automation

MDPI

Article

Optimal Control Implementation with Terminal Penalty Using Metaheuristic Algorithms

Viorel Minzu

Control and Electrical Engineering Department, "Dunarea de Jos" University of Galati, 800008 Galati, Romania; viorel.minzu@ugal.ro

Received: 14 September 2020; Accepted: 10 October 2020; Published: 15 October 2020

check for updates

Abstract: Optimal control problems can be solved by a metaheuristic based algorithm (MbA) that yields an open-loop solution. The receding horizon control mechanism can integrate an MbA to produce a closed-loop solution. When the performance index includes a term depending on the final state (terminal penalty), the prediction's time possibly surpasses a sampling period. This paper aims to avoid predicting the terminal penalty. The sequence of the best solution's state variables becomes a reference trajectory; this one is used by a tracking structure that includes the real process, a process model (PM) and a tracking controller (TC). The reference trajectory must be followed up as much as possible by the real trajectory. The TC makes a one-step-ahead prediction and calculates the control inputs through a minimization procedure. Therefore the terminal penalty's calculation is avoided. An example of a tracking structure is presented. The TC may also use an MbA for its minimization procedure. The implementation is presented in two versions: using a simulated annealing algorithm and an evolutionary algorithm. The simulations have proved that the proposed approach is realistic. The tracking structure does or does not work well, depending on the PM's accuracy in reproducing the real process.

Keywords: optimal control problem; metaheuristic algorithm; tracking controller; simulated annealing; evolutionary algorithm; process model

1. Introduction

In process engineering, many applications involve the optimal control of a dynamic system. Sometimes the structural properties of a dynamic system and the nature of the optimal control problem lead to theoretical control laws that are relatively easy to implement without significant computational complexity. On the other hand, there are situations in which the optimal evolution of the system requires important computational effort within a time interval. That is why metaheuristic-based algorithms (MbAs) (see [1–3]) have been used for over two decades in control engineering (see [4–6]).

The closed-loop control structure able to integrate an MbA is the receding horizon control (RHC) mechanism (see [7–9]). The controller of this structure makes optimal predictions using a process model (PM). Model predictive control is a particular case of RHC (see [10]). The integration of MbA and RHC are systematically described in [7,9,11]. The prediction horizon depends on the nature of the optimal control problem (OCP)'s performance index. A big computational complexity occurs when the performance index includes a Mayer-type term that measures the quality of the trajectory in its final extremity. This term is usually called terminal penalty. In this case, the final time of prediction and control horizons are identical. Unfortunately, in the first sampling periods, the prediction horizon covers all or near all control horizon. Therefore, the prediction's calculation takes a long time that possibly does not enter inside a sampling period.

This paper proposes an alternative control structure to avoid the big computational complexity of predicting the terminal penalty. The new approach also aims to give a closed-loop solution to the OPC

in question. We suppose that an MbA has already been developed for solving this OCP. An execution series will produce the best solution for the addressed problem. The sequence of the best solution state variables is called the reference trajectory in our approach.

The control structure proposed in this paper also includes a model of the real process (PM) and has, as data input, the reference trajectory produced by the MbA. The closed-loop structure has the following main objective: the reference trajectory must be followed up as much as possible by the sequence of its state variables (the real trajectory). That is why we may consider the closed-loop as a tracking structure.

Section 3 states the tracking problem as the new problem that has to be solved by the tracking structure. Neither the OCP in question nor the specific MbA is involved in this statement. The reference trajectory contains intrinsically both. The new control structure has a tracking controller that aims to approach the reference and real trajectories. The tracking controller will use only one-step-ahead predictions to minimize the distance between trajectories as much as possible. Hence, the computational complexity will be much diminished.

For its minimization task, the tracking controller may use, in turn, a metaheuristic based algorithm, denoted by MA in the sequel. Let us note that the MA is different from MbA. Section 4 gives an example of how to solve a benchmark OCP using a tracking structure. The tracking controller's implementation is presented in two versions: the first includes a simulated annealing algorithm and the second an evolutionary algorithm.

The tests have proved that the tracking structure is a pragmatic approach to solve the OCP in question when we already have a reference trajectory (eventually yielded by an MbA). The price to pay is the fact that the performance index is approximated with an error of a few percent. The key factor is the accuracy of the PM in reproducing the real process. Before a real-time implementation with a real process, the tracking structure's designer can simulate it and analyze the degradation of the process' quasi-optimal behavior.

2. Closed-Loop Structure for Optimal Control

2.1. Optimal Control Problem: Hypotheses

Let us consider an OCP regarding an invariant nonlinear dynamic system with differential and algebraic equations as a PM:

$$\dot{X} = f(X(t), U(t)) \tag{1}$$

$$X = [x_1, \cdots, x_n]^T; U = [u_1, \cdots, u_m]^T; X(t) \in R^n; U(t) \in R^m$$

The control horizon is $[t_0, t_N]$, with discrete moments $t_i = t_0 + i{\cdot}T$, $I = 0, \ldots, N$, where T is the sampling period, and t_0 is the initial moment ($t_0 = 0$). If the initial value $X_0 = X(t_0)$ and the sequence of control inputs

$$U_0 = U(t_0), U_1 = U(t_1), \ldots, U_{N-1} = U(t_{N-1})$$

are known, then the sequence of the state variables

$$X_1, \ldots, X_{N-1}, X_N,$$

can be calculated using the PM.

Solving an OCP implies calculation of the control sequence that maximizes or minimizes the performance index $J(t_0, t_N, X(t_0), U)$ on the control horizon, starting from $X(t_0)$.

Besides the initial conditions, there are also a certain number of algebraic and differential constraints imposed to control inputs and state variables.

In this work, we have adopted some working hypotheses that cover a lot of the processes involved in many OCPs.

Working hypotheses:

- The field $f = [f_1, \cdots, f_n]^T$ has all the smoothness properties (continuity and differentiability) needed by the sequel's calculation.
- The dynamic system described by (1) corresponds to slow nonlinear systems, such as chemical batch processes. The controller needs a sampling time, T, large enough to calculate the current optimal control input (denoted $U(k)$ in the sequel).
- For this kind of process, the initial state $X(t_0)$ can be known. When masses, volumes, and concentrations compose the initial state, this is a realistic hypothesis because it can be regenerated.
- The constraints refer only to the control inputs: $U(t) \in \Omega \subset R^m$, where Ω is an open set.

2.2. Closed-Loop Control Structure

In the majority of papers that address an OCP and use a specific MbA, the solution is limited to the calculation of control input values that optimize the performance index using the concerned metaheuristic. Emphasis is placed on the metaheuristic structure and its parameter tuning that involves good effectiveness. For a closed-loop control structure with a real process (not a process model) the sequence of optimal control inputs mentioned before is useless. The closed-loop implementation is a connected problem, important, and not easy to treat.

The receding horizon control (RHC) structure is a well-known method to achieve a closed-loop control structure that can be a solution for this kind of problem. We have already proposed a systematic design procedure in [7] that uses RHC as a closed-loop structure. This one includes the MbA adapted to play the role of the controller.

Because the theoretical framework of the RHC is already well known, we recall hereafter only the elements that define the RHC strategy:

- The controller calculates the next control input value by looking ahead for a number of steps regarding a given performance index. The control input is only implemented by one step.
- The controller predicts over the number of steps taken into account using a dynamic process model (PM).
- The implementation result is checked, and a new decision is made by taking updated information into account and looking ahead for the same number of steps.
- The prediction horizon "recedes" at each sampling period but keeps the final extremity (we considered here the most difficult situation when the objective function has a terminal penalty term). Hence, its length decreases by one unit at each sampling period.

The receding horizon controller is, roughly speaking, the MbA, slightly modified to be integrated into the closed-loop. It makes a prediction of the evolution of the process over the so-called prediction horizon.

2.3. Correspondence between Performance Index and Prediction Horizon

Usually, the performance index can be expressed for the sake of simplicity by its continuous general form as

$$J = \min_u I;\ I = \int_{t_0}^{t_f} L(x(t), u(t), t)dt + M(t_f, x_f).$$

The first part is a Lagrange-type term that measures the quality along the trajectory of the dynamic system. The second part is a Mayer-type term that measures the quality of the trajectory in its final extremity. The Mayer-type term will be called terminal penalty in the sequel. The structure of the performance index is decisive for the strategy of RHC related to the prediction horizon.

The prediction horizon can have different positions inside the control horizon $[0, H]$ (see [11]). Figure 1 shows the situation when the prediction horizon includes the final moment H of the control

horizon. Accordingly, the prediction horizon's length is variable, having the value $h = H - k$, where k is the current sampling time. Because k evolves from 0 to $H-1$, it holds

$$h = H, \ldots, 1.$$

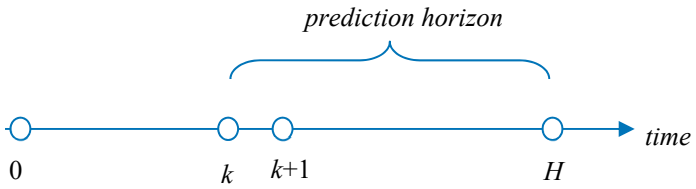


Figure 1. The prediction and control horizons with terminal penalty.

Generally, this scheme is compulsory when the performance index includes a terminal penalty because this must be calculated, and consequently, the prediction horizon must include the final states. It meets the elements that define the receding horizon control (RHC) strategy.

The prediction horizon "recedes" at each sampling period but keeps the final extremity. Hence, its length decreases by one unit at each sampling period. Unfortunately, in the first sampling periods, the prediction horizon covers all or near all control horizon. Therefore, the prediction's calculation takes a long time that possibly does not enter inside a sampling period. A favorable situation would be when there would be an estimation of the terminal penalty. However, generally speaking, there is no such estimation for each process. That is why we propose another approach in this paper, which avoids calculating the terminal penalty.

3. Tracking of the Quasi-Optimal Trajectories

This paper proposes a method to implement a closed-loop control structure devoted to the situation in which an MbA has already been developed, aiming to solve an optimal control problem having a terminal penalty. For example, the OCP described in Section 4 has been solved using an evolutionary algorithm, denoted EA1. The later one, which is a stochastic algorithm, yields a single solution after a single execution. Therefore, it generates a single realization of a stochastic process. The designer of EA1 has made the appropriate choices to ensure the convergence of the stochastic process. A solution yielded by EA1 is a sequence of control inputs that optimizes the performance index and allows simulation of the open-loop system's evolution over the control horizon. Practically, one must carry out EA1 many times (e.g., 30–40) to obtain quasi-optimal solutions. After a simulation series, the best quasi-optimal solution is available. Four elements can express this one:

- the initial state: X_0
- the sequence of quasi-optimal control inputs: $U_0^*, U_1^*, \cdots, U_{N-1}^*$
- the sequence of quasi-optimal state variables:

$$X_0, X_1^*, \cdots, X_{N-1}^*, X_N^* \tag{2}$$

- the value of the objective function: J^*

The simulated state evolution having the best performance index will be called in the sequel reference trajectory.

Remark 1:

- The solution found by EA1 is an open-loop solution. Its control input sequence is useless for a control structure that includes a real process. It cannot be used directly by a control structure. Generally speaking, the MbA has to be slightly modified and integrated into the controller of an eventual closed-loop (see [7,9,11]).

- If the OCP under consideration has a terminal penalty, there is an additional difficulty to integrate the MbA into the controller: the prediction horizon must include the final time for each sampling period. This fact involves a big computational complexity. The prediction's calculation takes a long time that possibly does not enter inside a sampling period. Therefore, the control structure could not be implemented.
- This paper proposes a new method to achieve the closed-loop control structure, and also the calculation of the terminal penalty is avoided.

Figure 2 presents the proposed control structure, which includes a tracking controller appellation that will be justified in the sequel. The reference trajectory is used by the tracking controller to yield the control inputs for the real process. These control inputs will determine a quasi-optimal behavior of the process (in closed-loop) if the tracking controller will reproduce the reference trajectory to a certain extent.

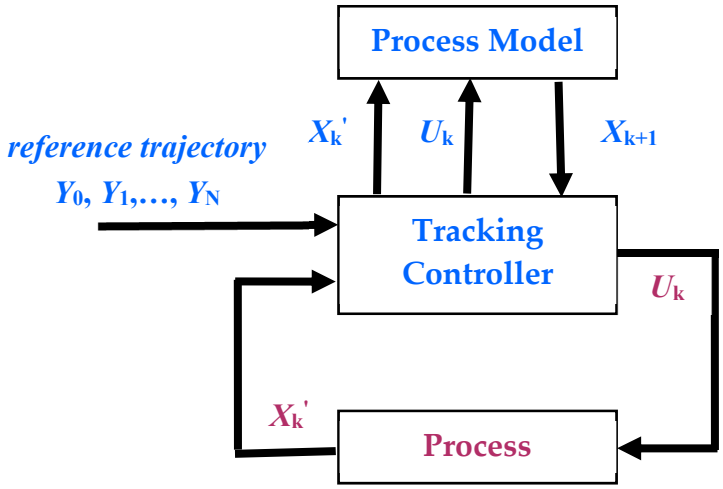


Figure 2. Quasi-optimal control structure.

This time the quasi-optimal controller does not incorporate the MbA itself, as in other closed-loop solutions. It uses only the state trajectory (2) of its best evolution (in open-loop), over a specific control horizon starting from a particular initial state.

The performance index $J(t_0, t_N, X(t_N))$ is a Mayer-type term called the terminal penalty in the sequel (e.g., $J = x_1(t_N) \cdot x_5(t_N)$). The lack of a Lagrange-type term is not aiming to reduce the difficulty of the treated OCP. For a closed-loop control structure, which uses an MbA, the terminal penalty is more difficult to treat, because it needs an estimation of the final state, which is, generally speaking, unavailable (see [7]). On the other hand, the solution proposed in this paper is devoted to this kind of OCP, having only a terminal penalty. That is the case, for example, of chemical batch processes.

3.1. Tracking Problem

In this section, we propose a statement for a new problem that will allow us to show what it does mean to reproduce the state trajectory (2) to a certain extent. Let us note that, because of the terminal penalty, our desideratum concerns only the final state of the process. It is suitable that the latter one is nearly identical to X_N^*. Consequently, the OCP's performance index J will be almost equal to J^*. However, this objective is not reachable in most applications. It may need an N-step ahead prediction that is a huge, time-consuming process. In this case, the best we can do is maintain the entire state trajectory in the reference trajectory's neighborhood, as close as possible. The controller will use only one-step-ahead predictions. That is why we may call the control structure from Figure 1 a tracking structure. The input data are a reference trajectory that must be followed up as much as possible by the sequence of its state variables. This pursuit is generated by the tracking controller (TC) that acts in a

manner adapted to this purpose. The control inputs' values are optimally determined for all sampling periods, as explained below.

To avoid confusion, we will denote the reference trajectory (2) by

$$Y_0, Y_1, \ldots, Y_{N-1}, Y_N \tag{3}$$

Figure 3 shows symbolically how the tracking structure would work. The variables have the following significance:

- U_k, $0 \leq k \leq N-1$, is a constant vector $U_k \in \Omega$ representing the control inputs' values for the sampling period $[k, k+1]$.
- X_k, $1 \leq k \leq N$, is the state vector at the moment k estimated by the TC, using the known PM (the state vector at the end of evolution on the sampling period $[k-1, k]$).
- X'_k, $1 \leq k \leq N$, is the real state vector "received" from the process at the moment k. In a real implementation, X'_k is measured and/or estimated.
- X_0 is the process' initial state (X_0 is Y_0 slightly perturbed).

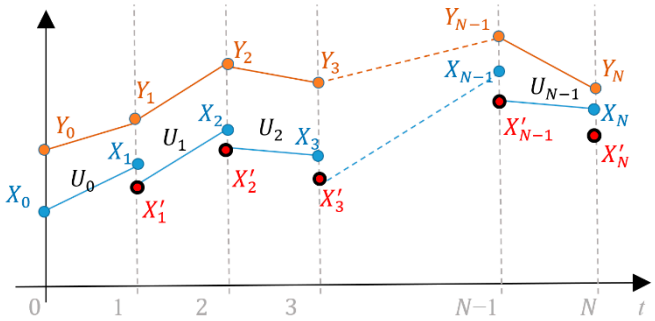


Figure 3. Tracking of the reference trajectory.

The tracking process is defined by several aspects listed below:

1. At the moment k, the TC "reads" X'_k and considers it the initial state for the current sampling period, $[k, k+1]$.
2. The TC calculates U_k and X_{k+1}. According to the PM, the control input U_k determines the following transfer:
$$X'_k \xrightarrow{U_k} X_{k+1}$$
3. The TC calculates the constant vector U_k through an optimization procedure such that
$$U_k = \underset{U \in \Omega}{\operatorname{argmin}} \|X_{k+1} - Y_{k+1}\| \tag{4}$$
4. The real process evolves according to the control input U_k and determines the following transfer:
$$X'_k \xrightarrow{U_k} X'_{k+1}$$

Aspects #2 and #3 define, actually, the one-step-ahead prediction, which the TC has to make to approach the reference trajectory.

Considering the control structure's objective, the TC has to solve the tracking problem (TP) stated as below:

$$\begin{aligned} \text{TP}: \ & \text{If the reference trajectory } (3) \text{ and the initial state } X_0 \text{ are settled,} \\ & \text{one has to } \textit{determine} \text{ the sequence of control input's values} \\ & U_0, U_1, \cdots, U_{N-1} \\ & \text{that meet the optimum criteria } (4), \text{ for } 0 \le k \le N-1. \end{aligned} \tag{5}$$

Some remarks can be made:

A. Solving the TP means that the reference trajectory was only followed up "as much as possible" using one-step-ahead prediction. Let note by d_k the minimum distance predicted at step $k-1$:

$$d_k = \min_{U \in \Omega} \|X_k - Y_k\|.$$

B. Of course, the best tracking process requires an $N-k$ step-ahead prediction because this one involves the final state X_N, which the terminal penalty depends on. In other words, the true objective would be to minimize only the distance $\|X_N - Y_N\|$ (which is not the same thing as to minimize $\|X_k - Y_k\|$, $k = 0, 1, \ldots, N-1$). However, in this way, we are coming back to a receding prediction horizon. Sometimes, this is unacceptable due to big computational complexity and a relatively small sampling period. The controller would not have enough time to make the computation.

C. On the other hand, the one-step-ahead prediction may be satisfactory from the complexity's perspective. The computation can be done inside one sampling period.

D. When the N step-ahead-prediction cannot be carried out inside one sampling period, the best we can do is to resort to the one-step-ahead prediction. This strategy allows the possibility to implement the quasi-optimal control structure satisfactorily.

3.2. Tracking Controller

A solution of the TP is the sequence of the control inputs' values (5). The TC constructs such a solution during the control horizon $[0, N]$ using the following input data:

- the PM, usually given by Equation (1);
- the reference trajectory (3);
- the initial state X_0;
- the real state vector X'_k, $1 \le k \le N$; these values are received, in real-time, from the process.

We can make some remarks concerning the tracking strategy presented before.

(I). Neither the OCP nor the MbA (used to generate the reference trajectory) are involved in the TP statement. The reference trajectory intrinsically contains both.

(II). Action #3 from Section 3.1 can be, in principle, very efficient, when $X_{k+1} = Y_{k+1}$. This particular case depends on the controllability of the system. Even if this property is satisfied, the access process can involve more sampling periods, and accordingly, the calculation would take more time. Moreover, only in particular cases, the controllability can be proved.

(III). Actions #3 and #2 will approach X_k and Y_k as much as possible while the constraint $U(t) \in \Omega$ is met. Therefore the two trajectories are near in the moment k. On the other hand, Y_{k+1} can be accessed from Y_k in only one step, using the control value U^*_k (see reference trajectory). According to the PM, it holds:

$$X'_k \xrightarrow{U_k} X_{k+1};\ Y_k \xrightarrow{U^*_k} Y_{k+1}$$

Therefore, there is an increased chance of having a small distance between the next pair of state variables, X_{k+1} and Y_{k+1}. The calculus of the input U_k, using Equation (4), transforms the tracking process into a greedy-type algorithm.

(IV). The key factor that makes the state trajectory to be close to the reference trajectory is PM's accuracy in reproducing the dynamic of the real process.

At the moment k, the process is in the state X'_k, and the TC search for the constant control input U_k that minimizes the performance index

$$d_{k+1} = \min_{U_k \in \Omega} \|X_{k+1} - Y_{k+1}\|. \tag{6}$$

In this way, we have another local optimal control problem, whose performance index is the minimization of the Euclidean distance from X_{k+1} to Y_{k+1}. Action #2 of the tracking process shows that the minimization (6) makes sense.

We now propose a particular implementation of the TC that yields an approximated solution for the tracking problem. An effective way to determine the vector $U_k \in R^m$ that minimizes the performance index (6) is to use a metaheuristic-based algorithm, denoted by MA in the sequel. Let us note that the MbA used a priori to generate the reference trajectory is different from MA.

The call of this algorithm may be represented by the following instruction:

$$[U_k, X_{k+1}, d_{k+1}] \leftarrow \text{MA}(X'_k, Y_{k+1}).$$

The MA has a fitness function that includes the numerical integration of the PM over the sampling period $[k, k+1]$. We give hereafter the outline of the TC, which is called for every single sampling period.

Outline of Tracking Controller
1. Get the current value of the state vector X'_k.
2. $[U_k, X_{k+1}, d_{k+1}] \leftarrow MA(X'_k, Y_{k+1})$
3. Send the control input U_k towards the process.
4. *Return*

In this paper, two MAs will be exemplified: the well-known simulated annealing algorithm (SAA), which is very simple to implement because it uses a single solution within its iterative searching process, and the evolutionary algorithm (EA) that works with a population of solutions.

The simulation tests have proved (see Section 4) that the tracking process, as described before, has practical relevance.

3.3. Quality of the Proposed Solution

This strategy does not guarantee that d_N would be small enough, such that the terminal penalty would have an acceptable value. The control structure designer may require that d_N is inferior to a pre-established value. Equivalently, the relative error of the performance index ε_r is less than a maximum value:

$$\varepsilon_r = 100 \cdot \frac{(J* - J)}{J^*}[\%];\ \varepsilon_r \leq \varepsilon_{\max} \tag{7}$$

Constraint (7) can be verified by simulation of the closed-loop presented in Figure 1 (at least in our research's actual theoretical stage). This simulation needs a real process model (RPM) that is different from the PM. The former is obtained by adding unmodeled dynamics and noises to the PM. The RPM construction is a difficult task in itself, which the designer has to solve before making the simulation. If the TP's solution is satisfactory under the adopted hypotheses, the control structure designer may pass to the closed-loop implementation with a real process.

Remark (I) from Section 3.2 recalls that the OCP's quasi-optimal solution results from an MbA, which has behind a convergent stochastic process. When choosing the specific MbA as an optimization tool, its convergence has already been studied.

In the presented TP, we are not faced with a convergence problem, first of all, because the TC works with two finite series of states: $X_0, X_1, \cdots, X_N$ and $X'_0, X'_1, \cdots, X'_N$. The two series are finite

because the control horizon is finite. On the other hand, the real process can perturb the tracking process significantly through its state variables. The perturbation must be realistically modeled and included in the TP statement to obtain a framework which could generate theoretical results.

4. Example of Using a Quasi-Optimal Trajectory

This section's main objective is to illustrate how a specific OCP can be solved through the intermediary of the quasi-optimal trajectory generated by an MbA. Moreover, we are not aiming to solve a specific OCP and compare our solution with other authors' approaches, but rather to exemplify the newly proposed method on a benchmark problem.

We have considered a well-known problem described in many articles among those we recall here [12]. The OCP is called Park-Ramirez bioreactor and concerns a nonlinear dynamic system. This problem is stated in the Appendix A of this article. One can see that it holds:

$$n = 5;\ m = 1;\ T = 1\text{h};\ N = 15;\ \Omega = [0, 2];$$

In the framework devoted to solving OCPs through metaheuristics, we have implemented and used an evolutionary algorithm that solves this problem. We will call it EA1 in the sequel. The EA1 has produced a quasi-optimal solution whose state trajectory is given by (A7). Some details concerning EA1 and the reference trajectory are given in Appendix B. Now, we want to implement a tracking structure using this reference trajectory. The initial state of the process in the closed-loop structure is given by (A8).

The implementation of the TC and the simulation of the closed-loop were carried out using the MATLAB language and system. To simulate the real process evolution, we have used the special functions devoted to integrating differential equations.

4.1. TC's Implementation Using Simulated Annealing Algorithm

This section presents the TC's implementation and the closed-loop simulation using the well-known simulated annealing algorithm (SAA). Let us consider SAA as a function that is symbolically called as bellow:

$$[U_k, X_{k+1}, d_{k+1}] \leftarrow SAA(X'_k, Y_{k+1}).$$

The searching process refers to U_k that has to minimize the performance index. It is naturally coded as a real vector having the components of the control input. SAA evaluates U_k calling an objective function (scripts: SAh1, eval_SA; see the folder TC_SA) that calculates the performance index (6). This function performs two actions:

- the numerical integration of the process over the interval [k, k+1] to calculate the next state X_{k+1} using the current U_k;
- the computation of the distance d_{k+1}.

Only a candidate solution meeting the constraint (8) is accepted before applying the Metropolis Rule, which will set the iterative process' current solution:

$$U_k \in \Omega \tag{8}$$

SAA has two ways to stop the solution's improvement. The first one is the convergence of the searching process, which can be declared when two conditions are met:

- the objective function had no improvement since a certain iterations number, and
- the annealing temperature is less than a minimum value.

The second corresponds to the situation when the convergence cannot be ascertained after a pre-established large number of iterations.

The first simulation series' objective was to evaluate the TC's behavior when the initial state is different from that considered in the OCP statement (see "influenceX01.m" inside the folder TC_SA). In our example, the initial state is X_{00} (see (A3)). X_0 is obtained from X_{00} to which a normally distributed noise is added:

$$X_0 = X_{00} + noise; \text{ with } noise = \text{normrnd}(\text{Mean, StDev}) \tag{9}$$

The noise is generated by a function normrnd (the same name as the MATLAB function) that generates a vector of random numbers chosen from a normal distribution with mean "Mean" and standard deviation "StDev". For the real process, we have considered an RPM identical to PM.

Table 1 presents the simulation's results for six noises having different mean values. These values are quite big compared to the variables' values x_1 and x_2, for example, on the interval $k = 0, \cdots, 13$.

Table 1. Results of the first simulation series (TC (tracking controller) with SAA, RPM (real process model) ≡ PM (process model)).

	Noise		**Relative Tracking Error [%]**				**Performance Index**		
	Mean	**StDev**	**min**	**avg**	**max**	**Rerr(*N*)**	***J****	**J**	ε_r**[%]**
1	0.02	0.01	0.41	2.01	4.24	0.41	32.28	32.89	−1.88
2	0.05	0.01	0.08	3.46	7.56	0.08	32.28	32.39	−0.34
3	0.08	0.01	0.51	5.77	12.66	0.51	32.28	31.46	2.54
4	0.1	0.01	0.76	6.73	14.52	0.76	32.28	31.05	3.81
5	0.12	0.01	1.32	8.67	18.22	1.32	32.28	30.10	6.75
6	0.15	0.01	2.01	10.64	22.04	2.01	32.28	28.97	10.25

It is important to see what the tracking's precision is. The absolute error at the moment k can be calculated using the following formula:

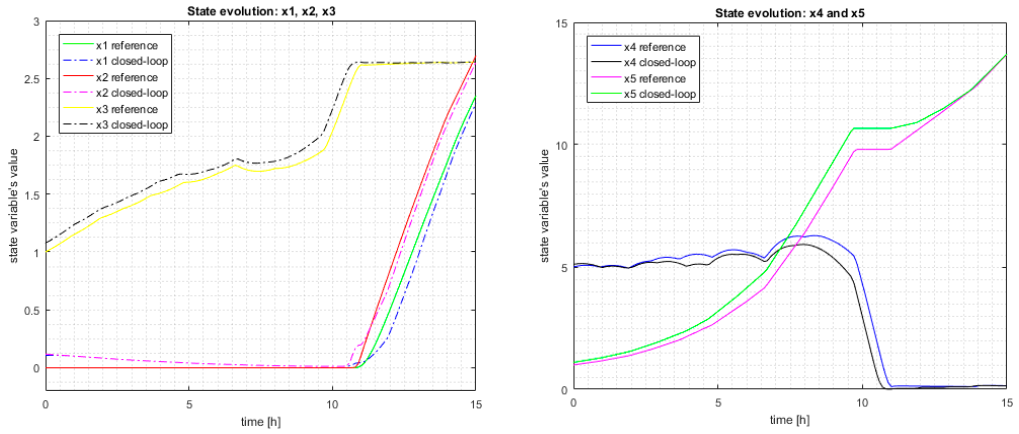
$$\text{Err(k)} = \|Y_k - X_k\|,$$

where $\|X\|$ is the Euclidean norm. The relative error at the moment k can be expressed as follows:

$$\text{Rerr(k)} = \frac{\|Y_k - X_k\|}{\|Y_k\|}. \tag{10}$$

The columns of Table 1 give the following values, respectively: the minimum, average, and maximum relative tracking error, the relative error in the final state (Rerr(N)), the reference and current performance index. In lines #1 and #2, the performance index determined by TC with SA has superior values to J^*. The explanation is related to the reference trajectory's quasi-optimal character, namely the value J^* is not the optimum.

Figure 4 presents comparatively the state evolution in the reference case and the closed-loop control system with TC using SAA. The medium value of the noise is Mean = 0.1, which is very big for our process especially for the variables x_1 and x_2. The relative error of the final state, which is more important for the terminal penalty, is only 0.76%. This fact is reflected in the relative error of the performance index, ε_r = 3.81%.



(**a**) (**b**)

Figure 4. Comparative state evolution: closed-loop versus reference trajectory (TC with SAA (simulated annealing algorithm)). (**a**) Comparative evolution of x_1, x_2 and x_3. (**b**) Comparative evolution of x_4 and x_5.

The control input's value is depicted in Figure 5 in the two cases. It is interesting to see that the control input given by the TC has relatively the same pattern. Figure 6 shows the evolution of the relative tracking error over the control horizon. Let us note that the tracking error is already near 5% initially, which is a big value.

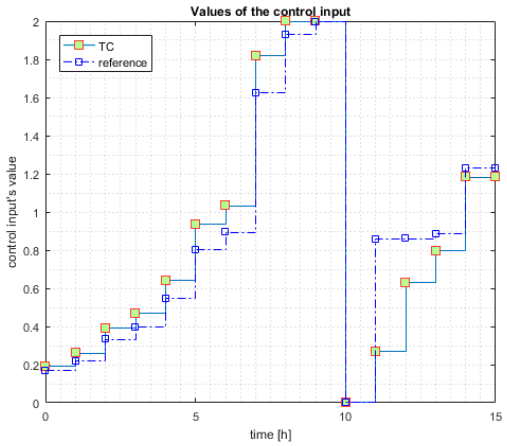


Figure 5. Evolution of the control input (TC with SAA).

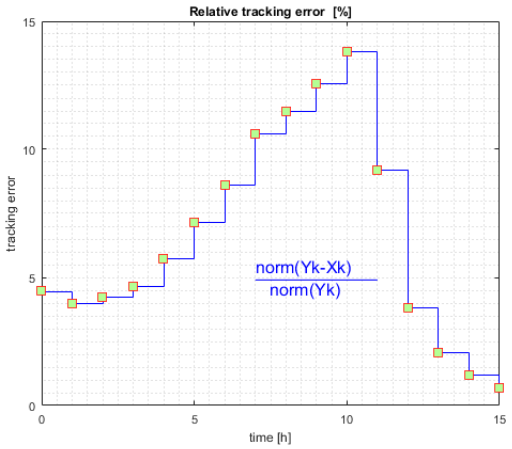


Figure 6. Evolution of the tracking error (TC with SAA).

To simulate the real process, we have considered that the RPM is equivalent to the PM to which a normally distributed noise is added. The latter one is a random variable having a normal distribution like in Equation (9). In this way, the state variables X_k and X'_k,$1 \leq k \leq N$, are different:

$$X'_k = X_k + noise. \tag{11}$$

We have implemented a similar program as before, to simulate the closed-loop functioning with TC and RPM. The only difference is that the state variables are adjusted using Equations (9) and (11) (see TC_SA_h1_MEAN0_04.m). The values characterizing the normal noise are Mean = 0.04 and StDev = 0.01. The resulting simulation program has a stochastic character; therefore, we cannot conclude after a single execution. We have to repeat the execution more times to give consistency to statistical parameters in such a situation.

That is why the second simulation series will repeat 30 times the closed-loop simulation program.

Table 2 gives some statistical parameters characterizing the 30 executions of the closed-loop simulation program. For the relative error of the performance index (ε_r, see Equation (7)), the minimum, average, maximum values, and the standard deviation are indicated. The relative tracking error in the final state, Rerr(N), is computed according to the Equation (10). Table 2 shows its minimum, average, and maximum value over the simulation series. The particular simulation among the 30, whose performance index is the closest to the average, can be considered typical execution. Its performance index is 29.96, which means the relative error has an acceptable value of 7.1%.

Table 2. Results of the second simulation series (TC with SA, RPM, Mean = 0.04, StDev = 0.01).

ε_r				**Rerr(N) [%]**			**Typical Execution**				
min	**avg**	**max**	**σ**	**min**	**avg**	**max**	$X^1{}_N$	$X^5{}_N$	J	**Rerr(N) [%]**	ε_r
0.044	0.073	0.099	0.013	2.29	2.68	3.19	2.171	13.801	29.96	2.55	0.071

Remark 1: We can make a peculiar simulation of the closed-loop, taking $X_0 = X_{00}$, and RPM ≡ PM (without noise). Obviously, the theoretical solution is trivial: the sequence of control input values calculated by the TC is just $U_0^*, U_1^*, \cdots, U_{N-1}^*$, and the sequence of the process' states is just Y_0, Y_1, ..., Y_{N-1}, Y_N. This solution is found on the condition that the MA used for local optimization (SAA in our case) works very well and finds $X_k \equiv Y_k, k = 1, \cdots, N$. Setting these conditions, the simulation (see TC_SA_X00_NONOISE.m) has proved that our SAA's implementation works very well. Generally speaking, this is a method to test the correctness of the MA's implementation.

4.2. TC's Implementation Using an Evolutionary Algorithm

The TC's implementation can use another MA. In this section, we use an evolutionary algorithm called EA2 that is different from EA1.

NB: EA1 and EA2 are two distinct implementations of the evolutionary metaheuristic. EA1 searches the optimal solution of the OCP described in Appendix A, while EA2 solves the TP presented before. There are two different elements that characterize the two OCPs:

- the performance index is (A5) for EA1, and (6) for EA2;
- the control horizon is $[0, \cdots, N]$ for EA1, and $[k, k+1]$, $0 \leq k \leq N-1$ for EA2. EA1 is executed one time to solve its problem, while EA2 is called for every sample period to solve local optimization problems.

These basic elements further involve many other implementation differences, such as solution encoding and genetic operators. Let us consider EA2 as a function that is symbolically called as below:

$$[U_k, X_{k+1}, d_{k+1}] \leftarrow EA2(X'_k, Y_{k+1}).$$

The optimization process refers to U_k that has to minimize the same performance index (6). It is naturally coded as a real vector having the components of the control input. In our case U_k is a scalar. EA2 evaluates U_k using an objective function (implemented inside the program TC_EA_h1_NOISE.m; see the folder TC_EA2) that calculates the performance index (6). This function performs two actions: the numerical integration of the process using the current U_k and the computation of the distance d_{k+1}.

The constraint (8) is implemented when the initial population is generated and any time the solution's value is modified (e.g., inside the mutation operator).

EA2 stops the solution's search after a certain number of generations (40 in our implementation) while the population evolves. This number is tuned according the preliminary tests of the program.

For our peculiar TP, every chromosome of the solution's population encodes the value of U_k. So, the length of the solution vector is $h = 1$ (control horizon). EA2 having the parameters given by Table 3 uses a direct encoding with real (non-binary) values. It has the usual characteristics listed below:

- The population of each generation has μ individuals.
- The offspring population has λ individuals ($\lambda < \mu$).
- NGen is the number of generations in which the population is evolving.
- The selection strategy is based on Stochastic Universal Sampling using the rank of individuals, which is scaled linearly using selection pressure.
- No crossover operator, because each chromosome encodes a single control input value (whatever the value m is). In our peculiar TP, the solution vector has a single component, i.e., a scalar.
- The mutation operator uses global variance adaptation ([1,2]) of the mutation step. The adaptation is made according to the "1/5 success rule".
- The replacement strategy: the offspring replace the λ worst parents of the generation.

Table 3. The main parameters of EA2.

	λ	μ	*N*Gen	h	s
EA2	20	30	40	1	1.8

We have simulated the closed-loop system using a program whose MA is EA2. A sketch of the simulation program is given in Table 4.

Table 4. Simulation program's pseudo-code list.

Outline of TC_EA_h1_NOISE
$Nt \leftarrow 15$;
Set other program's parameters
$k \leftarrow 0$;
while $k \leq Nt - 1$
1. Get the current value of the state vector X'_k.
2. $[U_k, X_{k+1}, d_{k+1}] \leftarrow EA2(X'_k, Y_{k+1})$
3. Send the control input U_k towards the process.
4. $X\prime_{k+1} \leftarrow X_{k+1} + noise$;
5. $k \leftarrow k + 1$
end while
Display data and graphics

Nt denotes the length of the control horizon as a sampling period number. Instructions #1 and #3 suggest the connection with the real process, in real-time. The state variable is read or estimated,

and the control input is sent toward the process. In our simulation, instruction #3 means that U_k is memorized for further simulations.

The RPM has been implemented, like in the previous section, using a normally distributed noise. For example, we have chosen Mean = 0.04 and StDev = 0.01. The single running's results of this program are presented in Table 5.

Table 5. Simulation's results for the closed-loop (TC with EA2, RPM).

Noise		Rerr [%]				Performance Index		
Mean	**StDev**	**min**	**avg**	**max**	**Rerr(*N*)**	***J****	***J***	ε_r[%]
0.04	0.01	1.68	11.35	29.740	2.36	32.28	30.82	0.045

Remark 2: Although the results from Table 5 are a bit better than those from Table 2, one can assert that the two TC (using SAA or EA2) lead practically toward similar results.

The discrete control input yielded by the TC is depicted in Figure 7. The evolution's pattern is similar to a large extent to that one presented in Figure 5.

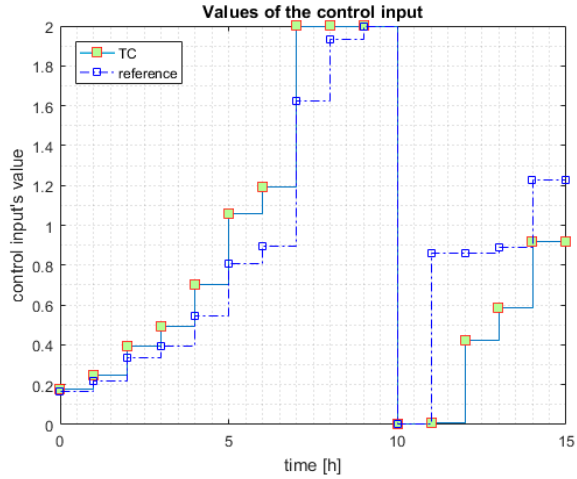


Figure 7. Evolution of the control input (TC with EA2).

Figure 8 presents the comparative evolution of the state variable x_1, x_3, and x_5 in the two cases: the reference trajectory and the closed-loop system with TC using EA2. The values of x_1 and x_3 are multiplied by 3, to render distinguishable the 4 curves.

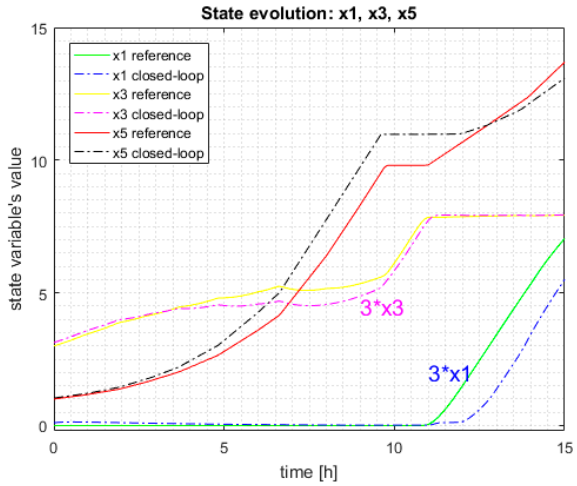


Figure 8. Comparative evolution of x_1, x_3, and x_5.

Remark 3: The state evolutions from Figures 4 and 8 have the same look, a fact that is comprehensible because the two algorithms (SAA and EA2) correctly execute their local optimization task and do not leave their mark on the system evolution.

We have also simulated the closed-loop system for different mean values and using lots of 30 executions. The results are presented in Table 6, where lines correspond to the means of different noises. For each mean's value, a lot of 30 program executions were carried out. The data from the last four columns describes only the typical execution of the lot. In lines #2 and #3, the performance index determined by TC with EA2 has superior values to J^*. The explanation is the same as that given for lines #1 and #2 of Table 1.

Table 6. Results of simulation series (TC with EA2, RPM).

	Noise		**Rerr(*N*) [%]**			**Typical Execution**			
	Mean	**StDev**	**min**	**avg**	**max**	**Rerr(*N*)**	J^*	J	ε_r[%]
1	no *noise*		0	0	0	0	32.28	32.28	0
2	0.01	0.01	0.56	1.29	1.89	1.274	32.28	33.88	−0.049
3	0.02	0.01	0.98	1.45	2.37	1.598	32.28	33.40	−0.036
4	0.03	0.01	1.51	1.84	2.22	1.63	32.28	32.07	0.006
5	0.04	0.01	2.13	2.59	3.06	2.52	32.28	30.34	0.059
6	0.05	0.01	3.14	3.56	4.16	3.52	32.28	28.99	0.102

Line #1 corresponds to the peculiar simulation of the closed-loop, when $X_0 = X_{00}$, and RPM ≡ PM (without noise, see TC_EA_h1_NONOISE.m). This simulation is the test that we mentioned in the previous section. It proves that our EA's implementation works very well, because the sequence of the process' states is just Y_0, Y_1, ... ,Y_{N-1}, Y_N and the sequence of control input values is just $U_0^*, U_1^*, \cdots, U_{N-1}^*$.

5. Discussion

This work proposes a new closed-loop control structure to solve an OCP with a terminal penalty. Section 4 gives an example of how to solve a benchmark OCP using a tracking structure. Emphasis is not placed on the problem but on how to implement the TC. The proposed method is a possible answer to the question: How can we pass from the paper solution computed by an MbA to a closed-loop solution including a real process?

The TC's minimization task involves the employment of an optimization procedure according to Equation (4). We have proposed to use a MA (SAA or EA2) and we have carried out simulations with both algorithms. Why have we presented the two versions?

The first reason is to emphasize that any choice for the optimization procedure is good, while it fulfils the minimization task. Actually, not only an MA may be used. Any other minimization method (deterministic or not) may be employed. Remark 2 and Remark 3 from Section 4.2 underline this idea. If the optimization procedure works well, it will find the best control input $U(k)$ inside a sampling period, whatever the computational complexity would be. In our case, the two MAs execute their task correctly and do not leave their mark on the system evolution. One can eventually notice a difference at the computational complexity level. This difference may be accepted between certain limits.

The second reason is related to Remark 2 as well. The two TCs (using SAA or EA2) lead practically toward similar results. This happens because the RPMs are very much alike in the two simulations: a normal noise having identical parameters added to the PM. In other words, the two RPMs are similar. However, this similarity has its limits because the results from Table 5 are a bit better than those from Table 2. The explanation resides in the noise's stochastic character: the noise's stochastic realizations are not identical.

Therefore one can state that similar RPMs will produce similar results; the TC has no influence (if it works well, that is, it fulfils its minimization task).

What happens when the RPMs are different? In our work, the difference among the real processes is modelled through additive noises. When the RPMs are not similar, the systems' evolutions are different despite the same reference trajectory. The evidence is the data from Table 6. Every line is devoted to a simulation lot with specific noise parameters. The increase of the noise mean implies the decrease of the performance index. The final state's relative tracking error increases as well. Hence, we can notice a degradation of quasi-optimal behavior in comparison with the reference trajectory.

In this paper, we have made a study through simulations of the proposed control structure. The designer of the real-time system has to decide whether this approach would give good results or not. How can the designer take this decision? The key factor is the PM that must replicate, to a large extent, the real process. The construction of a PM must be done following the desired dynamic accuracy. The PM may be enriched with perturbation models or unmodeled dynamics to emulate the real process as accurately as possible. However, this can be a difficult technical task.

With a realistic PM, the tracking structure simulation can help the real-time control system designer take the right decision. Anyhow, a certain degradation of the quasi-optimal behavior will be noticed. The closed-loop does or does not work well, depending on whether the performance index value is acceptable.

The future work on this topic may be focused on some theoretical directions regarding the distance between the real and reference trajectory. The following property results from Remark 1 (Section 4.1):

$$X'_k = X_k \Rightarrow (\exists)\ U_k \in \Omega \text{ s.t. } X_{k+1} = Y_{k+1}$$

Due to the robustness of the structural properties (e.g., the controllability), it is likely to have the following property:

$$(\exists)\varepsilon > 0 \text{ s.t. } \|X'_k - X_k\| < \varepsilon \Rightarrow (\exists)\ U_k \in \Omega \text{ s.t. } X_{k+1} = Y_{k+1}$$

This property could be useful to TC's algorithm in correlation with the accuracy of the PM.

Another direction for future research is to find a theoretical method to minimize the performance index (6) for the local optimal control problem stated in Section 3.2.

Funding: This research received no external funding.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A

The Elements of the Optimal Control Problem

Process Model

$$\begin{aligned}
\dot{x}_1 &= g_1 \cdot (x_2 - x_1) - (u/x_5) \cdot x_1;\\
\dot{x}_2 &= g_2 \cdot x_3 - (u/x_5) \cdot x_2;\\
\dot{x}_3 &= g_3 \cdot x_3 - (u/x_5) \cdot x_3;\\
\dot{x}_4 &= -7.3 \cdot g_3 \cdot x_3 + (u/x_5) \cdot (20 - x_4);\\
\dot{x}_5 &= u;\\
g_1 &= 4.75 \cdot g_3 / (0.12 + g_3);\\
g_2 &= [x_4/(0.1 + x_4)] \cdot e^{-5.0 \cdot x_4}\\
g_3 &= 21.87 \cdot x_4 / [(x_4 + 0.4) \cdot (x_4 + 62.5)];
\end{aligned} \tag{A1}$$

Constraints

Control horizon:

$$0 \le t \le 15 \text{ h}; t_0 = 0; t_f = 15. \tag{A2}$$

Initial conditions:

$$X_{00} = [0.\ 0.\ 1.\ 5.\ 1.]^{\mathrm{T}}. \tag{A3}$$

Bound constraints:

$$0 \leq U(t) \leq 2, \text{ with } 0 \leq t \leq t_f. \tag{A4}$$

Performance index

$$\max_{u(t)} J(x(t_f)), \text{ with } J(x(t_f)) = x_1(t_f) \cdot x_5(t_f) \tag{A5}$$

The maximum performance index

$$J^* = 32.2829 \tag{A6}$$

Reference trajectory:

	x1	x2	x3	x4	x5
Y0	0.0000	0.0000	1.0000	5.0000	1.0000
Y1	0.0000	0.0000	1.1562	4.9645	1.1675
.....					
Y13	1.1236	1.5166	2.6335	0.1133	11.6344
Y14	1.7626	2.1767	2.6408	0.1069	12.5220
Y15	2.3478	2.6983	2.6432	0.1450	13.7504

(A7)

The perturbed initial state of the process is:

$$X_0 = X_{00} + \text{normrnd}(0.08,\ 0.01,\ 1,\ 5). \tag{A8}$$

Appendix B

Evolutionary Algorithm EA1

The particular OCP described in Appendix A has been solved using the evolutionary algorithm EA1. Every chromosome of the solution's population encodes the control profile that is a series of values $U_0, U_1, \cdots, U_{N-1}$. So, the length of the solution vector is $h = 15$ (the control horizon). EA1 uses a direct encoding with real (non-binary) values and has the usual characteristics listed below:

- The population of each generation has μ individuals;
- The offspring population has λ individuals ($\lambda < \mu$);
- NGen is the number of generations in which the population is evolving;
- The selection strategy is based on Stochastic Universal Sampling using the rank of individuals, which is scaled linearly using selection pressure (s);
- A usual one-point crossover operator;
- The mutation operator uses global variance adaptation ([1,2]) of the mutation step. The adaptation is made according to the "1/5 success rule";
- The replacement strategy: the offspring replace the λ worst parents of the generation.

Table A1. The main parameters of EA1.

	λ	μ	*N*Gen	h	s
EA1	30	35	70	15	1.8

The EA1 is implemented by the script "EA_h15_A5.m" (see folder EA1 and *ReadMe.txt*). This program is executed 30 times in order to calculate some statistical parameters. The best solution's state evolution is the reference trajectory used in this paper to implement the closed-loop control structure.

References

1. Kruse, R.; Borgelt, C.; Braune, C.; Mostaghim, S.; Steinbrecher, M. *Computational Intelligence—A Methodological Introduction*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2016.
2. Patrick, S. (Ed.) *Metaheuristics*; Springer: Berlin/Heidelberg, Germany, 2014; ISBN 978-3-319-45401-6.
3. Talbi, E.G. *Metaheuristics-from Design to Implementation*; Wiley: Hoboken, NJ, USA, 2009; ISBN 978-0-470-27858-1.
4. Faber, R.; Jockenhövelb, T.; Tsatsaronis, G. Dynamic optimization with simulated annealing. *Comput. Chem. Eng.* **2005**, *29*, 273–290. [CrossRef]
5. Onwubolu, G.; Babu, B.V. *New Optimization Techniques in Engineering*; Springer: Berlin/Heidelberg, Germany, 2004.
6. Valadi, J.; Siarry, P. (Eds.) *Applications of Metaheuristics in Process Engineering*; Springer: Berlin/Heidelberg, Germany, 2014; ISBN 978-3-319-06507-6.
7. Minzu, V.; Serbencu, A. Systematic Procedure for Optimal Controller Implementation using Metaheuristic Algorithms. *Intell. Autom. Soft Comput.* **2020**, *26*, 663–677. [CrossRef]
8. Mayne, Q.D.; Michalska, H. Receding horizon control of nonlinear systems. *IEEE Trans. Autom. Control.* **1990**, *35*, 814–824. [CrossRef]
9. Hu, X.B.; Chen, W.H. Genetic algorithm based on receding horizon control for real-time implementations in dynamic environments. *IFAC Proc. Vol.* **2005**, *38*, 156–161. [CrossRef]
10. Chiang, P.-K.; Willems, P. Combine evolutionary optimization with model predictive control in real-time flood control of a river system. *Water Resour. Manag.* **2015**, *29*, 2527–2542. [CrossRef]
11. Hu, X.B.; Chen, W.H. Genetic algorithm based on receding horizon control for arrival sequencing and scheduling. *Eng. Appl. Artif. Intell.* **2005**, *18*, 633–642. [CrossRef]
12. Balsa-Canto, E.; Banga, J.R.; Aloso, A.; Vassiliadis, V. Dynamic optimization of chemical and biochemical processes using restricted second-order information. *Comput. Chem. Eng.* **2001**, *25*, 539–546. [CrossRef]