SIMULATION-BASED OPTIMIZATION FOR NONLINEAR OPTIMAL CONTROL

Niket S. Kaisare [*] **Jong Min Lee** [*] **Jay H. Lee** [*,1]

[*] *School of Chemical Engineering, Georgia Institute of Technology, Atlanta, GA 30332, U.S.A.*

Abstract: Optimal control of systems with complex nonlinear behaviour such as steady state multiplicity results in a nonlinear optimization problem that needs to be solved online at each sample time. We present an approach based on simulation, function approximation and evolutionary improvement aimed towards simplifying online optimization. Closed loop data from a suboptimal control law, such as MPC based on successive linearization, is used to obtain an approximation of the '*cost-to-go*' function, which is subsequently improved through iterations of the Bellman equation. Using this offline-computed cost approximation, an infinite horizon problem is converted to an equivalent single stage problem — substantially reducing the computational burden. This approach is tested on continuous culture of microbes growing on a nutrient medium containing two substrates that exhibits steady state multiplicity. Extrapolation of the cost-to-go function approximator can lead to deterioration of online performance. Some remedies to prevent such problems caused by extrapolation are proposed.

Keywords: Optimal Control, Continuous Bioreactor, Multiple Steady States, Neuro-Dynamic Programming, Cybernetic Modeling, *Klebsiella oxytoca*

1. INTRODUCTION

We are often faced with the task of solving nonlinear dynamic optimization problems — either off-line or more commonly on-line. An example is the popular method of Model Predictive Control (MPC) (Morari and Lee, 1999; Mayne *et al.*, 2000), which requires a nonlinear dynamic optimization problem to be solved at each sample time when applied on a nonlinear process model. Nonlinear dynamic optimization problems are intrinsically hard problems and it is difficult to assure the attainment of quick, reliable solutions, which are needed in most of the practical applications. Difficulties exist even in off-line optimization, when the problem involves a model of high dimension and a large time window, thus yielding a large set of optimization variables and constraints. In practice, these problems are often solved in a highly approximate sense (by using a linear approximation of the model, for example) or are avoided by adopting heuristic policies instead.

One approach for solving dynamic optimization is Dynamic Programming (DP). Here, the aim is to find the optimal *'cost-to-go'* function, which can be used to parameterize the optimal solution with respect to the system state – either as a continuous function or as a lookup table – thereby simplifying the task of obtaining on-line solutions. However, the approach is largely considered impractical as analytical solution of resulting dynamic program is seldom possible and numerical solution suffers from the 'curse of dimensionality' (Bellman, 1957).

Neuro-Dynamic Programming (NDP) approach was proposed as a way to alleviate the curse of dimensionality (Bertsekas and Tsitsiklis, 1996). It uses simulated process data obtained under suboptimal policies to fit an approximate cost-to-go function – usually by fitting artificial neural networks, hence the name. The initial approximate cost-to-go function is further improved by an iteration procedure based on the so called Bellman equation. In this context, the simulation's role is two-fold. First, by simulating the process under a reasonably chosen suboptimal policy and all possible operating parameters / disturbances, it provides a set of data points that define the relevant region in the state space. Second, the simulation provides the cost-to-go value under the suboptimal policy for each state visited, with which iteration of the Bellman equation can be initiated.

[1] The author to whom correspondence should be addressed. Email: jay.lee@che.gatech.edu

The NDP approach has received significant attention for its successes in several applications such as elevator dispatch problem and a program that plays Backgammon at the world championship level. Recently, we used NDP to develop a nonlinear model predictive controller for a benchmark *Van de Vusse reaction* system (Lee and Lee, 2001). In this paper, this method is applied to a more complex problem involving a continuous bioreactor displaying multiple steady states. The cybernetic modeling framework developed by Ramkrishna and coworkers (Kompala *et al.*, 1986) is used to model the system. For Nonlinear MPC (NMPC) method based on successive linearization of the nonlinear model (Lee and Ricker, 1994) to this system, we needed long prediction and control horizons due to certain peculiar dynamics of the bioreactor, *viz* quickly settling to an almost stable behavior that lasts for a long period followed by a sharp drift to another steady state, which is "triggered" by a change in the cells' metabolic states [2]. We seek to use this approach, not only to reduce the on-line computational demand but also to improve the performance of the suboptimal MPC method.

2. PRELIMINARIES

2.1 *Open Loop Optimal Control Problem*

A general dynamic optimization problem commonly found in the optimal control literature is as follows:

$$\min_{u_0,\cdots,u_{p-1}} \sum_{i=0}^{p-1} \phi(x_i,u_i) + \bar{\phi}(x_p) \qquad (1)$$

with

$$\begin{aligned} \text{Path Constraint: } & g_i(x_i,u_i) \geq 0, 0 \leq i \leq p-1 \\ \text{Terminal Constraint: } & \bar{g}(x_p) \geq 0 \\ \text{Model Constraint: } & \dot{x} = f(x,u) \end{aligned}$$

for a given initial state x_0 and a piecewise constant input $u(\tau) = u_i \; i\cdot h \leq \tau < (i+1)\cdot h$. ϕ is the single stage cost function and $\bar{\phi}$ is the terminal state cost function. Such a problem may be solved in the context of finding an open-loop input trajectory off-line for a fixed finite-time process (*e.g.*, a batch process). For a continuous system, the problem may be solved on-line at each sample time in order to find the optimal input adjustment for the given state – as in 'Receding Horizon Control' (Morari and Lee, 1999; Mayne *et al.*, 2000). In the latter case, it is shown to be advantageous to solve an infinite horizon problem (in which p is set to infinity). Obviously, the computational load for solving the optimization increases with the size of the horizon, thereby limiting the use of such an approach in practice, especially when nonlinear system models are involved and the use of a large optimization window is required to obtain a satisfactory result.

[2] The microbes display different nutrient uptake patterns in the two steady states, indicating distinct internal cellular mechanisms

2.2 *Dynamic Programming and Bellman Equation*

Dynamic Programming (DP) is an elegant way to solve the previously introduced optimization problems. It involves stagewise calculation of the *cost-to-go* function to arrive at the solution, not just for a specific x_0 but for general x_0. For (1), the cost-to-go at each stage is defined as

$$J_i = \min_{u_{p-i},\cdots,u_{p-1}} \sum_{j=p-i}^{p-1} \phi(x_j,u_j) + \bar{\phi}(x_p) \quad (2)$$

Then, the calculation of the cost-to-go function at each stage can be done recursively as

$$J_i(x) = \min_u \phi(x,u) + J_{i-1}(F_h(x,u)), \quad (3)$$

where $F_h(x,u)$ denotes the resulting state after integrating the differential equation for one sample interval with the starting state of x and constant input of u. The above is sequentially solved from $i = 1$ through $i = p$ with the initialization of $J_0 = \bar{\phi}(x)$. Of course, the pertinent terminal / path constraints need to be imposed at each stage. The cost-to-go function, once found, represents a convenient vehicle to obtain the optimal solution for a general state x_0.

By continuing the cost-to-go iteration of (3) until convergence within the above procedure, we can see that the infinite horizon cost-to-go function J_∞ satisfying the following *'Bellman Equation'* can be obtained.

$$J_\infty(x) = \min_u \{\phi(x,u) + J_\infty(F_h(x,u))\} \quad (4)$$

In very few cases can we solve the stagewise optimization analytically to obtain a closed-form expression for the cost-to-go function. The conventional numerical approach to the problem involves gridding the state space, calculating and storing the cost-to-go for each grid point as one marches backward from the last stage to the first. For an infinite horizon problem, the number of iterations required for convergence can be very large. Such an approach is seldom practically feasible due to the exponential growth of the computation with respect to state dimension. This is referred to as the 'curse of dimensionality', which must be removed in order for this approach to find a widespread use.

2.3 *A Simulation Based Alternative to Obtain Cost-to-go Approximation*

The *policy improvement theorem* states that a new policy that is greedy [3] with respect to the cost-to-go function of the original policy is as good as or better than the original policy; i.e. the new policy defined by $J^{i+1} = \min_u(\phi + J^i)$ is an improvement over the original policy (Sutton and Bartow, 1998). Indeed, when the new policy is as good as the original policy, the above equation becomes the same as Bellman optimality equation (4). Use of the Bellman equation to obtain iterative improvement of cost-to-go approximator forms the crux of various methods like Neuro-Dynamic Programming (NDP) (Bertsekas and Tsitsiklis, 1996), Reinforcement Learning (RL) (Sutton and Bartow, 1998), Temporal Difference (Tsitsiklis and Roy, 1997) and such.

In this paper, the basic idea from NDP and RL literature is used to improve the performance of a successive linearization based Nonlinear Model Predictive Control (NMPC) method applied to a bioreactor. Relevant regions of the state space are identified through *simulations* of the NMPC control law, and initial suboptimal cost-to-go function is calculated from the simulation data. A *functional approximator* is used to interpolate between these data points. *Evolutionary improvement* is obtained through iterations of the Bellman equation (5). When the iterations converge, this offline-computed cost-to-go approximation is then used for online optimal control calculation for the reactor.

In the remainder of the paper, we refer to our proposed algorithm as *simulation-approximation-evolution* (S-A-E in short) scheme as this term captures the essence of the algorithm much better than NDP or RL.

2.4 *The Algorithm*

The simulation-approximation-evolution scheme involves computation of the converged cost-to-go approximation offline. The following steps describe the general procedure for the infinite horizon cost-to-go approximation.

(1) Perform simulations of the process with chosen suboptimal policies under all representative operating conditions.
(2) Using the simulation data, calculate the ∞-horizon cost-to-go for each state visited during the simulation. For example, each closed loop simulation yields us data $x(0), x(1), \ldots, x(N)$, where N is sufficiently large for the system to reach equilibrium. For each of these points, one-stage cost $\phi(k)$ is computed. Cost-to-go is the sum of single stage costs from the next point to the end of horizon — $J(k) = \sum_{i=k+1}^{N} \phi(i)$.
(3) Fit a neural network to the data to approximate cost-to-go function as a smooth function of the states.
(4) With the cost-to-go function approximation $\tilde{J}^i(x)$, calculate $J^{i+1}(x)$ for the given sample points of x by solving
$$J^{i+1} = \min_u \phi(x,u) + \tilde{J}^i(F_h(x,u)) \quad (5)$$
which is based on the Bellman Equation.
(5) Repeat steps 3 and 4 until convergence of the cost-to-go function approximation. This procedure is known as *cost iteration*.
(6) **Policy Update** may sometimes be necessary to increase the coverage of the state space. In this case, more suboptimal simulations with the updated policy ($\tilde{J}^i$) are used to increase the coverage or the number of data points in certain region of state space.

Assuming that one starts with a fairly good approximation of the cost-to-go (which would result from using a good suboptimal policy), the cost iteration should converge fairly fast — faster than the conventional stagewise cost-to-go calculation.

3. MICROBIAL CELL REACTOR WITH MULTIPLE STEADY STATES

Continuous bioreactors often display steady state multiplicity, and significant delayed responses to changes in the environment due to the tendency of living cells to switch metabolic states in response to environmental pressures. Steady state multiplicity is a condition in which a system displays two or more distinct states and output conditions for the same set of input conditions. Product formation is associated with specific metabolic states that can be achieved only by carefully controlling the cells' environment. Analysis of models of microbial cell cultures indicate coexistence of multiple stable steady states in a certain range of operation (Namjoshi and Ramkrishna, 2001). This work focuses on the control of a continuous stirred tank containing bacterium *Klebsiella oxytoca* growing on a mixture of two sugars: glucose and arabinose. Specifically, switching between multiple steady states to drive the reactor to the preferred steady state is considered.

3.1 *Description of the Modeling Scheme*

This system was originally studied for a batch reactor by Kompala *et al.* (1986). They applied the cybernetic modeling framework for modeling

[3] A greedy policy is one whose current cost is the least.

the diauxic behavior of the system. The five states of the system correspond to substrate concentrations, biomass concentration and concentration of the two key enzymes within the cells. The model consists of five ODEs:

$$\frac{ds_i}{dt} = D[s_{if} - s_i] - Y_i[r_i v_i]c \qquad (6)$$

$$\frac{de_i}{dt} = r_{ei}u_i + r^*_{ei} - \beta_i e_i - r_g e_i \qquad (7)$$

$$\frac{dc}{dt} = r_g c - Dc \qquad (8)$$

where $i = 1, 2$ and the rates are given by

$$r_i = r_i^{max} \frac{s_i}{K_i + s_i} \left(\frac{e_i}{e_i^{max}} \right)$$

$$r_{ei} = \alpha_i \frac{s_i}{K_{ei} + s_i}$$

$$r_g = r_1 v_1 + r_2 v_2$$

As seen in equation (6), the monod type kinetics are modified by cybernetic regulation variables of the second type $v_i = \frac{r_i}{\max(r_1, r_2)}$, that modify enzyme activity. Similarly, the cybernetic regulation variables $u_i = \frac{r_i}{r_1 + r_2}$ in equation (7) modify the rates of enzyme synthesis.

State Variables	s_1	glucose (gm/L)
	s_2	arabinose (gm/L)
	e_1	key enzyme(s)-1 (gm/gm dry wt.)
	e_2	key enzyme(s)-2 (gm/gm dry wt.)
	c	biomass (gm/L)
M. V.	D	dilution rate (hr^{-1})
C. V.	c	biomass
Parameter	s_{2f}	s_2 feed rate (gm/L)

Table 1. Key variables and parameters of the system

Numerical bifurcation analysis of the above-mentioned cybernetic model of bacterial growth on substitutable substrates revealed the existence of two stable steady states in a certain range of operating parameters (Namjoshi and Ramkrishna, 2001), which arise due to cells' ability to switch their physiological states under nutritional pressures. Figure 1 shows the steady state bifurcation diagram for a bacterium *Klebsiella Oxytoca* growing on mixed feed of glucose and arabinose, in a continuous stirred reactor at dilution rate of $0.8 hour^{-1}$. The steady state which results in high biomass yield is the desired state. Values of the state variables for the two different steady states are shown in Table 2. One can observe that the "working steady state" is close to turning point bifurcation. Thus, relatively small changes in dilution rate and/or substrate feed concentrations could cause the reactor to drift to the other steady state.

The control objective is, therefore, to drive the reactor from the low biomass steady state to the desirable high biomass yield state. It may be viewed as a step change in the setpoint at time $t = 0$ from the low biomass to the high biomass yield steady state. The performance of the controller is evaluated under step disturbances of various magnitude in parameter s_{2f}.

State	s_1	s_2	e_1	e_2	c
High	0.035	0.081	0.0004	0.0006	0.0565
Low	0.0447	0.1425	0.0007	0.0003	0.02

Table 2. Steady state values for: $D = 0.8$, $s_{1f} = 0.078$, $s_{2f} = 0.146$. High and low represent biomass yield.

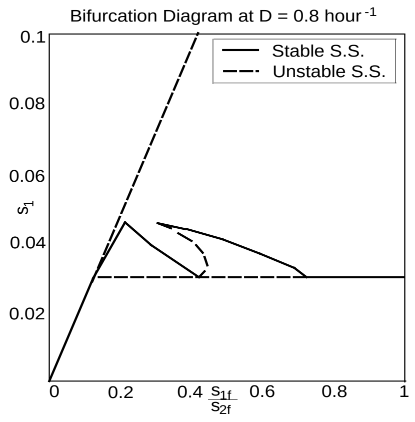

Fig. 1. Steady state bifurcation diagram for Klebsiella Oxytoca growing on glucose and arabinose. Note the proximity of the steady states to turning point bifurcation. Adapted from Namjoshi and Ramkrishna (2001)

4. IMPLEMENTATION AND DISCUSSIONS

4.1 *Suboptimal control law: Nonlinear MPC*

The successive linearization based NMPC algorithm (Lee and Ricker, 1994) was used as the initial suboptimal control law. This method linearizes the nonlinear model at each current state and input values to compute a linear prediction equation. The control is computed by solving a QP, with the hessian and the gradient computed from the new linear approximation. Readers are referred to the paper by Lee and Ricker (1994) for further details.

The closed loop simulation with the NMPC algorithm under a parameter conditions $s_{2f} = 0.146$ and $c_{SP} = 0.055$ is shown in Figures 2, 4 as thick line. At time $t = 0$, the system was at a low biomass steady state, when a step change in the set point to a high biomass steady state was applied. The constraints on the dilution rate were chosen to be $u_{\min} = 0.6$, $u_{\max} = 1.0$ and $\Delta u_{\max} = 0.05$, keeping in mind that the model is not valid for low dilution rates and to avoid washout condition that occurs at high dilution.

4.2 *Obtaining optimal cost approximator*

4.2.1. *Simulations using suboptimal controller* The suboptimal NMPC controller described above, was used to obtain closed loop simulation data for the proposed strategy. It was implemented for four values of $s_{2f} = [0.14\ 0.145\ 0.15\ 0.155]$, to cover the possible range of variations. For each of the parameter values, the reactor was started at three different $x(0)$ values around the low biomass yield steady state. We obtained 100 data points for each run. Thus a total of 1200 data points were obtained. The infinite horizon cost-to-go values were computed for all the 1200 points. Note that the calculated cost-to-go value is approximate infinite horizon cost, as described in section 2.4.

4.2.2. *Cost approximation* States were augmented with the parameter s_{2f} (see table 1). A functional approximation relating cost-to-go with augmented state was obtained by using a neural network — multi layer perceptron with five hidden nodes, six input and one output node. The neural network showed a good fit with mean square error of 10^{-3} after training for 1000 epochs. This is the zeroth iteration, denoted as $\tilde{J}^0(x)$.

4.2.3. *Improvement through Bellman iterations* Improvement to the cost-to-go function is obtained through iterations of the Bellman equation (5). This method, known as cost iteration, is described in section 2.4. The solution of the one-stage-ahead cost plus cost-to-go problem, results in improvements in the cost values. The improved costs were again fitted to a neural network, as described above, to obtain subsequent iterations $\tilde{J}^1(x)$, $\tilde{J}^2(x)$, and so on ..., until convergence. Cost was said to be "converged" if the sum of absolute error was less than 5%. The cost converged in 4 iterations for our system.

4.3 *Online implementation*

The converged cost-to-go function from above was used online in solving the one stage ahead problem. The control move was calculated as in 9 and implemented online in a receding horizon manner.

$$u(k) = \arg\min_{u(k)} \{\phi\,(x(k), u(k)) + \tilde{J}^4\,(f_h(x(k), u(k)))\Big\} \qquad (9)$$

The results are shown as broken line in Figure 2 and a numerical comparison is shown in Table 3. Clearly, the new scheme is less optimal than the original NMPC scheme. An overshoot is observed and the total cost is also increased. However, there is a dramatic reduction in computational time — from almost half an hour to under 2 minutes, for 100 time steps (50 hours). In the next section, we evaluate the possible reasons for the worse behavior and discuss possible solutions.

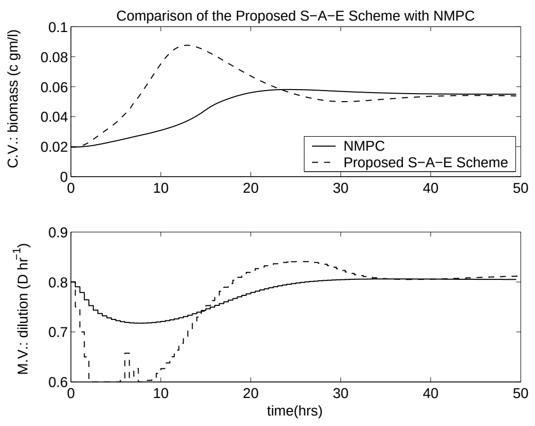

Fig. 2. Comparison of the online performance of NMPC control law and proposed S-A-E approach. Also see Table 3.

4.4 *Improvement in the Strategy*

The policy improvement theorem, described earlier, indicates that Bellman iteration is expected to improve the performance over the suboptimal controller. At worst, the performance of the proposed scheme should be at par with the original suboptimal scheme. The logical reasoning behind this is that Bellman iterations should choose the original policy over all other policies that lead to a less optimal result (Sutton and Bartow, 1998).

The possible causes of error could either be presence of local minima, poor fitting of the cost approximations (by the neural network), or extrapolation to previously unvisited regions in state space. An investigation of the state space plot in Figure 3 suggests that extrapolation to previously unvisited regions of the state space could have result in deterioration of the controller performance.

Possible remedies are discussed in brief; they will be discussed in full length paper [4].

- *Gridding and restricting the working region*: The optimizer was *restricted* to search only in the visited region of the state space during both offline Bellman iterations as well as online implementation.
- *Increasing data coverage through additional simulations*: Additional data is obtained from some more simulations of NMPC law and cost iteration is performed again. Points

[4] www.prism.gatech.edu/~gte207x/files/ifac02full.pdf

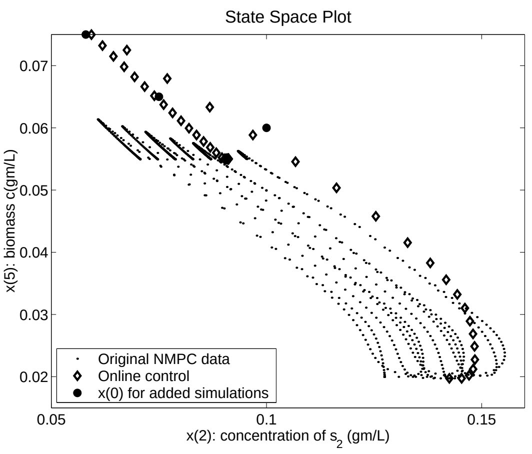

Fig. 3. State space plot of states visited during online implementation (diamond) and training date from NMPC controller (dots). Extrapolation to the unvisited states is likely to be the cause of overshoot.

shown by solid discs in Figure 3 are used as $x(0)$ for additional simulations.

- *Generalized Policy Update*: This technique is used to increase the coverage of the state space by performing generalized policy update within the cost iteration loop. In other words, data points that lie in unvisited region of the state space are added to the original data by performing control simulations using the suboptimal cost-to-go approximation $\tilde{J}^i$.

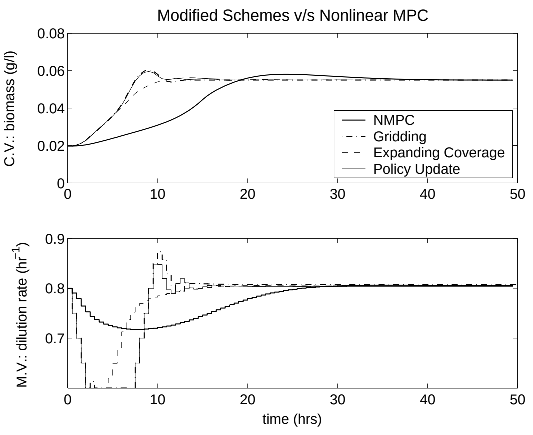

Fig. 4. Performance of the modified schemes. NMPC control law is shown as thick solid line for comparison

Control Algorithm	Total cost (at $x(0)$)	CPU Time (seconds)‡
NMPC	22.54	1080.3
S-A-E	24.18	98.7
w/ Gridding	9.06	127.7
w/ Add Sim†	9.37	79.5
w/ Pol. update	10.32	74.12

Table 3. Comparison of NMPC algorithm v/s S-A-E and its modifications. †Additional NMPC simulations; ‡Intel Pentium III, 800 MHz processor

5. CONCLUSIONS

Application of simulation based strategy for improving the performance of MPC for steady state switching in a microbial reactor provides a promising framework for nonlinear optimal control in a computationally amenable way. Presence of local minima, over-fitting of neural network or extrapolation to previously unvisited regions in state space may result in deteriorated performance. In this study, latter was found to result in poor controller performance. Three different modifications were suggested to obtain optimal performance. As a final comment, we suggest the use of *generalized policy update* over other two methods.

6. ACKNOWLEDGEMENTS

We thank Dr. Ramkrishna and A. A. Namjoshi for their help with cybernetic modeling and NSF (CTS-0096326) for financial support.

7. REFERENCES

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

Bertsekas, D. P. and J. N. Tsitsiklis (1996). *Neuro-Dynamic Programming.* Athena Scientific. Belmont, MA.

Kompala, D. S., D. Ramkrishna, N. B. Jansen and G. T. Tsao (1986). Investigation of bacterial growth on mixed substrates: Experimental evaluation of cybernetic models. *Biotechnology and Bioengineering* **28**, 1044–1055.

Lee, J. H. and N. L. Ricker (1994). Extended kalman filter based nonlinear model predictive control. *Ind. Eng. Chem. Res.* **33**, 1530–1541.

Lee, J. M. and J. H. Lee (2001). Neuro-dynamic programming method for mpc. In: *DYCOPS VI.* pp. 157–162.

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

Morari, M. and J. H. Lee (1999). Model predictive control: past, present and future. *Computers and Chemical Engineering* **23**, 667–682.

Namjoshi, A. A. and D. Ramkrishna (2001). Multiplicity and stability of steady states in continuous bioreactors: Dissection of cybernetic models. *Chemical Engineering Science* **56**(19), 5593–5607.

Sutton, R. S. and A. G. Bartow (1998). *Reinforcement learning: an introduction.* MIT Press. Cambridge Massachussets.

Tsitsiklis, J. N. and B. Van Roy (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control* **42**, 674–690.