Contribuciones a la Estadística
y a la Investigación Operativa
Sicilia et. al. (editores)

ISBN 84-689-8552-X
páginas 972–987

Optimal control of fed-batch processes with particle swarm optimization

A. Ismael F. Vaz[1], *Eugénio C. Ferreira*[2]

[1]aivaz@dps.uminho.pt, Departamento de Produção e Sistemas, Escola de Engenharia, Universidade do Minho, Campus de Gualtar, 4710-057 Braga-P
[2]ecferreira@deb.uminho.pt, Centro de Engenharia Biológica, Universidade do Minho, Campus de Gualtar, 4710-057 Braga-P

Abstract

Optimal control problems appear in several engineering fields. These problems are often described by sets of nonlinear differential and algebraic equations, usually subject to constraints in the state and control variables. Some bioprocess optimal control problems are revisited and a numerical approach to its solution is introduced.

The numerical procedure used to solve que problems takes advantage of the well know modeling AMPL language, providing an external dynamic library that solve the nonlinear differential equations. The optimal control problem as generally presented belong to the class of semi-infinite programming (SIP) problems. A transformation of the SIP problem results in a nonlinear optimization problem (NLP) that can be address by off-the-shelf optimization software. The NLP formulation results in non-differentiable optimization problems were the global solution is mostly desirable. We apply a particle swarm optimization strategy implemented in the MLOCPSOA [13] solver. Particle swarm optimization (PSO) is a stochastic technique that mimics the social behavior of a swarm.

Keywords: Fed-batch fermentation process, optimal control, particle swarm optimization.

1. Introduction

A great number of valuable products are produced using fermentation processes and thus optimizing such processes is of great economic importance.

Fermentation modeling process involves, in general, highly nonlinear and complex differential equations. Often optimizing these processes results in control optimization problems for which an analytical solution is not possible.

The formulated problem belong to a well known class of semi-infinite programming (SIP) problems (se e.g. [7]). Addressing the SIP problem poses a great challenge as off-the-shelf software for nonlinear, non-differentiable, SIP optimization is not available.

By reformulating the SIP problem as a nonlinear, non-differentiable, optimization problem we are able to solve it with a derivative free optimization technique.

While some of these optimal control model are differentiable its complexity turns the use of gradient information unpractical. The use of numerical algorithms that do not request derivatives alleviates the user from an additional burden of its computation.

We propose a numerical environment to address the optimization of a fermentation process using available software for nonlinear optimization and a development of an external library to handle the dynamic equations resulting from the complex differentiable equations.

In section 2 we introduce the reader to optimal control of fermentation process. Section 3 describes the trajectory optimization process and section 5 presents the implementation details. The numerical results are discussed in section 6 and we conclude in section 7.

2. The optimal control fermentation process

Many microorganisms are used for producing valuable bio-pharmaceuticals products. During the fermentation process the biomass and product concentrations changes considerably. The system dynamic behavior motivates the development of optimization techniques to find the optimum input feeding trajectory of substrate in order to obtain a maximum outcome from the process. The outcome can be, for example, the maximum biomass production with a fixed duration time or the minimum time with a fixed amount of substrate.

The optimal control problem is described by a set of differential equations $\dot{x} = f(x, u, t)$, $x(t_0) = x^0$, $t_0 \leq t \leq t_f$, where x are the state variables and u the input variables which are a function of time t. t_0 and t_f are the inicial and final time, respectively. The performance index J can be generally stated as

$$J(t_f) = \varphi(x(t_f), t_f) + \int_{t_0}^{t_f} \phi(x, u, t)dt,$$

where φ is the performance index of the state variables at final time t_f and ϕ is the integrated performance index during the operation.

Additional constraints on the state and input variables can be imposed that often reflet some physical limitation of the system. The general maximization problem (P) can be posed as

$$\max \; J(t_f) \tag{373}$$

$$s.t. \quad \dot{x} = f(x, u, t) \tag{374}$$

$$\underline{x} \leq x(t) \leq \overline{x}, \quad \forall t \in [t_0, t_f] \tag{375}$$

$$\underline{u} \leq u(t) \leq \overline{u}, \quad \forall t \in [t_0, t_f] \tag{376}$$

Problem (P) belong to a well known class of semi-infinite programming problems [7]. $x(t)$ and $u(t)$ are functional vector whose components are $x_1(t)$, $x_2(t)$, $\ldots$, $u_1(t)$, $u_2(t)$, $\ldots$, respectively. Whenever x represents an n dimensional vector its components are addressed as x_1, x_2, $\ldots$, x_n and vice-versa.

3. The optimization problem

The optimization occurs when determining the optimal input variables u or operational final time t_f. The input variables often represent feeding (or temperature, see [11]) trajectories, *i.e.*, in determining the amount of substrate to be fed into the bioreactor per time unit.

Solving problem (P) in its original formulation is not advisable and unpractical, since there is no available derivative free software for dealing with semi-infinite programming problems (see SIPAMPL [15] and NSIPS [14] for some tools related with semi-infinite programming.) Instead problem (P) can be reformulated as a non-differentiable global optimization problem by imposing a penalty for dealing with constraint (375) and to use linear interpolation for dealing with constraints (376).

Imposing the penalty function for constraints (375) results in redefining the objective function as

$$\hat{J}(t_f) = \begin{cases} J(t_f) & \text{if } \underline{x} \leq x(t) \leq \overline{x}, \forall t \in [t_0, t_f] \\ -\infty & \text{otherwise} \end{cases}$$

An already proposed strategy to deal with constraints (376) is to interpolate the function $u(t)$ by a polynomial (e.g. [11]). A parameter that makes a great influence in the final trajectory precision is the number of discretization points (knots) of the domain $[t_0, t_f]$. A well known effect on increasing the number of knots (and consequently in the polynomial degree) is the increasing poor smoothness of the resulting polynomial. Linear spline interpolation is therefore better suited for approximate function $u(t)$, where increasing the number of knots increases precision without a great affect on smoothness.

We will use a linear interpolating function $w(t)$ (linear spline) to approximate the feeding trajectory function $u(t)$. Let t_i, $i = 0, \ldots, n$, denote the time

instants (knots) and $h_i = t_i - t_{i-1}$, $i = 1, \ldots, n$, the time displacements. We will use fixed time intervals while the linear spline function values $w_i = w(t_i)$, $i = 0, \ldots, n$, are to be computed (in fact we may also use t_i as variables to be optimized, but keeping in mind that ill condition can occur in this case). The linear spline is composed of n linear segments. The spline segment $w^i(t)$, $i = 1, \ldots, n$, is defined as:

$$w^i(t) = w_{i-1} + (w_i - w_{i-1})(t - t_{i-1})(t_i - t_{i-1}), \text{ for } t \in [t_{i-1}, t_i], \quad i = 1, \ldots, n.$$

By using the linear spline $w(t)$ to approximate the feeding trajectory $(u(t))$ we obtain a SIP problem where constraints (376) are replaced by $\underline{u} \leq w(t) \leq \overline{u}$. By careful inspecting this constraint and by using the optimality conditions for SIP we observe that candidate points to make the infinite constraint active, at the solution, are the spline knots and therefore the constraint can be replace by constraints imposing the limit at knots. Constraint (376) can then be replaced by $\underline{u} \leq w_i \leq \overline{u}$, $i = 1, \ldots, n$.

The optimization nonlinear optimization problem (NLP) is then redefined as:

$$\begin{aligned} & \max_{w \in R^{n+1}} \hat{J}(t_f) \\ s.t. \quad & \dot{x} = f(x, w, t) \\ & \underline{u} \leq w_i \leq \overline{u}. \end{aligned}$$

If the initial dynamic system conditions $(x(t_0))$ are to be considered as variable we may also impose some simple bound constraints on its attainable values. We can also consider $h \in R^{n+1}$ and t_f as variables to be optimized increasing the problem dimensional and complexity.

The major motivation for using derivative free optimization codes has to do with the fact that the objective function and the resulting $w(t)$ trajectory function are not differentiable. Recall that even when the constraints are differentiable using $w(t)$ in the dynamic equation makes them non-differentiable. By using a stochastic algorithm we can also expect to obtain the global optimum for the NLP problem.

4. Particle swarm optimization

The particle swarm technique (PS) is based on a population (swarm) of particles. The PS algorithm mimics the social behavior of a swarm in the search of an certain objective (for example a bird swarm looking for food). Each particle represents a point in space and is associated with a velocity that indicates to where the particle is *traveling*. Let k be a time instant (iterations in the optimization context). The new particle position is computed by adding the velocity vector to the current position, i.e., $w^p(k+1) = w^p(k) + v^p(k+1)$, being $w^p(k)$

the particle p, $p = 1, \ldots, s$, position at time instant k, $v^p(k+1)$ the new velocity (at time $k+1$) and s the population size. The velocity update equation is given by $v_j^p(k+1) = \iota(k)v_j^p(k) + \mu\omega_{1j}(k)\left(y_j^p(k) - x_j^p(k)\right) + \nu\omega_{2j}(k)\left(\hat{y}_j(k) - x_j^p(k)\right)$, for $j = 1, \ldots, n$, where $\iota(t)$ is a weighting factor (inertial), μ is the *cognition* parameter and ν is the *social* parameter. $\omega_{1j}(k)$ and $\omega_{2j}(k)$ are random numbers drown from the uniform $(0, 1)$ distribution used for each dimension $j = 1, \ldots, n$. $y^p(k)$ is the particle p position with the best objective function value so far and $\hat{y}(k)$ is a particle position with best function value so far.

A simple scheme of the particle swarm algorithm for optimization is described in the following algorithm.

a. Choose a population size s and a stopping tolerance $v_{tol} > 0$. Randomly initialize the initial swarm $\left\{w^1(0), \ldots, w^s(0)\right\}$ and the initial swarm velocities $v^1(0), \ldots, v^s(0)$.

b. Set $y^i(0) = w^i(0)$, $i = 1, \ldots, s$, and $\hat{y}(0) = \arg\min_{z \in \{y^1(0), \ldots, y^s(0)\}} \hat{J}(z)$. Let $k = 0$.

c. Set $\hat{y}(k+1) = \hat{y}(k)$.

   For $i = 1, \ldots, s$ do (for every particle i):

   - If $\hat{J}(w^i(k)) > \hat{J}(y^i(k))$ then
     - Set $y^i(k+1) = w^i(k)$ (update the particle i best position).
     - If $\hat{J}(y^i(k+1)) > \hat{J}(\hat{y}(k+1))$ then $\hat{y}(k+1) = y^i(k+1)$ (update the particles best position).
   - Otherwise set $y^i(k+1) = y^i(k)$.

d. Compute $v^i(k+1)$ and $w^i(k+1)$, $i = 1, \ldots, s$.

e. If $\|v^i(k+1)\| < v_{tol}$, for all $i = 1, \ldots, s$, then stop. Otherwise, increment k by one and go to Step c.

5. Implementation details

In this section a description of the used environment is made. In the next subsection we introduce the reader to the AMPL [5] modelling language. In subsection 2 we describe the external AMPL library developed for computing the objective function $J(\hat{t}_f)$ (solving the differential equations (374)) and in the last subsection we present the solver used to solve the proposed problems.

5.1. AMPL

AMPL [5] is a modeling language for mathematical programming. While there are other also well known modeling languages for mathematical programming (e.g. GAMS [2]), AMPL provides an easy to use and powerful language. AMPL also provides an interface that allows communication with a wide variety of solver (e.g. LOQO [12], NPSOL [6]). The possibility to load an external dynamic library is exploited in this paper in order to solve ordinary differential equations.

The optimization problem described in section 3 can easily be written in AMPL. A short example is presented bellow and details regarding the external function `chemotherapy` in the objective function and the used solver are postponed to the next subsections.

```
function chemotherapy;        # external function to be called
param Tumor_mass := log(100); # Tumor cells N=10^12*exp(-x1)
param Drug       := 0;        # Drug concentration in the body
param Cumulative := 0;        # Cumulative effect of the drug
param n := 4;                 # Number of times instants (knots-1)
param h{1..n} := 21;          # Time instants, could be variables.
var w{1..n+1};                # Spline knots

maximize obj:          # maximize objective function
    chemotherapy(0, n, {i in 1..n} h[i], {i in 1..n+1} w[i],
       Tumor_mass, Drug, Cumulative);
subject to hbounds {i in 1..n}:  # constraints on time instants
    1<= h[i] <= 100;             # AMPL just checks for correctness
subject to wbounds {i in 1..n+1}: # constraints on drug delivery
    0.01<= w[i] <= 50;            # problem constraints

option solver mlocpsoa;   # mlocpsoa solver
option mlocpsoa_options 'mlocal=0 size=60 maxiter=1000';
         # global search, population size of 60, maximum of 1000 iterations
solve;                    # solve problem
```

Additional constraints can easily be incorporated into the model. If, for example, a constraint in the total allowed glucose addition (t_G) is to be imposed, the constraint $\sum_{i=0}^{n-1} h_{i+1}(w_i + w_{i+1})/2 \leq t_G$ can easily be considered in the model file by adding

```
subject to totalfeed:
   sum {i in 0..n-1} (h[i+1]*(w[i]+w[i+1])/2)<=t_G;
```

and to properly define the `t_G` parameter.

5.2. External dynamic library

In its execution AMPL is able to load an external library that provides ad-

ditional functions. By default AMPL load a library named `amplfunc.dll`. By setting the operation system environment variable `AMPLFUNC`, AMPL is able to load any other dynamic library instead (in this case under a Microsoft DOS prompt we write `set AMPLFUNC=fed-batch.dll`). The `fed-batch.dll` library provides five external functions to AMPL whose names are: `penicillin` for case study 1, `ethanol` for case study 2, `chemotherapy` for case study 3, `hprotein` for case study 4 and `rprotein` for case study 5.

The `chemotherapy` function prototype is

```
chemotherapy(0, n, {i in 1..n} h[i], {i in 1..n+1} w[i],
             Tumor_mass, Drug, Cumulative);
```

where the first parameter selects a linear spline (on future research we plan to test other approximation techniques), `n` is the number of `h[i]` displacements, `w[i]` are the linear spline values at the time instants t_i, $i = 0, \ldots, n$ (with $t_0 = 0$) and the remaining parameters are the initial conditions of the differential equations (374).

The ordinary differential equations (374) are solved by calling the CVODE [4] package where the Newton iteration with the CVDiag module was selected.

At each call to the `chemotherapy` function the linear spline is computed with the provided data and the objective function value is returned. The objective function expression is therefore coded in the external library.

5.3. MLOCPSOA

MLOCPSOA [13] stands for Multi-LOCal Particle Swarm Optimization Algorithm. Multi-local optimization addresses the finding of all the local and global optima for an optimization problem. While MLOCPSOA was developed with multi-local optimization in mind by setting an option it reverses to the traditional particle swarm algorithm.

MLOCPSOA provides an interface to AMPL, allowing problems to be easily coded and solved in this modeling language. The NLOCPSOA allows a wide variety of algorithm parameters to be set. The used parameters are `size` for the population size (defaults to $\min(6^n, 1000)$), `maxiter` for the maximum allowed iterations (defaults to 2000) and `mlocal` for multi-local search (defaults to 0 – global search instead of multi-local search). The reader is pointed for the user manual for further details. Parameter can either be set in the model file, as shown in subsection 1, or in any way allowed by AMPL (see [5] for details).

6. Numerical results

Numerical results were obtained for the five case studies described in the appendix. We have used a linear spline interpolating function on all problems.

The time displacements were kept fixed and the best control feeding trajectory was approximated by computing the knots function value. The parameters used are presented together with the numerical results in Table 1. 'Problem' column refers to the case study (AMPL model file); NT is the number of trajectories in the model; n is the number of time displacements (problems with $n+1$ variables) and equal displacements are considered ($h_i = t_f/n$, $i = 1, \ldots, n$).

MLOCPSOA used a population size of 60 and a maximum of 1000 iterations (reaching a maximum of 60000 function evaluations). Since MLOCPSOA is a stochastic algorithm we performed 10 solver runs for each problem and the best solutions obtained are report on Table 1. $\hat{J}(t_f)$ is the objective function value and t_f is the final time (t_0 is assumed 0 for all cases). We present the solution obtained by the MLOCPSOA solver and the previous (indicated in 'Ref.') solution.

			MLOCPSOA		Previous		
Problem	NT	n	$\hat{J}(t_f)$	t_f	$\hat{J}(t_f)$	t_f	Ref.
`penicillin`	1	5	88.29	132.00	87.99	132.00	[1]
`ethanol`	1	5	20379.50	61.20	20839.00	61.17	[1]
`chemotherapy`	1	4	16.83	84.00	17.48	84.00	[1]
`hprotein`	1	5	32.73	15.00	32.40	15.00	[9, 10]
`rprotein`	2	5	0.12	10.00	0.16	10.00	[9, 8]

Table 1: Numerical results

We provide, in figures 1 to 5, plots of the state and control profiles for all study cases problems.

No attempt is made on comparing our results with the published ones, since implementation details are not reported in previous papers. The use of different techniques to solve the differential equations, discretization step and precision can influence the objective value at the solution found.

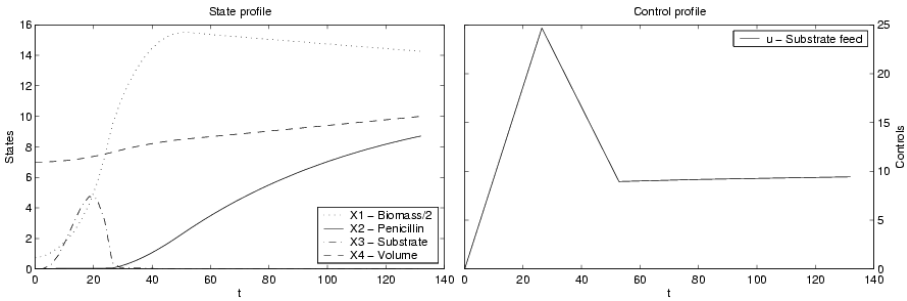

Figure 1: State and control profiles for best optimal control, case study 1

By allowing the spline displacements to be variable and using the objective function $\bar{J}(t_f, h) = (\hat{J}(t_f))/(\sum_{i=1}^{n} h_i)$ we can easily compute the profile with

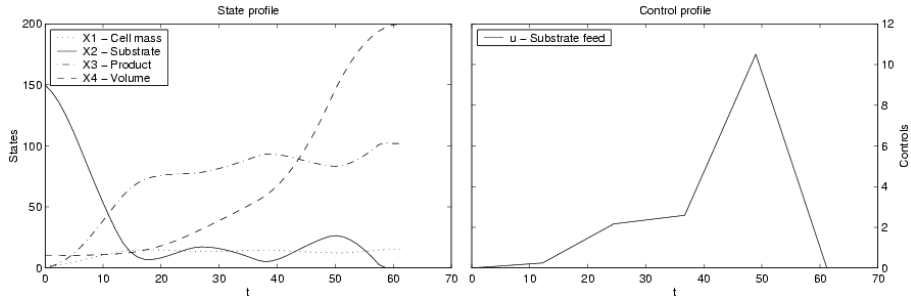

Figure 2: State and control profiles for best optimal control, case study 2

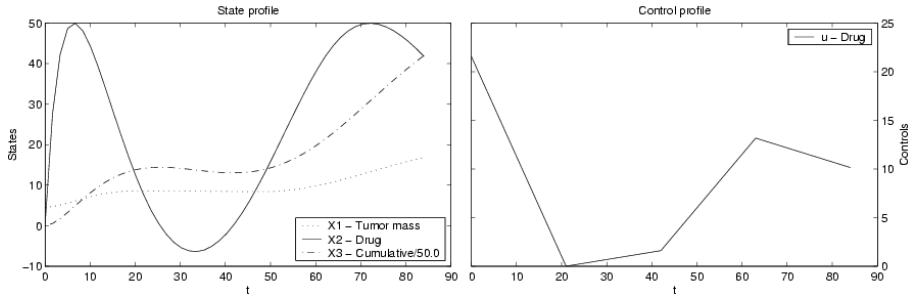

Figure 3: State and control profiles for best optimal control, case study 3

the best ratio per unit time. The problem that considers both the objective of optimizing the performance $J(t_f)$ and computing the minimum time is a biobjective problem. Solving a multiobjective optimization problem by a uniobjective optimization problem is only possible if some requirements are met (see, for example, [3]).

In order not to allow the solution to diverge and a division by zero to occur in the linear splines we have imposed additional constraints on the h_i, $i = 1, \ldots, n$ variables ($0.01 \leq h_i \leq h_i^{max}, i = 1, \ldots, n$). We reports solution for all case studies in Table 2. We repeat the numerical results obtained for the fixed time to allow a better comparison with the problems instances where the time displacements are variables.

	Fixed time		Variable time			
Problem	$\hat{J}(t_f)$	t_f	$\hat{J}(t_f)/\sum_{i=1}^{n} h_i$	$\hat{J}(t_f)$	t_f	h_i^{max}
`penicillin`	88.29	132.00	0.92	76.16	83.20	60
`ethanol`	20229.50	61.20	604.20	14417.50	23.86	20
`chemotherapy`	16.83	84.00	0.70	8.06	11.52	25
`hprotein`	32.73	15.00	17.57	439.18	25	5
`rprotein`	0.12	10.00	2.38	59.61	25	5

Table 2: Numerical results with variable time

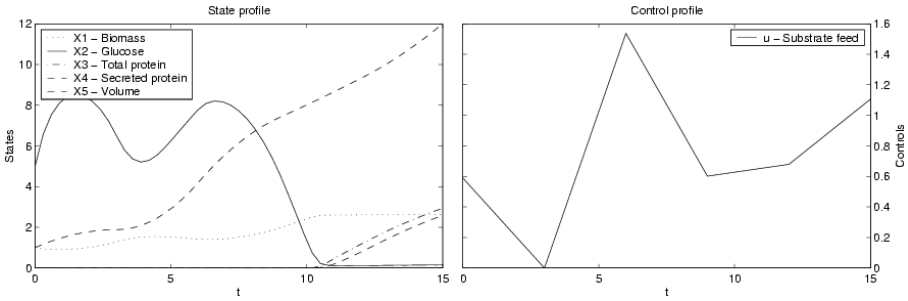

Figure 4: State and control profiles for best optimal control, case study 4

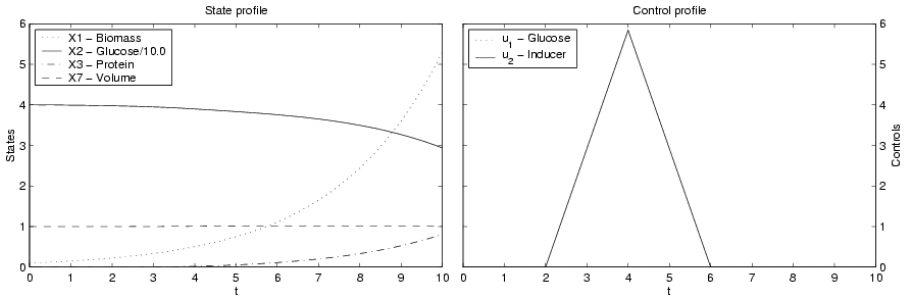

Figure 5: State and control profiles for best optimal control, case study 5

$\bar{J}(t_f)$ gives an equal importance to time and performance $\hat{J}(t_f)$. Note that in the first three case studies a decrease in time led to a worst performance while in the last two case studies the opposite occurred.

The bioprocesses initial conditions can also be considerer as variables. While in the optimization context the problems are well defined in the biological sense the solution may not be reasonable. For instance it makes no sense to initialize a bioreactor to produce ethanol with an initial ethanol concentration greater than zero. The new initial condition variables are allowed to vary between a factor of 0.1 and 10 of its initial value presented in the appendix. Initial condition variables with an initial value of zero are allowed to vary between 0 and 10. The numerical results obtained are shown in Table 3.

Problem	$\hat{J}(t_f)$	$x_1(0)$	$x_2(0)$	$x_3(0)$	$x_4(0)$	$x_5(0)$	$x_6(0)$	$x_7(0)$
`penicillin`	122.33	15.00	0.00	0.00	6.06	-	-	-
`ethanol`	43285.20	4.56	400.00	10.00	100.00	-	-	-
`chemotherapy`	54.53	46.05	0.00	10.00	-	-	-	-
`hprotein`	1002.74	10.00	44.73	10.00	0.00	10.00	-	-
`rprotein`	165.85	1.00	400.00	10.00	0.00	0.10	2.98	10.00

Table 3: Numerical results with variable initial conditions

7. Conclusions

The optimal control of fed-batch bioprocesses present challenge nonlinear optimization problems where derivatives do not exist or are unpractical. We address the state constraints by combining it with the problem objective function in an infinite penalty function. The resulting SIP problem is approximated by a nonlinear finite problem where the control constraints are approximated by linear splines. The resulting nonlinear optimization problem is characterized by possessing a nonconvex nondifferentiable objective function subject to bound constraints in the variables.

Particle swarm optimization belongs to a class of stochastic algorithms for global optimization and its main advantages are the easily parallelization and simplicity. In this paper we use the MLOCPSOA [13] implementation of the particle swarm paradigm to obtain numerical results with some problem formulations.

No attempt is made to compare the obtained solutions with previous works, since the implementation depends on many parameters external to the problem and algorithm (ordinary differential equation solver, discretization step, etc.). Nevertheless the MLOCPSOA proved to be able to find the problem solution with reasonable accuracy and the particle swarm paradigm proved to be a valuable tool in solving these optimal control problems.

The MLOCPSOA interface with AMPL allowed an easy and fast way to code the five case studies problems. Using the AMPL modeling language together with a developed external dynamic library allows a great flexibility in the problem formulation. We considered three instances for each case study, one allowing only the spline knots values to be variables, other by allowing also the spline knots values and spline knots (time displacements) to be variables and the last one by considering the spline knots values and the initial conditions to be variables.

1. Case studies

1.1. Optimal control of a fed-batch fermentor for penicillin production

This problem considers a fed-batch process for penicillin production, as described in [1]. This problem was studied in several previous works and the reader is pointed to [1] for further references.

Local gradient methods applied to this problem have experienced convergence problems if initialized with poor initial guesses.

The optimization problem (in (P) formulation) is:

$$\begin{aligned}
&\max_{u(t)} \; J(t_f) \equiv x_2(t_f)x_4(t_f) \\
&s.t. \quad \dot{x}_1 = h_1 x_1 - u x_1/(500 x_4), \;\; \dot{x}_2 = h_2 x_1 - 0.01 x_2 - u x_2/(500 x_4) \\
&\qquad \dot{x}_3 = -(h_1 x_1)/0.47 - h_2 x_1/1.2 - 0.029 x_1 x_3/(0.0001 + x_3) + \\
&\qquad\quad + u(1 - x_3/500)/x_4, \;\; \dot{x}_4 = u/500 \\
&\qquad 0 \le x_1(t) \le 40, \;\; 0 \le x_3(t) \le 25, \;\; 0 \le x_4(t) \le 10, \;\; 0 \le u(t) \le 50, \\
&\qquad \forall t \in [t_0, t_f]
\end{aligned}$$

with

$$h_1 = 0.11(x_3/(0.006 x_1 + x_3)) \;\text{ and }\; h_2 = 0.0055(x_3/(0.0001 + x_3(1 + 10 x_3))$$

where x_1, x_2 and x_3 are the biomass, penicillin and substrate concentrations (g/L), and x_4 is the volume (L). The initial conditions are $x(t_0) = (1.5, 0, 0, 7)^T$.

1.2. Optimal control of a fed-batch reactor for ethanol production

This optimal control problem considers a fed-batch reactor for ethanol production as described in [1]. Once again the reader is refereed to [1] for other references on this case study. As with the previous case study convergence problems have been reported with gradient based methods.

The optimization problem is:

$$\begin{aligned}
&\max_{u(t)} \; J(t_f) \equiv x_3(t_f)x_4(t_f) \\
&s.t. \quad \dot{x}_1 = g_1 x_1 - u x_1/x_4, \;\; \dot{x}_2 = -10 g_1 x_1 + u(150 - x_2)/x_4, \;\; \dot{x}_4 = u \\
&\qquad \dot{x}_3 = g_2 x_1 - u x_3/x_4, \;\; 0 \le x_4(t_f) \le 200, \;\; 0 \le u(t) \le 12, \;\; \forall t \in [t_0, t_f]
\end{aligned}$$

with

$$\begin{aligned}
g_1 &= (0.408/(1 + x_3/16))(x_2/(0.22 + x_2)) \\
g_2 &= (1/(1 + x_3/71.5))(x_2/(0.44 + x_2))
\end{aligned}$$

where x_1, x_2 and x_3 are the cell mass, substrate and product concentrations (g/L), and x_4 is the volume (L). The initial conditions are $x(t_0) = (1, 150, 0, 10)^T$.

1.3. Optimal drug scheduling for cancer chemotherapy

This problem consists in determining the optimal cancer drug scheduling to decrease the size of a malignant tumor. The drug concentration must be kept below some level throughout the treatment period and the cumulative toxic effect of the drug must be kept below the ultimate tolerance level. This problem was also address in previous works (see [1] for further references.)

The optimization problem is:

$$\begin{aligned}
&\max_{u(t)} \ J(t_f) \equiv x_1(t_f) \\
&s.t. \ \ \dot{x}_1 = -k_1 x_1 + k_2(x_2 - k_3) \times H\{x_2 - k_3\} \\
&\qquad \dot{x}_2 = u - k_4 x_2, \ \ \dot{x}_3 = x_2 \\
&\qquad x_2(t) \leq 50 \ \ x_3(t) \leq 2.1 \times 10^3, \ \ 0 \leq u(t), \ \ \forall t \in [t_0, t_f]
\end{aligned}$$

with $H\{x_2 - k_3\} = 1$ if $x_2 \geq k_3$ and $H\{x_2 - k_3\} = 0$ if $x_2 < k_3$, where the tumor mass cells is given by $N = 10^{12} \times exp(-x_1)$, x_2 is the drug concentration in the body in drug units [D] and x_3 is the cumulative effect of the drug. The parameters are: $k_1 = 9.9{\times}10^{-4}$days, $k_2 = 8.4{\times}10^{-3}$days^{-1}[D^{-1}], $k_3 = 10$[D^{-1}] and $k_4 = 0.27$days^{-1}. The initial conditions are $x(t_0) = (\ln(100), 0, 0)^T$.

Some extra constraints are imposed as there should be at least a 50% reduction in the size of the tumor every three weeks. The treatment period considered is 84 days and therefore the extra constraints are $x_1(21) \geq \ln(200)$, $x_1(42) \geq \ln(400)$ and $x_1(63) \geq \ln(800)$.

These extra constraints are also handled by the penalty procedure as described in section 3.

1.4. Optimizing of the glucose feeding in a fed-batch bioreactor for protein production

This problem consists in the optimization of an heterologous protein process in a fed-batch bioreactor described in [9, 10].

The optimization problem is:

$$\begin{aligned}
&\max_{u(t)} \ J(t_f) \equiv x_4(t_f) x_5(t_f) \\
&s.t. \ \ \dot{x}_1 = \mu x_1 - D x_1, \ \ \dot{x}_2 = -7.3\mu x_1 - D(x_2 - x_2^0) \\
&\qquad \dot{x}_3 = f_P x_1 - D x_3, \ \ \dot{x}_4 = \chi(x_3 - x_4) - D x_4, \ \ \dot{x}_5 = u \\
&\qquad 0 \leq u(t) \leq 10, \ \ \forall t \in [t_0, t_f]
\end{aligned}$$

with

$$\begin{aligned}
&\mu = 21.87 x_2/((x_2 + 0.4)(x_2 + 62.5)), \ \ f_P = x_2 exp(-5x_2)/(x_2 + 0.1) \\
&\chi = 4.75\mu/(0.12 + \mu), \ \ D = u/x_5
\end{aligned}$$

where x_1, x_2, x_3 and x_4 are the biomass, glucose, total protein and secreted protein concentrations (g/L), and x_5 is the volume (L). The parameter x_2^0 is 20g/L and the initial conditions are $x(t_0) = (1.0, 5.0, 0.0, 0.0, 1.0)^T$.

1.5. Optimal control of a fed-batch fermentation for protein production

This optimal control problem is a maximization of a fed-batch fermentation

for a protein production by recombinant bacteria ([8, 9]). In this problem two streams are fed to the fermenter at different volumetric rates. Glucose is fed at a volumetric feed rate $u_1(t)$ and concentration in the feed of $x_2^F = 100.0$g/L, while the inducer is fed with a different stream at a rate $u_2(t)$ with concentration in the stream of $x_4^F = 4.0$g/L.

The optimization problem is:

$$\begin{aligned}
&\max_{u(t)} \; J(t_f) \equiv x_3(t_f)x_7(t_f)/Q - \int_{t_0}^{t_f} u_2(\tau)x_4^F d\tau \\
&s.t. \quad \dot{x}_1 = \mu x_1 - Dx_1, \;\; \dot{x}_2 = -Y^{-1}\mu x_1 - Dx_2 + u_1 x_2^F/x_7 \\
&\qquad \dot{x}_3 = R_{fp}x_1 - Dx_3, \;\; \dot{x}_4 = -Dx_4 + u_2 x_4^F/x_7 \\
&\qquad \dot{x}_5 = -a_1 x_5, \;\; \dot{x}_6 = a_2(1 - x_6), \;\; \dot{x}_7 = u_1 + u_2 \\
&\qquad 0 \le u_1(t) \le 1, \;\; 0 \le u_2(t) \le 1, \;\; \forall t \in [t_0, t_f]
\end{aligned}$$

with

$$\mu = 0.407\psi(x_5 + 0.22x_6/(0.22 + x_4)), \; R_{fp} = 0.095\psi(0.0005 + x_4)/(0.022 + x_4)$$
$$D = (u_1 + u_2)/x_7, \;\; \psi = x_2/(0.108 + x_2 + x_2^2/14814.8)$$
$$a_1 = a_2 = 0.09x_4/(0.034 + x_4)$$

where $Y = 0.51$ is the growth yield coefficient, $Q = 5$ is the ratio of protein value to inducer cost, x_1 is the biomass (g/L), x_2, x_3, and x_4 are the glucose, protein and inducer concentrations (g/L), x_5 and x_6 are the inducer shock and inducer recovery factors, and x_7 is the volume (L). The initial conditions are $x(t_0) = (0.1, 40, 0.0, 0.0, 1.0, 0.0, 1.0)^T$.

2. Bibliography

[1] J.R. Banga, E.Balsa-Canto, C.G. Moles, and A.A. Alonso. Dynamic optimization of bioprocesses: Efficient and robust numerical strategies. *Journal of Biotechnology*, 117:407–419, 2005.

[2] A. Brooke, D. Kendrick, A. Meeraus, and Ramesh Raman. GAMS: A User's Guide. http://www.gams.com/, December 1998.

[3] C.A. Coello Coello, D.A. Van Veldhuizen, and G.B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic Algorithms and Evolutionary Computation)*. Kluwer Academic Publishers, New York, 2002. ISBN 0-3064-6762-3.

[4] S.D. Cohen and A.C. Hindmarsh. CVODE, a Stiff/Nonstiff ODE solver in C. *Computers in Physics*, 10(2):138–143, 1996.

[5] R. Fourer, D.M. Gay, and B.W. Kernighan. A modeling language for mathematical programming. *Management Science*, 36(5):519–554, 1990.

[6] P.E. Gill, W. Murray, M.A. Saunders, and M.H. Wright. *User's Guide for NPSOL: A Fortran Package for Nonlinear Programming*. Stanford University, 1986.

[7] R. Hettich and K.O. Kortanek. Semi-infinite programming: Theory, methods, and applications. *SIAM Review*, 35(3):380–429, 1993.

[8] J. Lee and W.F. Ramirez. Optimal fed-batch control of induced foreign protein-production by recombinant bacteria. *AIChE Journal*, 40(5):899–907, 1994.

[9] R. Oliveira and R. Salcedo. Benchmark testing of simulated annealing adaptative random search and genetic algorithms for global optimization of bioprocesses. In B. Ribeiro, R.F. Albrecht, A. Dobnikar, D.W. Pearson, and N.C. Steele, editors, *Adaptative and Natural Computing Algorithm*, pages 292–295, Wien, 2005. Springer-Verlag.

[10] S. Park and W.F. Ramirez. Optimal production of secreted protein in fed-batch reactors. *AIChE Journal*, 34(9):1550–1558, 1988.

[11] R. Simutis and A. Lübbert. A comparative study on random search algorithms for botechnical process optimization. *Journal of Biotecnology*, 52:245–256, 1997.

[12] R.J. Vanderbei and D.F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.

[13] A.I.F. Vaz. MultiLOCal particle swarm optimization algorithm v1.1: User's manual. http://www.norg.uminho.pt/aivaz/.

[14] A.I.F. Vaz, E.M.G.P. Fernandes, and M.P.S.F. Gomes. NSIPS: Nonlinear Semi-Infinite Programming Solver. *Technical Report ALG/EF/1-2001*, October 2001. http://www.norg.uminho.pt/aivaz/.

[15] A.I.F. Vaz, E.M.G.P. Fernandes, and M.P.S.F. Gomes. SIPAMPL: Semi-infinite programming with AMPL. *ACM Transactions on Mathematical Software*, 30(1):47–61, March 2004.