Implementation of IDP Technique by the use of MATLAB as a Tool for the Chemical Engineering Education

Pereira, P. R. A.
Municipal College Professor Franco Montoro, Environmental Engineering,
CP 293, CEP13843-971, Mogi Guaçu, SP, Brazil
prapereira@hotmail.com

Fileti, A. M. F.[1]

Abstract - **The present work is concerned with the use of MATLAB as an important tool for the Chemical Engineering Education in order to demonstrate the behavior of chemical processes. To illustrate its importance, a methyl metacrylate (MMA) batch polymerization reactor using ethyl acetate as solvent and benzoyl peroxide as reaction initiator was chosen. To optimize the MMA conversion rate, the phenomenological model and energy balance equations were solved successfully through an algorithm based on Iterative Dynamic Programming (IDP) strategy developed for MATLAB, making use of ode23tb function to integrate the system differential equations. It is known from the literature that IDP mechanism demands high computational efforts to reach a global optimum convergence, depending on the process complexity and the way how IDP configuration is set up. In this work, due to MMA polymerization reaction complexity, its high nonlinearity and IDP configuration, strong computational efforts were experimented. In despite of that, the simulation results demonstrated that MATLAB is quite suitable to implement IDP reinforcing its importance for engineering education. To make MATLAB even more popular among IDP users and overcome the need of creating specific algorithms in MATLAB environment, it is highly recommended the development of an IDP MATLAB Standard Toolbox.**

Index Terms – Batch Polymerization, Dynamic Programming, Engineering Education, IDP, MATLAB.

INTRODUCTION

Undoubtedly the use of computing tools is fundamental for the engineering education and consequently for the engineering professional practice. The advent of modern and more powerful computers has contributed to a fast and irreversible change on the way of facing the research and the development of engineering and other sciences [1]. It is urging that the present engineers be prepared to develop solid abilities to model, program, simulate and visualize chemical system dynamic behavior [2].

Most of the problems related to chemical process present nonlinear nature and high dimensionality. They are represented by differential equations that require numerical solutions, often demanding an extraordinary computational effort to be achieved. To overcome it and all other difficulties in solving computationally engineering problems, the use of MATLAB package appears as an important tool for both undergraduate and graduate engineer students involved with system and control areas [3].

In practice, the use of MATLAB package for teaching and research is overwhelming supported by an extensive literature that provides a representative sampling of various engineering topics. It certainly helps in reducing the time spent in developing computational tasks to solve problems and reinforce the students understanding of theoretical principles through simulations and graphical interfaces easy to learn [3], [4].

Moreover, MATLAB package has being updated with a large number of specific application toolboxes supplying reliable routines for diverse analyses and optimizations [4]. In general, the optimization tasks are common routines in engineering practice and can be divided into two categories: static and dynamic optimization (often recognized as “optimal control problems (OCPs)”). It is well known up to now that dynamic optimization has not achieved a development level as static optimization; at the same time as numerical methods for solving differential equations have not reached the level of methods for solving differential equations have [5].

Although it is possible to find different packages for numerically solving dynamic optimization or OCPs, as such SOCS, RIOTS_95, DIRCOL, and MISER3, none of them can solve all the variety of existing problems by itself. On the other hand, the Bellman’s method denominated dynamic programming (DP) is powerful and could solve all types of OCPs by the sharing of a complex optimization problem, into a number of simpler problems and their solution would lead to the original problem solution [5]. As a matter of fact, the use of DP to solve OCPs is still not popular due to some limitations widely explained on the literature, restringing its application for problems of very low dimensionality. In order to overcome these limitations, Luus proposed an innovative and robust method known as Iterative Dynamic Programming (IDP) able to solve a wide range of hard OCPs [6]-[7].

1 Fileti, A. M. F., State University of Campinas, School of Chemical Engineering, CP 6066, CEP13083-970, Campinas, SP, Brazil, frattini@desq.feq.unicamp.br

To certify IDP robustness, a well known characteristic problem of high dimensionality was chosen for this work. The system is composed by a methyl metacrylate (MMA) batch solution polymerization reactor using ethyl acetate as solvent and benzoyl peroxide as the reaction initiator. The justification for the choice of MMA polymerization reaction is related to its growing commercial valorization and the constant necessity of controlling its main properties to keep the final product quality. For this study, the MMA conversion rate was adopted among others as the variable to be optimized.

The explanation for the choice of IDP to solve the above OCP is based on the fact that in most of batch polymerization processes the use Pontryagin's Maximum Principle [8] for optimization is predominant among other techniques. In a complementary way, the motivation for the choice of MATLAB as environment of IDP implementation is due to the facts:

1. It is easy to program, manipulate the routines, simulate and visualize the behavior of dynamic process in MATLAB environment;
2. Mathworks has still not developed a toolbox for the optimization task through IDP strategy. Consequently, it seems clear that the most part of optimizations are performed through other packages than MATLAB;
3. The implementation of IDP in MATLAB environment can be understood as a challenging task of programming.

Thus, the intention of this work was principally to show the relevance of MATLAB package as an important tool for the engineering education, and at the same time, certify the robustness of IDP in solving OCPs of high dimensionality, and demonstrate that the implementation of IDP in MATLAB is quite feasible, reinforcing the need of Mathworks in developing a standard toolbox for IDP, eliminating the hard task and long time consuming of programming.

MATHEMATICAL MODELING

The phenomenological model adopted for the MMA solution polymerization reaction was firstly proposed by Seth and Gupta [9]. Chakravarthy [10] improved this model by introducing the gel, cage and glass effects. This way, the mass balance and moment equations considering all the reagents in the reactor generated the following Ordinary Differential Equations (ODE):

BPO initiator consumption:

$$\frac{dI}{dt} = -k_d I + R_{li}(t) \tag{1}$$

Mass balance to the primary radical:

$$\frac{dR}{dt} = 2fk_d I - k_i \frac{RM}{V_l} \tag{2}$$

Monomer consumption:

$$\frac{dM}{dt} = -(k_p + k_f)\frac{M\lambda_o}{V_i} - k_i\frac{RM}{V_l} - k_s S\frac{\lambda_0}{V_l} + R_{lm}(t) - R_{vm}(t) \tag{3}$$

Solvent consumption:

$$\frac{dS}{dt} = R_{lm}(t) - R_{vs}(t) \tag{4}$$

Moments of the live polymer concentrations distributions, γ_i:

$$\frac{d\lambda_0}{dt} = k_i\frac{RM}{V_l} - k_t\frac{\lambda_0^2}{V_l} \tag{5}$$

$$\frac{d\lambda_1}{dt} = k_i\frac{RM}{V_l} + k_p M\frac{\lambda_0}{V_l} - k_t\frac{\lambda_0\lambda_1}{V_l} + (k_f M + k_s S)\frac{(\lambda_0 - \lambda_1)}{V_l} \tag{6}$$

$$\frac{d\lambda_2}{dt} = k_i\frac{RM}{V_l} + k_p M\frac{(\lambda_0 + 2\lambda_1)}{V_l} - k_t\frac{\lambda_0\lambda_2}{V_l} + (k_f M + k_s S)\frac{(\lambda_0 - \lambda_2)}{V_l} \tag{7}$$

Moments of the dead polymer concentration distributions, μ_i:

$$\frac{d\mu_0}{dt} = \left(\frac{1}{2}ktc + k_{td}\right)\frac{\lambda_0^2}{V_l} + (k_f M + k_s S)\frac{\lambda_0}{V_l} \tag{8}$$

$$\frac{d\mu_1}{dt} = k_t\frac{\lambda_0\lambda_1}{V_l} + (k_f M + k_s S)\frac{\lambda_1}{V_l} \tag{9}$$

$$\frac{d\mu_2}{dt} = k_{tc}\frac{\lambda_1^2}{V_l} + k_t\frac{\lambda_0\lambda_2}{V_l} + (k_f M + k_s S)\frac{\lambda_2}{V_l} \tag{10}$$

Gel effect parameters:

$$\frac{d\xi_m}{dt} = R_{lm}(t) - R_{vm}(t) \tag{11}$$

$$\frac{d\xi_{ml}}{dt} = R_{lm}(t) \tag{12}$$

Energy balance in the reactor:

$$\rho c\frac{dT}{dt} = (-\Delta H_P)k_P M\lambda_0 + UA(T_j - T) \tag{13}$$

Energy balance in the jacket:

$$\rho_w c_w\frac{dT_j}{dt} = UA(T - T_j) + U_\infty A_\infty(T_\infty - T_j) + [Q - F_{cw}\rho_w c_w(T_j - T_{cw})] \tag{14}$$

The additional equations related to the gel, cage and glass effects were taken from the Chakravarthy work [10]. To complete the group of ordinary differential equations (1 to 14) that must be integrated, below follows the monomer conversion rate equation, which was defined as the performance index to be maximized by the optimization IDP:

$$X_m = 1 - \frac{M}{\zeta_{m1}} \tag{15}$$

Finally, the state vector **x** can be expressed as:

$$x = [I, M, R, S, \lambda_0, \lambda_1, \lambda_2, \mu_0, \mu_1, \mu_2, \zeta_m, \zeta_{m1}, T, T_j, X_m] \tag{16}$$

1 Fileti, A. M. F., State University of Campinas, School of Chemical Engineering, CP 6066, CEP13083-970, Campinas, SP, Brazil, frattini@desq.feq.unicamp.br

IMPLEMENTATION OF IDP IN MATLAB ENVIRONMENT

As all the steps involving the traditional IDP algorithm are very well documented on the literature [6]-[7], they are not described in its totality in this work. To illustrate them in a summarized way to permit the readers a fast understanding on the sequence of IDP steps, they are explained concomitantly with their insertion in MATLAB environment.

To accomplish the IDP implementation, it was necessary to develop a specific algorithm subdivided in principal routine and additional subroutines in such way to permit a minimum flexibility in setting up IDP configuration. This algorithm permits to divide the time interval [0, *tf*] into *P* [0, 20] stages of equal length L. To choose the number of values for the control variable **u** (cooling water *F*) and the number of grid points *N*, two subroutines were created making use of "randn" MATLAB function (an N-by-N matrix with random entries) generate random values. In this subroutine it was introduced the clipping technique to assure the random values inside the region permitted for control, as:

- Control variable:

$$F = F_{\max} = 1.6x10^{-6}\,mL/s \qquad if \qquad F \geq F_{\max} \geq 1.6x10^{-6}\,mL/s \qquad (17)$$

$$F = F_{\min} = 0 \qquad if \qquad F \leq F_{\min} \leq 0 \qquad (18)$$

- State variables:

$$T = T_{\max} = 340.2 \qquad if \qquad T \geq T_{\max} \geq 340.2 \qquad (19)$$

$$T = T_{\min} = 332.2 \qquad if \qquad T \leq T_{\min} \leq 332.2 \qquad (20)$$

$$X_m = 1 \qquad if \qquad X_m > 1 \qquad (21)$$

$$X_m = 0 \qquad if \qquad X_m < 0 \qquad (22)$$

The system heat necessity was supplied by two turn on and turn off heaters, following the clipping rules:

$$Q = Q_{\max} = 0.12\,kJ/s \qquad if \qquad T \leq T_{\min} \leq 332.2K \qquad (23)$$

$$Q = Q_{\min} = 0.075\,kJ/s \qquad if \qquad T \geq T_{\max} \geq 340.2K \qquad (24)$$

From the principal routine it is possible to define any total number of iterations *j* to be used in every pass, which can be also defined without restrictions. Also in this routine the region size vector **r** can be set up for the chosen control policy. After setting up these variables the principal routine calls an integration subroutine to solve the ODE (1) to (15), by means of ode23tb MATLAB function. This function is structured by an implicit Runge-Kutta formula with a first stage that is a trapezoidal rule step. The second stage is a backward differentiation formula of order two, where the same iteration matrix is used in evaluating both stages [11], being suitable for solving stiff differential equations.

Using the best control policy (the initial control policy for the first iteration), the ODE (1) to (15) are integrated from $t_o = 0$ to t_f *N* times with different **u** values for control. *N* **x**-trajectories are generated and the values of x (state variable represented by (1) to (15)) at the beginning of each time stage are stored, so that **x**(k – 1) corresponds to the value of **x** at the beginning of stage k. The next step is beginning the backward integration. The principal routine calls the same integration subroutine as before. Starting at stage *P*, corresponding to time $t_f - L$, for each *N* stored values for **x**(*P* – 1), the integration from $t_f - L$ to t_f is performed. Each of the *R* allowable random values previously calculated for the **u** vector is used for that.

At this point, it was necessary to formulate a subroutine to analyze the results and after an exhaustive comparison among them, store the control values that give the maximum monomer conversion rate value as **u**(*P* – 1), for each of the *N* grid points. In the sequence, the main routine calls the integration routine as before and stepping back to stage (*P* – 1), it performs the integration of ODE (1 to 15) until the stage *P*, corresponding to time $t_f - 2L$ to $t_f - L$. The choice of *R* values for **u**(*P* – 2) and initial state **x**(*P* – 2) as in the previous step, are mandatory to execute these calculations.

The same way as before, an exhaustive comparison among the results for each *N* trajectory permits the maximum values for the monomer conversion rate are stored. This procedure continues until stage 1, corresponding to the initial time $t_o = 0$. At this point, the given initial state is reached and the best control **u**(0) that gives the maximum monomer conversion rate value is stored, completing an iteration.

So, the region **r** for the allowable control is reduced by a γ factor (0.9 for this work) and all the process starting at the first integration task is repeated until the system reaches the convergence to the global optimum. To visualize the results graphically, it was necessary to use after each comparison task, the "save" MATLAB function, to keep stored the best values for the control variables **u** and the state variables **x**. At the end of the iterations, the main routine restore the best stored values by using the "load" and "plot" MATLAB functions, to rearrange and plot the results.

RESULTS AND DISCUSSION

Once finished the algorithm development in MATLAB environment, the simulation runs for the MMA solution polymerization reaction took place by the use of a Pentium IV 2.0 GHz and 512Mb.

It is a fact that the optimization time and the final result for the performance index can be seriously affected by the IDP configuration depending on the chosen of the number of *P* stages, the number *N* of grid points and the allowable values for the **u** control variables [6]-[7]. Concerning this affirmation and keeping in mind the main purpose of this work in solving an OCP in MATLAB through the IDP strategy, a decision was taken. During the runs, the main preoccupation was to choose a fixed number of *P* stages and promote only a few variations on *N* grids and *R* control values to optimize the performance index (monomer conversion rate). This way, in a standard way, it was chosen *P* = 10 stages, *j* = 20 iterations and 2 *Passes* for all the runs.

Besides that, it was defined *R* = 10 and *R* = 20, *N* = 1 and *N* =5, which were combined alternately for the runs. These choices with a few variations on the *R* and *N* variables are justified due to the fact that the main concern at this point was to show that IDP was capable of solving this OCP through the algorithm developed in MATLAB environment, but not in reaching the real global optimum for the

1 Fileti, A. M. F., State University of Campinas, School of Chemical Engineering, CP 6066, CEP13083-970, Campinas, SP, Brazil, frattini@desq.feq.unicamp.br

conversion rate with high precision, which can be tried in future works.

It is also important to mention that the initial conditions for the state and control variables were based on the experimental work trialed by Antunes [12], described as:

$$x = [0.0897, 2.413, 0, 8.833, 0, 0, 0, 0, 0, 0, 2.413, 2.413, 333.2, 298.2, 0] \quad (25)$$

$$Q(0) = 0.12 \qquad (kJ \ / \ kmol) \qquad (26)$$

$$F_{cw}(0) = 8x10^{-6} \qquad (m^3 / s) \qquad (27)$$

After defining all the initial variables, the time chosen to perform the optimization runs also followed the experimental trial mentioned above [12]. The time was defined as $t_f = 350$ min, and then converted into a dimensionless variable τ, assuming values in the interval $0 \leq \tau \leq 1.0$.

Figures 1 and 2 show the results reached by the use of only one *N* grid point alternately with *R* = 10 and *R* = 20.

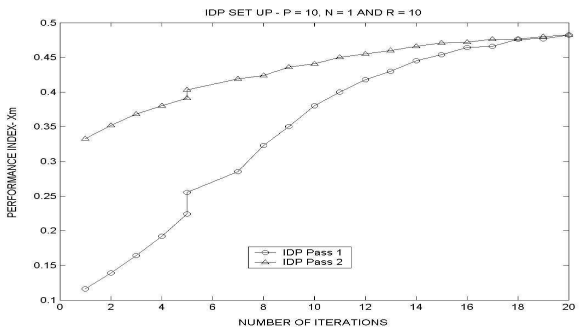

FIGURE 1
MONOMER CONVERSION RATE FOR *N* = 1 AND *R* = 10

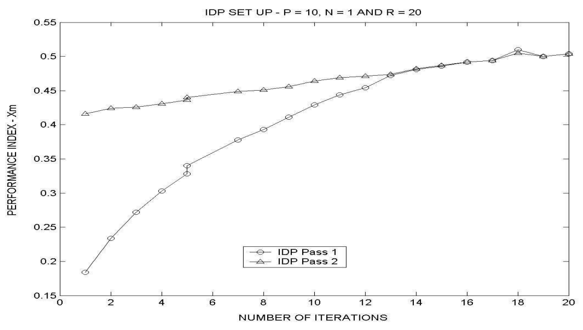

FIGURE 2
MONOMER CONVERSION RATE FOR N = 1 AND R =20

Comparing Figures 1 and 2, although they look very similar, it is possible to notice that there is a better refinement in the optimum search from the Pass 1 to Pass 2 on the Figure 2 than Figure 1. It can be explained by the fact that there is a higher probability of getting the maximum optimum when using *R* = 20 than *R* = 10. The refinement occurs because of the beginning of Pass 2 optimization, which uses the best values stored at the end of Pass 1 as initial condition. Besides, the adoption of a higher number of iterations could contribute to a better refinement on the monomer conversion rate achieved in both cases. However, these results already represent a good accordance with the experimental ones from the trial performed by Antunes [12].

Figures 3 and 4 illustrate the results obtained by the use of N = 5 grid points alternately with R = 10 and R = 20.

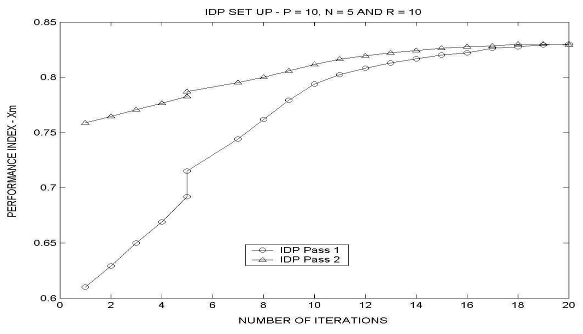

FIGURE 3
MONOMER CONVERSION RATE FOR N = 5 AND R =10

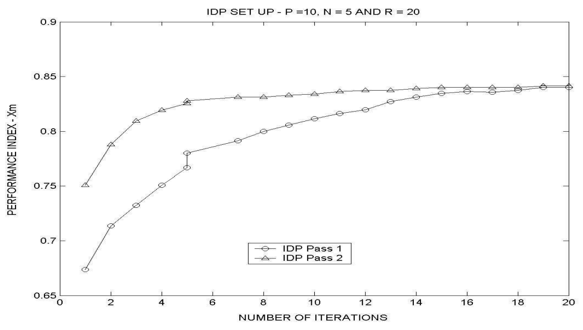

FIGURE 4
MONOMER CONVERSION RATE FOR *N* = 5 AND *R* =20

Similarly to the previous explanation, the refinement in the optimum search also occurs when comparing Figures 3 and 4. Interesting enough was the observation that the monomer conversion rate tended to a global optimum around 0.84, in a satisfactory accordance with the results published by Chakravarthy [10]. Naturally, it indicates the existence of a local maximum around 0.5 as registered in Figures 1 and 2. This transition from a maximum local to a maximum global can be explained by the temperature trajectory calculated during the optimization. It is known from the literature that a higher reactor temperature trajectory leads to a higher monomer conversion rate evidenced by Figures 5 and 6.

A simple analysis on the reactor temperature trajectories permits to find out that the one tending to around 334.5 K (Figure 5) is associated with the monomer conversion rate tending to around 0.5 (Figures 1 and 2), as the same way the temperature tending to around 340 K (Figure 6) is associated with the monomer conversion rate tending to around 0.84 (Figures 3 and 4). Possibly but not necessarily, better adjusted temperature trajectories could be achieved by increasing progressively the values of *N*, *R*, *j* and *Pass* [9]-[10].

In fact, the temperature tendencies taken during the optimization runs in both cases (Figures 5 and 6), were sufficient to reflect quite well the temperature process behavior found by Antunes [12], and Chacravarthy [10], respectively.

1 Fileti, A. M. F., State University of Campinas, School of Chemical Engineering, CP 6066, CEP13083-970, Campinas, SP, Brazil, frattini@desq.feq.unicamp.br

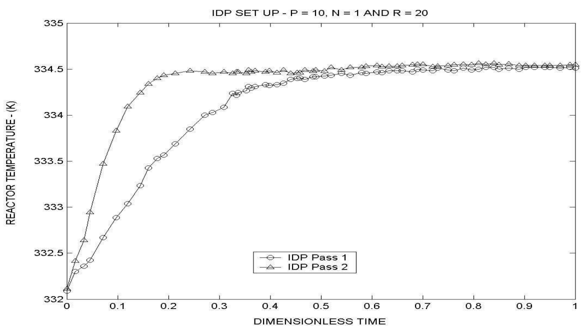

FIGURE 5
REACTOR TEMPERATURE TRAJECTORY FOR *N* = 1 AND *R* =20

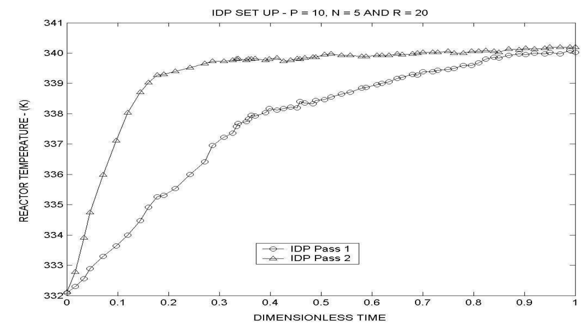

FIGURE 6
REACTOR TEMPERATURE TRAJECTORY FOR N = 5 AND R =20

Another significant result from the optimization runs is concerned with the cooling water flow taken as the control variable for the polymerization process. A careful evaluation on Figures 7 and 8 permits to correlate the cooling water flow stabilized around 17 mL/s (Figure 8), with the higher temperature trajectory and higher monomer conversion rate (Figures 6 and 4 or 3, respectively), as the same way the cooling water flow around 10 mL/s is correlated with lower temperature trajectory and lower monomer conversion rate (Figures 5 and 2 or 1, respectively).

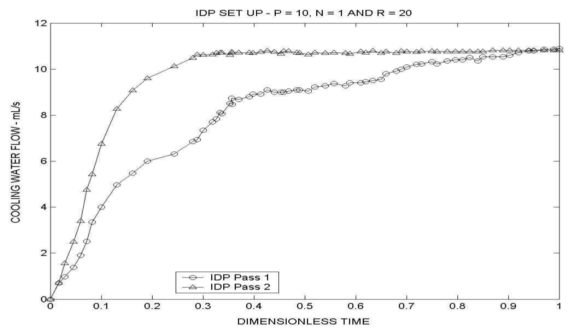

FIGURE 7
COOLING WATER FLOW TRAJECTORY FOR *N* = 1 AND *R* =20

This is quite comprehensible since the reactor temperature is limited to 340 K and naturally, the cooling water flow must be enough (17 mL/s) to keep it bellow this limit. On the other hand, clearly for a temperature of 334 K a lower cooling water flow is enough (10 mL/s).

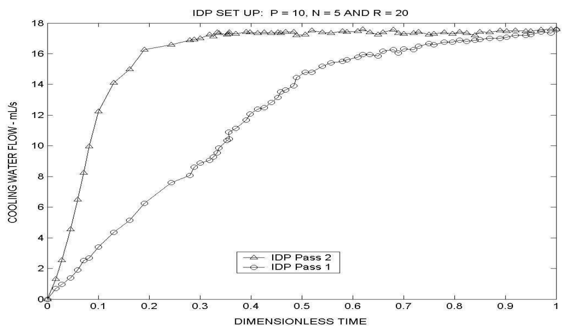

FIGURE 8
COOLING WATER FLOW TRAJECTORY FOR *N* = 1 AND *R* =20

Another relevant information is the optimization time required to reach the convergence, which can vary substantially depending on the chosen values for *N*, *R*, *P*, *j* and *Pass* among others variables, used to set up IDP. Table I illustrates the effects of the variables *N* and *R* on the optimization time and the performance index (monomer conversion rate).

TABLE I
EFFECTS OF THE VARIABLES N AND R ON OPTIMIZATION PROCESS

Number of *N* grids	Number of *R* values	Number of *P* stages	CPU spent time (min)	Performance index (X_m)
1	10	10	20,78	0,445
1	20	10	35,35	0,495
5	10	10	78,67	0,830
5	20	10	120,38	0,842

It is clear to see that larger values of *N* and *R* variables affect substantially the time required for reaching the global optimum and consequently, the performance index precision. It means that to achieve a higher precision on the performance index, many other variations on *N*, *R* and even *P* among others, should be taken for new optimization runs.

Additionally, these results indicate that IDP is a robust methodology in searching the global optimum even to systems of high dimensionality and non-linearity as the MMA polymerization process. At the same time, it can demand a very high computational cost, mainly when compared to other methods as Sequential Quadratic Programming (SQP) for example, considered fast in reaching the convergence but deficient in robustness [13]. In despite of that, the developed algorithm in MATLAB environment is quite feasible to demonstrate chemical process behavior such as batch polymerization reactors, through the IDP strategy.

CONCLUSIONS

The present work is concerned with the use of MATLAB as an important tool for the Chemical Engineering Education, in order to demonstrate the behavior of chemical processes. It was successfully developed a specific algorithm to perform an off line optimization for the MMA solution polymerization reaction by using the IDP technique. As performance index it was chosen the monomer conversion rate, which was maximized by the manipulation of the control variable defined as cooling water flow.

1 Fileti, A. M. F., State University of Campinas, School of Chemical Engineering, CP 6066, CEP13083-970, Campinas, SP, Brazil, frattini@desq.feq.unicamp.br

The results reached during the optimization runs were satisfactorily in accordance with the literature data, validating the used mathematical modeling [9]-[10], and consequently, the algorithm developed in MATLAB environment. Thus, this IDP algorithm can be considered suitable to optimize off line OCP for high dimensionality and non linearity problems, such as MMA polymerization reaction, needing only a few adaptations for other applications.

While Microsoft Excel Solver is extensively used in undergraduate courses, such as "Process Analysis and Simulation" and "Chemical Processes Optimization", MATLAB software is more suitable for graduate students in order to solve more complex engineering problems. The development of an IDP computational program in MATLAB helps engineering students to see this software as another feasible tool, mainly due to its simplicity of use, practicability of programming and its graphical interface that permits to observe an optimization process step by step. More emphasis on modeling and less on the teaching of optimization algorithms are to be achieved by using both tools. Widening real-world examples and case analysis may increase the computing role in the engineering courses and make the classes even more interesting.

Considering the relevance and power of IDP, all engineers and graduate students who are involved in solving OCP problems should be familiarized with this technique [5]. Because of this, it is highly recommended the development of an IDP MATLAB Standard Toolbox. It certainly would bring a significant contribution to make MATLAB even more popular among IDP users and overcome the need of creating specific algorithms in MATLAB environment.

NOMENCLATURE

A, A_∞	Reactor-jacket and surrounding-jacket heat transfer surface areas, m^2
c, c_w	heat capacity of reacting mixture and water, kJ/kg.K
F_{cw}	inlet flow rate of cooling water, m^3/s
k	rate constants for the reactions (equations 1 to 15) at any time t (s^{-1} or m^3/mol.s)
I	concentration of initiator, kmol/m^3
M	concentration of monomer, kmol/m^3
N	number of grid points for the state variable **x**
P	number of stages
Q	water heating, kJ/s
R	allowable random numbers for control
R	concentration of primary radical, kmol/m^3
S	concentration of solvent, kmol/m^3
t_o, t_f	initial and final times, s
T	reactor temperature, K
T_{cw}	temperature of inlet cooling water, K
T_j, T_∞	jacket and room temperatures, respectively, K
u	control variable vector
V_l	volume of reacting mixture, m^3
U, U_∞	overall heat-transfer coefficients of reactor-jacket and jacket-surrounding, respectively, kJ/m^2.s.K
x, X_m	state variable and monomer conversion rate
ΔH_p	heat of propagation reactions, kJ/kmol
λ_i, μ_i	ith (i=0,1,2) moment of live and dead polymer radicals, respectively, kmol/m^3
ρ, ρ_{cw}	density of reacting mixture and water, kg/m^3
ζ_m, ζ_{m1}	net monomer added to the reactor [10]

REFERENCES

[1] Okoro, O. I., Govender, P., Chikuni, E, *A new User-Friendly Software for Teaching and Research in Engineering Education*, The Pacific Journal of Science and Technology, Vol 7, No 2, 2006, pp. 130-136.

[2] Dabney, B. J., Ghorbel, F. H., *Enhancing an Advanced Engineering Mechanics Course Using MATLAB and Simulink*, International Journal of Engineering Education, Vol 21, No 5, 2005, pp. 885-895.

[3] Azemi, A., Yaz, E. E., *Using MATLAB a in Graduate Electrical Engineering Optimal Control Course*, Proceedings of the 27th Frontier in Education Conference, Pittsburgh: PA 1997, pp. 13-17.

[4] Edgar, T. F., *Chemical Engineering Education and the Three C´s: Computing, Communication, and Collaboration*, AIChE Annual Meeting, Session: Chemical Engineering Issues of the New Millennium: Beyond Vision 2020, Los Angeles, 2000, pp. 12-17.

[5] Chen, Y. Q., *Book Review for submission to The International Journal of Robust and Nonlinear Control*, 2000. Available: http://www.csois. usu.edu/ index.php.

[6] Luus, R., Optimal *Control by Dynamic Programming Using Systematic Reduction in Grid Size*, International Journal of Control, Vol 19, 1990, pp. 144-151.

[7] Luus, R., *Iterative Dynamic Programming*, Florida: Chapman & Hall/CRC, 2000.

[8] Ekpo, E. E., Mujtaba, I. M., *Optimal Control Trajectories for a Batch Polymerization Reactor*, International Journal of Chemical Reactor Engineering, Vol 5, Article A4, 2007.

[9] Seth, V., Gupta, S. K., *An Experimental Study on Bulk and Solution Polymerization of Methyl Methacrylate with Responses to Steps Changes in Temperature*, Journal of Polymer Engineering, Vol 15, 1995, pp. 283.

[10] Chakravarthy, S. S. S., Saraf, D. N., Gupta, S. K., *Use of Genetic Algorithms in the Optimization of Free Radical Polymerization Exhibiting the Trommsdorf Effect*, Journal of Applied Polymer Science, Vol 63, 1997, pp. 529-548.

[11] Mathworks, Inc. *Matlab User's Guide*, Natick, 2005, NJ.

[12] Antunes, A. J. B., Pereira, J. A. F. R., Fileti, A. M. F., *Fuzzy Control of a PMMA Batch Reactor: Development and Experimental Testing*, Computers and Chemical Engineering, Vol 30, 2005, pp. 268-276.

[13] Schröder, A., Mendes, M. J., *Time Optimal Control of Distillation Columns: a Mixed IDP-SQP Approach*. Computer and Chemical Engineering Supplement, 1999, pp. S491-S494.

1 Fileti, A. M. F., State University of Campinas, School of Chemical Engineering, CP 6066, CEP13083-970, Campinas, SP, Brazil, frattini@desq.feq.unicamp.br