, D. A. (1986). *Optimization theory with applications*. Dover Publications, Inc., New York.

Ruipérez, H. (1977). *Sobre ciertos aspectos algorítmicos de la programación dinámica*. Degree Dissertation. Universidad de Madrid, Facultad de Ciencias Físicas. (In Spanish)