AN ALTERNATIVE METHOD FOR THE DESIGN OF TIME-VARYING FEEDBACK CONTROL SYSTEMS

MÉTODO ALTERNATIVO PARA EL DISEÑO DE SISTEMAS DE CONTROL RETROALIMENTADOS VARIANTES EN EL TIEMPO

CARLOS GÓMEZ

Electronics Engineer, Universidad Industrial de Santander, car_cl3@hotmail.com

IVÁN AMAYA

Mechatronics Engineer, Ph.D.(c), Universidad Industrial de Santander, iamaya2@gmail.com

RODRIGO CORREA

Professor, Ph.D., Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones, Universidad Industrial de Santander, crcorrea@uis.edu.co

Received for review June 4 th, 2012, accepted October 9th, 2012, final version October, 16 th, 2012

ABSTRACT: This article proposes to use a numeric strategy based on a discrete particle swarm optimization algorithm, to solve a problem related to compensator design in a time-varying feedback control system. At first, it is shown why it is possible to transform a problem of solving a system of linear Diophantine equations, into an optimization one. Some exemplary problems are shown. High quality solutions, i.e. in terms of accuracy and precision, were achieved in relatively short computation times.

KEYWORDS: Particle swarm optimization, compensators, time-varying feedback control, optimization.

RESUMEN: El presente artículo propone utilizar una estrategia numérica basada en un algoritmo de optimización metaheurístico de enjambre de partículas discreto, para resolver un problema de diseño de compensadores en sistemas retroalimentados variantes en el tiempo. Se demuestra inicialmente como se puede convertir el problema de solución del sistema de ecuaciones Diofánticas lineales resultante en la solución de un problema de optimización. Se desarrollan ejemplos demostrativos que ilustran la idea principal. Se lograron soluciones de excelente calidad en cuanto a precisión y exactitud en tiempos de computación relativamente breves.

PALABRAS CLAVE: Enjambre de partículas, compensadores, control realimentado variante en el tiempo, optimización.

1. INTRODUCTION

The study and analysis of Diophantine equations and their solution approaches, has been an open problem for both, mathematicians and engineers, who strive to solve them for a given application. Since the appearance of Hilbert's tenth problem, the literature reports several attempts at proving that it is unsolvable, *i.e.,* that there is no algorithm that determines if a given polynomial, with integer coefficients has roots in the integer domain [1], [2]. The time and effort of the researchers has paid off: it does not exist. However, instead of meaning the end of a line, it has reoriented the question, especially in engineering, to determine if there are other solution approaches, that without being generic, allow to solve, somehow, "any" Diophantine equation. Here, several proposals can be included, by no means being exhaustive, which are based, for example, on the concepts of state spaces, of inversion in said spaces and of polynomial approximation, as well as on algebraic and geometric studies and on the extension of the well known Euclidean algorithm for the analysis of data dependence, among several others [3–7]. Likewise, some metaheuristics have been used, such as genetic algorithms, simulated annealing, particle swarm optimization and evolutionary computing [8–10]. The purpose of this paper is to demonstrate how to solve the Diophantine equation obtained during the design process of compensators by an optimization strategy, such as the discrete particle swarm optimization algorithm. Some examples are presented to illustrate the procedure.

2. Mathematical Foundation

2.1. Solving a Diophantine equation

In principle, a Diophantine problem is the solution of an equation, or a system of equations, in the integer

($\mathbb{Z}$) or rational ($\mathbb{Q}$) domains, or their generalizations, such as rings generated over $\mathbb{Z}$, or fields over $\mathbb{Q}$. Consequently, a Diophantine equation has the form $F(x_1, x_2, x_3 \dots, x_n) = 0$ where F is a polynomial with integer coefficients and whose solution is restricted, in most cases, to the non-negative integers. A linear Diophantine equation, with n unknowns, is defined by eq. (1), where $a_1, a_2, \dots, a_n$ are known rational, or integer, numbers, and $x_1, x_2, \dots, x_n$ are unkı vns, i.e., the numbers that should satisfy them, [11]; bb is a known integer.

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b \tag{1}$$

This type of equation is, generally, undetermined and with more than one unknown. It is important to remark that it only has a solution if the greatest common divisor (g.c.d.) of the a_i coefficients, is a divisor of b. In the case of two unknowns, for example, an equation as the one shown by eq. (2) appears, where (a, b, c) are known integers, and the solution only exists if the g.c.d. of a and b is a divisor of c.

$$a * x + b * y = c \tag{2}$$

Thus, a general solution can be written, as shown by eq. (3), where β is an integer, d is an integer which represents the g.c.d. and (x_0, y_0) are two particular solutions.

$$\begin{aligned} x &= x_0 + \beta * \frac{b}{d} \\ y &= y_0 - \beta * \frac{a}{d} \end{aligned} \tag{3}$$

The traditional approach for this case, even though others are available, is based on the famous extended Euclid algorithm [11], [12].

2.2. System of linear Diophantine equations

Consider the system of linear Diophantine equations over $\mathbb{Z}[X_1, \cdots, X_n]$, given by eq. (4).

$$\begin{gathered} f_1 = A_{11}X_1 + A_{12}X_2 + \cdots + A_{1n}X_n - C_1 = 0 \\ f_2 = A_{21}X_1 + A_{22}X_2 + \cdots + A_{2n}X_n - C_2 = 0 \\ \vdots \\ f_i = A_{i1}X_1 + A_{i2}X_2 + \cdots + A_{in}X_n - C_i = 0 \\ \vdots \\ f_n = A_{n1}X_1 + A_{n2}X_2 + \cdots + A_{nn}X_n - C_n = 0 \end{gathered} \tag{4}$$

It is evident that if the solution exists in $\mathbb{Z}$, it must also exist in $\mathbb{R}$. This means that it must comply with the Roche-Frobenius theorem, which ultimately poses that:

range **A** = *range* (***A;C***)

where **A** is the coefficient matrix of the system, and *(A;C)* is the expanded one. Therefore, in order to find the solutions in the integer or rational set, each f_i function must comply such that its g.c.d.:

$$g.c.d.(A_{i1}, A_{i2}, \cdots, A_{in}) | C_i \ \forall\ i = 1, 2, \cdots, n$$

This way, complying with these two requirements is a sufficient condition for the system to have a solution.

2.3. Diophantine equations in control systems

Diophantine equations appear in the design and synthesis of feedback compensators. According to Kucera's *review,* the Diophantine equation approach is a transfer function based control theory in which the transfer functions are viewed and handled as algebraic objects [13–16].While conceived for linear finite dimensional time invariant systems, it was generalized and currently includes some time varying infinite dimensional and non linear systems. The approach, in general, is based on the factorization of transfer functions over an appropriate ring, thus reducing the mathematical synthesis of control systems to the solution of linear Diophantine equations in that ring. More recently, Wu [17], developed a systematic approach for solving both linear time-varying and time-invariant Diophantine equations. The pole-placement in a closed-loop structure using output feedback can be done by means of solving those equations. The poles of the overall transfer function are assigned in order to meet some given performance requirements. The approach is based on successively reducing the order of the Diophantine equation using the well known Euclidean algorithm. A complete explanation of this strategy is not included here. It can be found in [17].

2.4. Systems of equations and optimization

Consider the equation given in (1). Let $f\colon \mathbb{X} \to \mathbb{R}$ be the function defined by:

$$f(x) = \sum_{i=1}^{m} f_i(x)^2 \; ; \; x \in \mathbb{X} \qquad (5)$$

$Theorem$ 1 Suppose that the eq. (1) has a solution on $\mathbb{X}$, and let $a \in \mathbb{X}$. Therefore:

$a \in \mathbb{X}^*$ if, and only if, a minimizes f, defined by (5).

$Proof$. If $a \in \mathbb{X}^*$ then $f_i(a) = 0$ for each $i = 1, \dots, m$. Thus, $f(a) = 0$ and, since $f(x) \geq 0$ for every $x \in \mathbb{X}$, then a is a minimum for f. Now, if a minimizes f but it does not satisfy equation (1) then $f(a)$ must be positive, since $f(x) \geq 0$ for every $x \in \mathbb{X}$. Since the system has a solution over $\mathbb{X}$, there exists $x^* \in \mathbb{X}^*$ such that $f(x^*) = 0$ and $x^* \neq a$. Therefore, $f(x^*) < f(a)$ which shows a is not a minimum for f. According to the previous statements, a problem of finding the roots of a system of non-linear equations, over a given set $\mathbb{X}$, can be transformed into an optimization one (minimization for the current case), with an objective function f built in the way shown by equation (5) over the same domain (i.e. $\mathbb{X}$). It is important to remark that this set can be of any nature, as long as it is not empty. In the case of finding the roots of a Diophantine system, the set $\mathbb{X} \subseteq \mathbb{R}^n$ must guarantee that $\mathbb{X} \cap \mathbb{Z}^n \neq \emptyset$, with $\mathbb{Z}$ being the set of integer numbers. In other words, $\mathbb{X}$ must contain points whose coordinates are integers. Another application of this theorem can be seen in [18], [19].

2.5. The algorithm

The implemented algorithm is built up from various interconnected blocks and is similar to the structure of traditional PSO (for real numbers), [20]. A first stage is given by the random assignation of a swarm of user defined integers. Any size can be used here. Likewise, the definition of these values is subject to previous knowledge of the objective function (*fitness*), as well as to the presence of restrictions. Moreover, an initial velocity of zero can be defined for the particles. After that, the algorithm evaluates, in the given search space, the objective function. With it, local and global best values are established, and both, the velocity and position, of each particle, are reevaluated as shown below. This procedure is iterative and is repeated until the convergence criteria are met, or until all solutions in the search domain are found. An algorithm, considered as a variant of the traditional PSO, was used during this research. In the same fashion as PSO, its version for discrete solutions includes two vectors $\mathrm{X_i}$ and $\mathrm{V_i}$, related to the position and velocity of each particle, for every iteration. The first one is a vector of random numbers, initially, in a valid solution interval. The second one can also be a random vector, but it can be assumed to be zero for the first iteration, in order to simplify the algorithm. When the problems become multidimensional, the vectors transform into position and velocity matrices, since there is a value for each unknown. Discrete PSO differs from its traditional version in which the new velocity and position depend on both, an equation and a decision rule, which chooses between the local and global best values for the next iteration. Assuming there is a vector $y_i = (y_{i1}, y_{i2}, \cdots, y_{1n})$ that allows the transition between continuous and discrete PSO, and which takes the value of (-1, 1, or, 0) according to eq. (6), where glo is the global optimum of the swarm, and loc the local one, [20].

$$y_i = \begin{cases} 1 & \text{if} \quad \mathrm{X_i} = glo \\ -1 & \text{if} \quad \mathrm{X_i} = loc \\ 0 & \text{if} \quad \mathrm{X_i} \neq glo \neq loc \\ -1 \; o \; 1 & \text{if} \quad \mathrm{X_i} = glo = loc \end{cases} \qquad (6)$$

Afterwards, velocity is updated according to eq. (7).

$$\begin{aligned} \mathrm{V_{i+1}} = \mathrm{V_i} * \mathrm{w} + \mathrm{c_1} * \mathrm{r_1} * (-1 - \mathrm{y_i}) \\ + \mathrm{c_2} * \mathrm{r_2} * (1 - \mathrm{y_i}) \end{aligned} \qquad (7)$$

Then, the decision parameter, vector $B_i = (B_{i1}, B_{i2}, \cdots, B_{in})$, is calculated according to eq. (8).

$$B_i = y_i + \mathrm{V_{i+1}} \qquad (8)$$

This parameter decides if the next position of the particle is chosen as the local or global optimum, or if it is chosen as a random number in the search domain. Thus, position updating is performed according to eq. (9), where α is a constant that defines the intensification (new position equal to the local or global optimum) or diversification (new position equal to a random number), [9].

$$X_{i+1} = \begin{Bmatrix} glo & \text{if} & B_i > \alpha \\ loc & \text{if} & B_i < -\alpha \\ \text{rand int} & \text{if} & -\alpha \leq B_i \leq \alpha \end{Bmatrix} \qquad (9)$$

3. Results and Analysis

This section shows some of the results achieved after solving two examples, which illustrate the suggested procedure. A computer with an AMD Turion X2 Dual Core RM-72 processor, at 2.1 GHz, and with 4 GB of

RAM memory, was used. For the two examples, the following parameters were used: w = 0.75, $C_1 = 0.9$, $C_2 = 0.2$, and, $\alpha = 0.5$. These values were chosen based on some preliminary tests and on the information available in the literature [9], [20].

4.1. Example 1

As a starting point, the same example presented by Wu in [17] was used. If one wants the control system to have certain modes, that is, to have its finite poles located at certain positions, it is necessary to compensate the given plant so that the closed-loop system has a pre-specified characteristic polynomial. Thus, the closed-loop control system has unitary feedback. It is required to define the compensator, $C(s)$, as a polynomial which satisfies the condition that six poles are located at -1. Let the plant be:

$$\mathrm{G(s)} = \frac{s^2 + s + 1}{s^3 + 3s + 4s + 3}$$

In the frequency domain, s , the closed-loop transfer function is given by eq. (10),

$$\frac{Y(s)}{R(s)} = \frac{C(s) * G(s)}{1 + C(s) * G(s)} \qquad (10)$$

where, $Y(s)$ is the output, $R(s)$ is the set point and $C(s)$ and $G(s)$ are defined as,

$$G(s) = \frac{B(s)}{A(s)} \qquad (11)$$

$$C(s) = \frac{N(s)}{D(s)} \qquad (12)$$

After rearranging,

$$\frac{Y(s)}{R(s)} = \frac{B(s) * N(s)}{B(s) * N(s) + A(s) * D(s)} \qquad (13)$$

And thus, a Diophantine equation can be established as,

$$D(s) * A(s) + N(s) * B(s) = F(s) \qquad (14)$$

If there are six poles at $s = -1$, it can be found that $F(s)$ is,

$$\begin{aligned} F(s) &= (s+1)^6 \\ &= s^6 + 6s^5 + 15s^4 + 20s^3 + 15s^2 + 6s + 1 \end{aligned} \qquad (15)$$

Considering the information provided in [17], it is known that:

$$A(s) = s^3 + 3s^2 + 4s + 3 \qquad (16)$$

$$B(s) = s^2 + s + 1 \qquad (17)$$

In this example, it is then required to solve the Diophantine equation (14) and find $D(s)$ and $N(s)$, given by,

$$D(s) = X_1 s^3 + X_2 s^2 + X_3 s + X_4 \qquad (18)$$

$$N(s) = X_5 s^2 + X_6 s + X_7 \qquad (19)$$

The solution, with the modified Euclid algorithm as appears in [17], is:

$$X_1 = 1, X_2 = 3, X_3 = 2, X_4 = 2, X_5 = 0, X_6 = -3, X_7 = -5$$

Using this solution set, the controller given by eq. (20) can be implemented.

$$C(S) = \frac{-3 * s - 5}{s^3 + 3 * s^2 + 2 * s + 2} \qquad (20)$$

Now, solving the current example with the proposed algorithm, equations (15) to (19) are substituted into the Diophantine equation (eq. (14)), so it is found that,

$$\begin{aligned} &(X_1) * s^6 + (3X_1 + X_2) * s^5 \\ &+(X_1 + 3X_2 + X_3 + X_5) * s^4 \\ &+\begin{pmatrix} 3X_1 + 4X_2 + 3X_3 + X_4 \\ +X_5 + X_6 \end{pmatrix} * s^3 \\ &+\begin{pmatrix} 3X_2 + 4X_3 + 3X_4 + X_5 \\ +X_6 + X_7 \end{pmatrix} * s^2 \\ &+(3X_3 + 4X_4 + X_6 + X_7) * s \\ &+(3X_4 + X_7) \\ &= s^6 + 6s^5 + 15s^4 + 20s^3 \\ &+15s^2 + 6s + 1 \end{aligned} \qquad (21)$$

Parting from eq. (21), the following system of Diophantine equations can be established:

$$\begin{aligned} &X_1 - 1 = 0 \\ &3X_1 + X_2 - 6 = 0 \\ &4X_1 + 3X_2 + X_3 + X_5 - 15 = 0 \\ &3X_1 + 4X_2 + 3X_3 + X_4 + X_5 + X_6 - 20 = 0 \\ &3X_2 + 4X_3 + 3X_4 + X_5 + X_6 + X_7 - 15 = 0 \\ &3X_3 + 4X_4 + X_6 + X_7 - 6 = 0 \\ &3X_4 + X_7 - 1 = 0 \end{aligned} \qquad (22)$$

Thus, the objective function given by eq. (23) can be built.

$$\begin{aligned} F(x) &= (X_1 - 1)^2 + (3X_1 + X_2 - 6)^2 \\ &+(4X_1 + 3X_2 + X_3 + X_5 - 15)^2 \\ &+(3X_1 + 4X_2 + 3X_3 + X_4 + X_5 + X_6 - 20)^2 \\ &+(3X_2 + 4X_3 + 3X_4 + X_5 + X_6 + X_7 - 15)^2 \\ &+(3X_3 + 4X_4 + X_6 + X_7 - 6)^2 + (3X_4 + X_7 - 1)^2 \end{aligned} \quad (23)$$

With it, several simulations were run. This article reports, as an example, a set of 20 algorithm runs. It was found that for each repetition, the same answer was found, even though in different run times and number of iterations. This means that the results are of excellent quality, in terms of accuracy and precision. The minimum convergence time was 102.080 s, for 5961 iterations, whilst the highest one was of 445.080 s, for 22805 iterations. The average time was of 226.575 s, with 12173 iterations on average. Figure 1 shows an almost linear relation between convergence time and number of iterations.

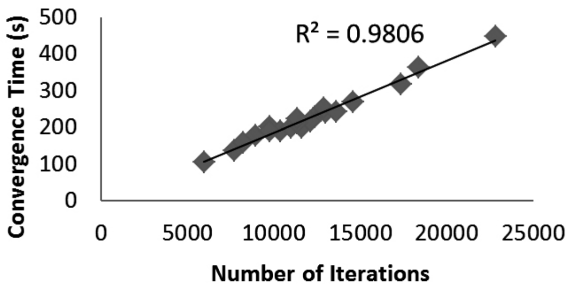

Figure 1. Relationship between the convergence time and the number of iterations

4.2. Example 2

Considering a closed loop control system with unity feedback, it is required to find the controller, $C(s)$, with the polynomial method, guaranteeing stability and an establishing time below 10 s. The closed-loop dominant pole is located at $s = 0.4016$. Thus, the plant is:

$$G(s) = \frac{10s+3}{(s+1)*(s+2)*(s+8)}.$$

The closed-loop transfer function is given by eq. (24). If there is a dominant pole at $s = 0.4016$, then,

$$F(s) = 3s^6 + 37s^5 + 125s^4 + 196s^3 + 166s^2 + 97s + 22.$$

$$\frac{Y(s)}{X(s)} = \frac{C(s)*G(s)}{1 + C(s)*G(s)} \quad (24)$$

As in the previous example, the Diophantine equation given by (25) is derived. Its solution was found by expressing $C(s)$ as a function of $N(s)$ and $D(s)$. Therefore, now the problem resides in finding the values of $X_1, X_2, \dots, X_7$ that satisfy $F(s)$.

$$D(s)*A(s) + N(s)*B(s) = F(s) \quad (25)$$

$$C(s) = \frac{X_5 s^2 + X_6 s + X_7}{X_1 s^3 + X_2 s^2 + X_3 s + X_4} \quad (26)$$

Using Euclid's algorithm as proposed in [17], the solution was found to be:

$$X_1 = 3, X_2 = 4, X_3 = 3, X_4 = 1$$

$$X_5 = 1, X_6 = 1, X_7 = 2$$

In order to solve the Diophantine equation (25) with the PSO discrete algorithm, it must be first expanded, generating:

$$\begin{aligned} &(X_1 s^3 + X_2 s^2 + X_3 s + X_4) * \binom{s^3 + 3s^2}{+4s + 3} \\ &+(X_5 s^2 + X_6 s^2 + X_7 s + X_8) * (s^2 + s + 1\ ) \\ &= 3s^6 + 37s^5 + 125s^4 + 196s^3 \\ &+166s^2 + 97s + 22 \end{aligned} \quad (27)$$

which leads to the system of Diophantine equations given by (28).

$$\begin{aligned} &X_1 - 3 = 0 \\ &11X_1 + X_2 - 37 = 0 \\ &26X_1 + 11X_2 + X_3 - 125 = 0 \\ &16X_1 + 26X_2 + 11X_3 + X_4 + 10X_5 - 196 = 0 \\ &16X_2 + 26X_3 + 11X_4 + 3X_5 + 10X_6 - 166 = 0 \\ &16X_3 + 26X_4 + 3X_6 + 10X_7 - 97 = 0 \\ &16X_4 + 3X_7 - 22 = 0 \end{aligned} \quad (28)$$

Now, the objective function can be constructed as:

$$\begin{aligned} F(x) &= (X_1 - 3)^2 + (11X_1 + X_2 - 37)^2 \\ &+(26X_1 + 11X_2 + X_3 - 125)^2 \\ &+\binom{16X_1 + 26X_2 + 11X_3 + X_4 + 10X_5}{-196}^2 \\ &+\binom{16X_2 + 26X_3 + 11X_4 + 3X_5 + 10X_6}{-166}^2 \\ &+(16X_3 + 26X_4 + 3X_6 + 10X_7 - 97)^2 \\ &+(16X_4 + 3X_7 - 22)^2 \end{aligned} \quad (29)$$

Several simulations were run, from which 20 of them are reported in this article. Once again, it was found that the same answer was always reported, with different run times and number of iterations. Thus, an excellent answer quality was also found in this example, which corresponds to the one provided by Euclid's algorithm. The minimum convergence time was 145.032 s for 7001 iterations, while the maximum was of 436.083 s for 19386 iterations. The average run time was 281.263 s with an average of 12774 iterations. Figure 2 shows the step response of the plant and the controller. It is important to remark that it is stable and that it has an establishing time lower than 10 s, complying with the initial requirements.

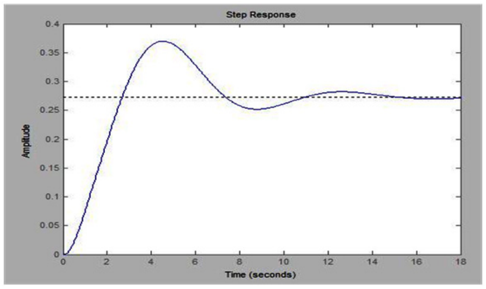

Figure 2. Step response of the controlled system, where the x-axis represents time [s] and the y-axis represents the step response amplitude.

5. CONCLUSIONS

Based on two illustrative examples, it was shown how a discrete PSO algorithm can be used for the design of compensators in a time-varying feedback control system. It seems plausible to assume that this optimization approach is valid, at least for these examples shown here. The main inconvenience found using this strategy is the algorithm's use of random parameters, which has a definite impact on the computational time. On the other hand, the main advantage, in our opinion, is its simplicity and its straightforward programming.

REFERENCES

[1] Davis, M., Putnam, H. and Robinson, J., The Decision Problem for Exponential Diophantine Equations, The Annals of Mathematics, 74, 3, pp. 425–436, 1961.

[2] Matiyasevich, Y. V., Hilbert's Tenth Problem. MIT Press, 1993.

[3] Fang, C.-H., A Simple Approach to Solving the Diophantine Equation, IEEE Transactions on Automatic Control, 37, 1, pp. 152–155, 1992.

[4] Bonilla, E. M., Figueroa, G. M. and Malabare, M., Solving the Diophantine Equation by State Space Inversion Techniques : An Illustrative Example, Proceedings of the 2006 American Control Conference, pp. 3731–3736, 2006.

[5] Jones, L., A Polynomial Approach to a Diophantine Problem, Mathematics Magazine, 72, 1, pp. 52–55, 1999.

[6] Mvondo, E. C. B., Cherruault, Y. and Mazza, J.C., Computational resolution of Diophantine equations by means of alpha-dense curves, Kybernetes, 41, 1/2, pp. 51–67, 2012.

[7] Grosslinger, A. and Schuster, S., On Computing Solutions of Linear Diophantine Equations with One Non-linear Parameter, 2008 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pp. 69–76, 2008.

[8] Abraham, S. and Sanglikar, M., Finding Numerical Solution to a Diophantine Equation: Simulated Annealing as a Viable Search Strategy, Proceedings of the International Conference on Mathematical Sciences, 2, pp. 703–712, 2008.

[9] Abraham, S., Sanyal, S. and Sanglikar, M., Particle Swarm Optimization Based Diophantine Equation Solver, ArXiv, pp. 1–15, Mar. 2010.

[10] Smart, N., The Algorithmic Resolution of Diophantine Equations. Cambridge University Press, , 1998.

[11] Contejean, E., An Efficient Incremental Algorithm for Solving Systems of Linear Diophantine Equations, Information and Compuation, 113, pp. 143–172, 1994.

[12] Hanta, V., Solution of Simple Diophantine Equations by Means of Matlab. [Online]. Available: http://dsp.vscht.cz/konference_matlab/matlab02/hanta.pdf. [Accessed: 19-Apr-2012].

[13] Kučera, V., Diophantine equations in control—A survey, Automatica, 29, 6, pp. 1361–1375, Nov. 1993.

[14] Dostal, P., Bobal, V. and Tomastik, M., Application of polynomial method in control of time delay systems, 2004 IEEE International Symposium on Computer Aided Control Systems Design, Taipei, pp. 89–94, 2004.

[15] Matusu, R., Prokop, R. and Dlapa, M., Robust control of temperature in hot-air tunnel, 16th Mediterranean Conference on Control and Automation, pp. 576–581, Jun. 2008.

[16] Kong, H., Zhou, B. and Zhang, M., A Stein equation approach for solutions to the Diophantine equations, 2010 Chinese Control and Decision Conference, Xuzhou, pp. 3024–3028, 2010, no. 1.

[17] Wu, S.-H., Time-varying feedback systems design via Diophantine equation order reduction, Ph.D. Dissertation, The University of Texas at Arlington, 2007.

[18] Amaya, I., Cruz, J. and Correa, R., Real Roots of Nonlinear Systems of Equations Through a Metaheuristic Algorithm, Revista Dyna, 78, 170, pp. 15–23, 2011.

[19] Amaya, I., Cruz, J. and Correa, R., Solution of the Mathematical Model of a Nonlinear Direct Current Circuit Using Particle Swarm Optimization, Revista Dyna, 79, 172, pp. 77–84, 2012.

[20] Jarboui, B., Damak, N., Siarry, P. and Rebai, A., A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems, Applied Mathematics and Computation, 195, 1, pp. 299–308, Jan. 2008.