Algebraic optimal control in R_{MS} ring: A case study

Libor Pekař and Roman Prokop

Abstract**—**In [1], [2] some basic and detailed ideas, respectively, of algebraic controller design in the ring or quasipolynomial meromorphic functions (R_{MS}) with optimal controller tuning via spectral abscissa minimization using some advanced iterative algorithms were presented and introduced. The aim of this paper is to follow with these theoretic contributions in order to examine and verify the usability and applicability of the whole methodology. A case study of controller design, tuning and simulation of a mathematical model of a real-life unstable time-delay system (TDS), namely, the roller skater on the controlled swaying bow is presented. Four introduced iterative optimization algorithms are tested and benchmarked.

Keywords**—**Time-Delay Systems, Optimization, Spectral Abscissa, Pole Placement, Iterative Algorithms, Artificial Intelligence.

I. Introduction

OPTIMIZATION became a necessary part of managing many technical, industrial and even everyday processes and activities. It became naturalized also in control theory; especially while controller design and tuning. Obviously, the more complex the controlled object or its model is the more advanced optimization method ought to be utilized.

Time delay systems (TDS) and models can be found in many theoretical and practical applications covering various fields of human activity, such as technology, informatics, biology, economy etc for decades. Already Volterra in [3] did formulate differential equations incorporating the past states when he was studying predator-pray models. Examples of other models can be found e.g. in [4]-[7]. The well-known fact is that TDS has mostly a complex structure and they are not easy to control by a standard means [8]. In [9] modern, algebraic controller design in the ring of proper and stable meromorphic functions (R_{MS}) has been introduced. This conception have been then extended e.g. in [10], [11].

In [1], [2] optimal pole placement for TDS by means of controller structure design in R_{MS} and via the spectral abscissa minimization using four advanced iterative algorithms was introduced. Namely, the Quasi-Continuous Shifting Algorithms (QCSA) [12]–[13], the (Extended) Gradient Sampling Algorithm (EGSA) [14]-[16], the Nelder-Mead algorithm (NM) [17] the Self-Organizing Migration Algorithm (SOMA) [18] were described in these papers to find optimal controller parameters.

In this part of the contribution, a mathematical model of an attractive real-life system with internal delays serves as a benchmark for the optimization algorithms. It describes unstable system with the roller skater on a controlled swaying bow, according model presented e.g. in [10]. An infinite dimensional control system is achieved using a simple feedback loop and via algebraic approach in R_{MS}. Pole assignment strategy minimizing the spectral abscissa is then used as a tuning procedure to set free controller parameters suitably. The efficiency of the four iterative algorithms mentioned above are then tested and benchmarked by simulations. Alternatively, one may also minimize the abscissa of closed-loop zeros in order to suppress closed-loop response overshoots.

The chapter is organized as follows. A concise overview of algebraic controller design in R_{MS} is presented in Section II. A brief description of optimization techniques from [1], [2] utilized in this paper is introduced in Section III. In Section IV, a mathematical model of the skater on a swaying bow is presented. The derivation of the controller structure for the particular plant, in the R_{MS} ring, is provided in Section V. Finally, a simulation comparison of the algorithms is given in Section VI.

II. Overview of Controller Design in R_{MS} Ring

A brief summary of the basic steps and usability of R_{MS} in controller design follows, for details, see details e.g. [10], [19].

For algebraic controller design in R_{MS} it is initially supposed that not only the plant is expressed by the transfer function over R_{MS} but a controller and all system signals are over the ring. Let $W(s)$ be the Laplace transform of the reference signal, $D(s)$ stands for that of the load disturbance, $E(s)$ is transformed control error, $U_0(s)$ expresses the controller output (control action), $U(s)$ represents the plant input affected by a load disturbance, and $Y(s)$ is the plant output controlled signal in the Laplace transform. The plant transfer function is depicted as $G(s)$, and $G_R(s)$ stands for a controller in the scheme depicted in Fig. 1.

The authors kindly appreciate the financial support which was provided by the European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089.

L. Pekař is with the Tomas Bata University in Zlín, Faculty of Applied Informatics, nám. T. G. Masaryka 5555, 76001 Zlín, Czech Republic (corresponding author to provide phone: +420576035161; e-mail: pekar@fai.utb.cz).

R. Prokop is with the Tomas Bata University in Zlín, Faculty of Applied Informatics, nam. T. G. Masaryka 5555, 76001 Zlín, Czech Republic (e-mail: prokop@fai.utb.cz).

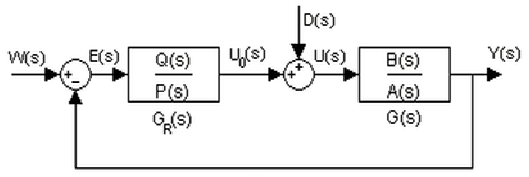

Fig. 1 Control feedback scheme

External inputs, reference and load disturbance signals, respectively, have forms

$$W(s)=\frac{H_W(s)}{F_W(s)},\ D(s)=\frac{H_D(s)}{F_D(s)} \tag{1}$$

where $H_W(s),\ H_D(s),\ F_W(s),\ F_D(s)\in R_{MS}$.

Let the plant be governed by the transfer function

$$G(s)=\frac{B(s)}{A(s)} \tag{2}$$

where $A(s), B(s)\in R_{MS}$ are coprime, i.e. there does not exist a non-trivial (non-unit) common factor of both elements. Details about divisibility can be found in [20] where the ring is also defined and some its properties introduced.

A. Feedback Stabilization

Given a Bézout coprime pair $A(s), B(s)\in R_{MS}$ the closed-loop system is stable if and only if there exists a coprime pair $P(s), Q(s)\in \mathbf{R}_{MS}$ of controller transfer function denominator and numerator, respectively, satisfying the Bézout identity

$$A(s)P(s)+B(s)Q(s)=1 \tag{3}$$

A particular stabilizing solution of (3), say $P_0(s), Q_0(s)$, can be further parameterized as

$$\begin{aligned} P(s)&=P_0(s)\pm B(s)Z(s)\neq 0\\ Q(s)&=Q_0(s)\mp A(s)Z(s) \end{aligned} \tag{4}$$

where $Z(s)\in R_{MS}$. Parameterization (4) is used to satisfy remaining control and performance requirements, such as reference tracking, disturbance rejection etc.

The proof of the statement above can be done analogously as in [9].

B. Reference Tracking

The task is to find $Z(s)\in R_{MS}$ in (4) so that the reference signal is being tracked. The analysis on the scheme in Fig. 1 yields the following: $F_W(s)$ must divide the product $A(s)P(s)$ in the ring.

C. Load Disturbance Rejection

Similarly as in the previous subsection, the load disturbance is tracked if $F_D(s)$ divides $B(s)P(s)$ in R_{MS}.

III. Optimization Iteration Methods Summary

A concise description of some numerical optimization techniques we decided to utilize for the minimization of the objective function (= spectral abscissa) of the study case follows. All these approaches enable to overcome all the difficulties with non-convexity and non-differentiability of the spectral abscissa function. The reader is referred to [2], [12]-[18] for details.

The objective function for retarded TDS agrees with the spectral abscissa $\alpha(\mathbf{K})$, i.e.

$$\min_{\mathbf{K}}\Phi(\mathbf{K})=\min_{\mathbf{K}}\alpha(\mathbf{K})=\min_{\mathbf{K}}\max_{i}\operatorname{Re}(s_i) \tag{5}$$

where

$$\mathbf{K}=\{k_1, k_2, ..., k_r\} \tag{6}$$

is the set of (free, selectable) controller parameters and s_i stand for system poles.

For LTI-TDS of neutral type it is necessary to introduce more complex objective function including the requirements on so-called strong stability [21], see details herein and e.g. in [2], [13], [16].

A. QCSA

The QCSA for retarded systems was introduced in [12] and it was extended in [13] for neutral ones. The algorithm is based on the iterative shifting of the right-most poles of the spectrum to the left by small changes $\mathbf{K}$. The version for retarded systems can be described as follows.

Input: Objective function $\Phi(\mathbf{K})$.

Step 1: Set termination parameters and the number of moved (controlled) poles $m=1$.

Step 2: Compute the right-most poles, e.g. using QuasiPolynomial Mapping based Rootfinder (QPMR), [22].

Step 3: Compute the sensitivity of m right-most poles w.r.t. changes in $\mathbf{K}$, i.e the sensitivity matrix.

Step 4: Move m right-most poles to the left-half plane by applying small changes in $\mathbf{K}$ using the sensitivity matrix.

Step 5: If necessary, increase m. Stop when the available degrees of freedom in the controller do not allow to further reduce $\alpha(\mathbf{K})$; otherwise, go to Step 2.

Output: Values of $\mathbf{K}_{opt}$.

Note that neutral LTI-TDS require additional test to determine the position of the right-most vertical strip of poles, see details in [14]. The strong stability condition has to be tested as well.

B. NM

The NM algorithm belongs to the class of comparative (direct search) algorithms and it was originally published in [17]. This easy-to-use and very popular method does not require the calculation of derivatives of the objective function and thus it is suitable also for non-smooth functions. The basic steps of the general algorithm can be done as follows.

Input: Objective function $\Phi(\mathbf{K})$.

Step 1: Construct the initial working simplex *S*, set transformation and termination parameters.

Step 2: Calculate the termination test information (the number of iterations or the minimum simplex size). If the test is satisfied, stop the algorithm.

Step 3: Order simplex vertices as the worst, second worst and the best one.

Step 4: Calculate the central point and reflex the worst vertex. If the reflection is successful, accept the reflected point in the new working simplex and go to Step 3.

Step 5: Try to use contraction (for a poor reflection) or expansion (for a fair reflection). If this succeeds, the accepted point becomes the new vertex; otherwise, shrink the simplex towards the best vertex. Go to Step 3.

Output: The best vertex $\mathbf{K}_{opt}$ and its function value $\Phi(\mathbf{K}_{opt})$.

C. EGSA

The EGSA is based on the gradient sampling algorithm developed in [14] as its extension to (neutral) TDS, see [15], [16]. The original algorithm is essentially an extension of the well-known steepest descent method, where a numerical estimation of the gradient is calculated instead. This enables to calculate it even in points where the objective function is not differentiable. The basic steps of the EGSA can be given as follows.

Input: Objective function $\Phi(\mathbf{K})$.

Step 1: Initialize a starting point $\mathbf{K}_0$ arbitrarily. Set control (how to calculate gradient estimation, step length etc.) and termination parameters.

Step 2: Choose $r+1$ points near by $\mathbf{K}_0$. Compute the Clarke subdifferential and the (non-smooth) steepest descent direction using the gradient sampling method. If the norm of the direction is very small, then terminate the algorithm.

Step 3: Calculate the step length along the direction from Step 2. If it fails choose another (substitute) direction. If all possible directions fail, stop.

Step 4: Update the current position $\mathbf{K}_i$ to $\mathbf{K}_{i+1}$ and go to Step 2.

Output: The best position and its function value.

D. SOMA

The SOMA is ranked among genetic algorithms, dealing with populations, see e.g. [18]. Population specimens cooperate while searching the best solution (the minimum of the cost function) and, simultaneously, each of them tries to be a leader, i.e. "the best one". They move to each other and the searching is finished when all specimens are localized on a small area. The main steps of the basic algorithm strategy called "All to One" can be formulated as follows.

Input: Objective function $\Phi(\mathbf{K})$.

Step 1: Set control (step length, a perturbation vector, etc.) and termination parameters. Generate a population based on a selected prototypal specimen.

Step 2: Find the best specimen (leader), i.e. that with the minimal function value.

Step 3: Move all other specimens towards the leader and evaluate their function values in each step.

Step 4: Select the new population and test the minimal divergence of the population. If it succeeds, stop. Otherwise, go to Step 2.

Output: The leader and its function value.

IV. Model and Control of the Skater on a Swaying Bow

In this section, a mathematical model of a real-life controlled system is introduced which will be used subsequently when benchmark and comparison of the four minimization techniques. Namely, an unstable model of the roller skater on a controlled swaying bow is described briefly by mean of the transfer function. Consequently, a corresponding controller using the R_{MS} ring is derived.

A. Model of the Skater on a Controlled Swaying Bow

Consider an unstable system as in Fig. 2 expressing a roller skater a swaying bow. In [10] it has been stated that the transfer function of the system reads

$$G(s)=\frac{Y(s)}{U(s)}=\frac{b\exp(-(\tau+\vartheta)s)}{s^2\left(s^2-a\exp(-\vartheta s)\right)} \tag{7}$$

where $y(t)$ is the skater's deviation from the desired position, $u(t)$ expresses the slope angle of a bow caused by force *P*, delays τ,ϑ means the skater's and servo latencies, respectively, and *b*, *a* are real parameters. Skater controls the servo driving by remote signals into servo electronics. As presented in literature, let *b* = 0.2, *a* = 1, $\tau=0.3$ s, $\vartheta=0.1$ s.

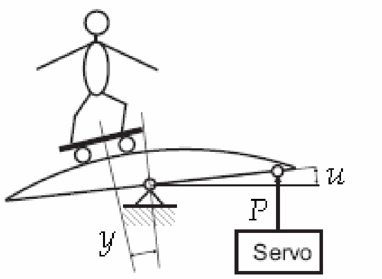

Fig. 2 Roller skater on a controlled swaying bow

B. Roller Skater Controller Structure Design

Prior the numerical spectral abscissa optimization, anisochronic controllers have to be derived (Pekař and Prokop 2011b).

First of all, factorize the plant transfer function as

$$G(s)=\frac{B(s)}{A(s)}=\frac{\dfrac{b\exp(-(\tau+\vartheta)s)}{(s+m_0)^4}}{\dfrac{s^2\left(s^2-a\exp(-\vartheta s)\right)}{(s+m_0)^4}},\ A(s),B(s)\in\mathbf{R}_{MS} \tag{8}$$

where $m_0>0$ is a selectable real parameter. Consider the reference and load disturbance in the form of a step-wise function, hence their Laplace forms are respectively

$$W(s)=\frac{H_W(s)}{F_W(s)}=\frac{\dfrac{w_0}{m_W(s)}}{\dfrac{s}{m_W(s)}},\ D(s)=\frac{H_D(s)}{F_D(s)}=\frac{\dfrac{d_0}{m_D(s)}}{\dfrac{s}{m_D(s)}} \tag{9}$$

where w_0, $d_0\in\mathbb{R}$, $m_W(s)$ and $m_W(s)$ are arbitrary stable (retarded) (quasi)polynomials of degree one and $H_W(s), H_D(s), F_W(s), F_D(s)\in R_{MS}$.

Stabilization formula (3) using the generalized Euclidean algorithm, see details in [23], yields e.g.

$$\begin{aligned}&Q_0(s)=\\&\frac{\left(q_3s^3+q_2s^2+q_1s+q_0\right)(s+m_0)^4}{s^2\left(s^2-a\exp(-\vartheta s)\right)\left(s^3+p_2s^2+p_1s+p_0\right)+b\exp(-(\tau+\vartheta)s)\left(q_3s^3+q_2s^2+q_1s+q_0\right)}\\&P_0(s)=\\&\frac{\left(s^3+p_2s^2+p_1s+p_0\right)(s+m_0)^4}{s^2\left(s^2-a\exp(-\vartheta s)\right)\left(s^3+p_2s^2+p_1s+p_0\right)+b\exp(-(\tau+\vartheta)s)\left(q_3s^3+q_2s^2+q_1s+q_0\right)}\end{aligned} \tag{10}$$

where p_2, p_1, p_0, q_3, q_2, q_1, $q_0\in\mathbb{R}$ are free parameters.

In order to satisfy tasks of step-wise reference tracking and load disturbance rejection, parameterization (4) can be used so that both, $F_W(s)$and $F_D(s)$, divide $P(s)$; hence, the numerator of $P(s)$ satisfies $P(0)=0$. To make $P(s)$ as simple as possible, choose

$$\begin{aligned}&Z(s)=\\&\frac{z_0(s+m_0)^4}{s^2\left(s^2-a\exp(-\vartheta s)\right)\left(s^3+p_2s^2+p_1s+p_0\right)+b\exp(-(\tau+\vartheta)s)\left(q_3s^3+q_2s^2+q_1s+q_0\right)}\end{aligned} \tag{11}$$

where $z_0\in\mathbb{R}$ is a selectable parameter. Both divisibility conditions are satisfied if

$$z_0=\frac{-p_0m_0^4}{b} \tag{11}$$

This results in the final anisochronic controller of the transfer function

$$\begin{aligned}&G_R(s)=\frac{Q(s)}{P(s)}\\&=\frac{b\left(q_3s^3+q_2s^2+q_1s+q_0\right)(s+m_0)^4+p_0m_0^4s^2\left(s^2-a\exp(-\vartheta s)\right)}{b\left[\left(s^3+p_2s^2+p_1s+p_0\right)(s+m_0)^4-p_0m_0^4\exp(-(\tau+\vartheta)s)\right]}\end{aligned} \tag{12}$$

The reference-to-output transfer function reads

$$\begin{aligned}&G_{WY}(s)=\frac{1}{(s+m_0)^4}\\&=\frac{b\left[b\left(q_3s^3+q_2s^2+q_1s+q_0\right)(s+m_0)^4+p_0m_0^4s^2\left(s^2-a\exp(-\vartheta s)\right)\right]\exp(-(\tau+\vartheta)s)}{s^2\left(s^2-a\exp(-\vartheta s)\right)\left(s^3+p_2s^2+p_1s+p_0\right)+b\exp(-(\tau+\vartheta)s)\left(q_3s^3+q_2s^2+q_1s+q_0\right)}\end{aligned} \tag{13}$$

which gives rise to the characteristic quasipolynomial

$$\begin{aligned}&m(s)=(s+m_0)^4\\&\left[s^2\left(s^2-a\exp(-\vartheta s)\right)\left(s^3+p_2s^2+p_1s+p_0\right)+b\exp(-(\tau+\vartheta)s)\left(q_3s^3+q_2s^2+q_1s+q_0\right)\right]\end{aligned} \tag{14}$$

Obviously, there are two factors in (14), a polynomial and a quasipolynomial one. Since the spectral assignment for the polynomial factor is trivial, the goal is to minimize the spectral abscissa of the quasipolynomial factor with seven unknown parameters. To cancel the impact of the quadruple real pole $s_1=-m_0$, it must hold that $m_0>>\alpha(\mathbf{K})$.

To sum up, the task is to solve the problem

$$\mathbf{K}_{opt}=\arg\min_{\mathbf{K}}\Phi(\mathbf{K})=\arg\min_{\mathbf{K}}\alpha(\mathbf{K}) \tag{15}$$

where $\alpha(\mathbf{K})$ is the abscissa of zeros of the quasipolynomial

$$\begin{aligned}&s^2\left(s^2-\exp(-0.1s)\right)\left(s^3+p_2s^2+p_1s+p_0\right)\\&+0.2\exp(-0.4s)\left(q_3s^3+q_2s^2+q_1s+q_0\right)\end{aligned} \tag{16}$$

and

$$\mathbf{K}=[\,p_2,p_1,p_0,q_3,q_2,q_1,q_0\,]^T \tag{17}$$

Alternatively, the optimization problem can be formulated as

$$\begin{aligned}\mathbf{K}_{opt}&=[\,p_2,p_1,p_0,q_3,q_2,q_1,q_0\,]_{opt}^T=\arg\min_{\mathbf{K}}\alpha(\mathbf{K})\\&=\arg\min_{\mathbf{K}}\left\{\operatorname{Re}s_i:[m_1(s)]_{s=s_i}=0\right\}\end{aligned} \tag{18}$$

V. Optimization Techniques Comparison

The aim of the section is to compare the performance and efficiency of optimizing algorithms presented above based on the minimization of the spectral abscissa of the feedback characteristic quasipolynomial factor (16), derived when control of the skater on a swaying bow.

Let the minimization starts by the QCSA from the point $\mathbf{K}_0 = [1,1,1,1,1,1,1]^T$ defined in (17). This initial setting gives rise to the spectrum Ω (system poles s_i are from the region with Re s_i > -2)

$$\Omega_0 = \{0.849185, 0.477189, 1, 1, -0.604644, -0.820218, -1\} \quad (19)$$

Obviously, the feedback system is unstable with $\alpha(\mathbf{K}) = 0.849185$. The QCSA is capable to move some controlled poles to the left. Unlike the original paper [12], the number of controller poles is not increased gradually here, however, this quantity is changed depending on poles locations. More precisely, whenever a dominant root of (16) (or a bunch or dominant roots) secedes from the rest of the spectrum and the number of currently controlled roots is higher then the number of seceded ones, the number of controlled roots decreases so that only of seceded roots are controlled.

The evolution of system poles is displayed in Fig. 3 where the controlled ones are in bold lines. Notice that in an iterations range approximately of *i* = 600-1750, there is a huge number of bifurcations of a complex pair of roots or that a double real root into a pair of single real roots, and viceversa. This yields many changes in the number of controlled roots. Whenever a root remains uncontrolled, it eventually reaches the controlled rightmost bunch of roots. A detailed view on the iterations range of *i* = 600-1750 is in Fig. 4.

From Fig. 3 it is obvious that the procedure can adjust the spectral abscissa such that $\alpha(\mathbf{K}) < -1.5$ and it seems that this improvement may continue long. However, as it is presented below, the improvement is limited – as was found during the preparing of this paper. Therefore, we decided to test the three remaining optimization methods so that they initiate at the point

$$\mathbf{K}_{2400} = [5.4426, 9.838022, 27.592742, 148.615441, \\ 161.171962, 5.245781, 0.51334]^T \quad (20)$$

and we compare their efficiencies with the QCSA in a number of subsequent iterations. Time consumption of an appropriate method is measured as well.

The overall development of $\mathbf{K}$ can be seen in Fig. 5; however, due to the noticeable rise in values for *i* > 1700, the detailed view on the iterations range of 1-1700 is in Fig. 6.

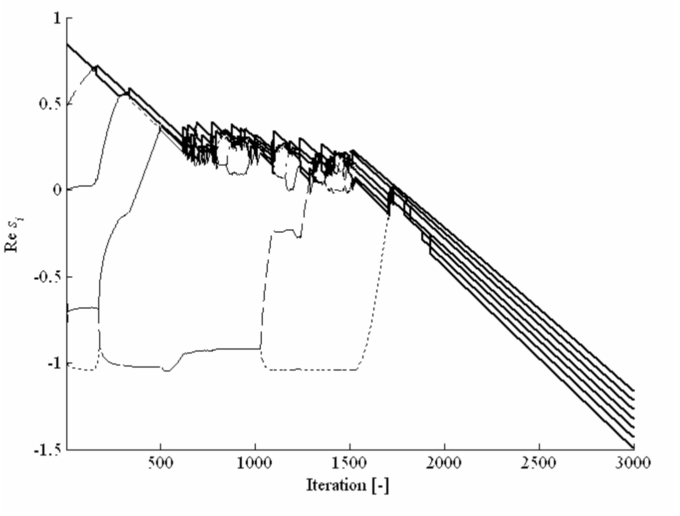

Fig. 3 Evolution of real parts of the rightmost roots of (16) using the QCSA

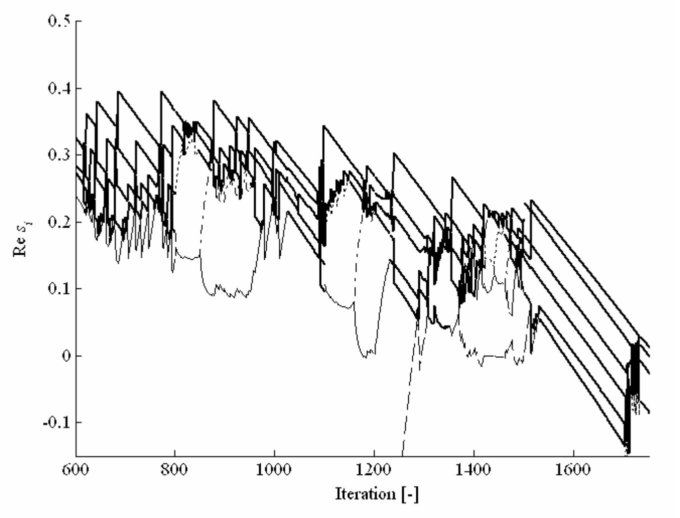

Fig. 4 A detailed view on Fig. 3 for iterations range of *i* = 600-1750

A. NM

The evolution of $\alpha(\mathbf{K})$ using the NM with the setting $ni = 300$, $\varepsilon_S = 10^{-6}$, $h_j = 1, j = 2,...,r+1$, see [2] for parameters definitions, is depicted in Fig. 7. Iterations are terminated in *i* = 288 since the simplex size becomes smaller then ε_S. The results for $h_j = 10, j = 2,...,r+1$ are also added into the figure for the comparison. Corresponding developments of $\mathbf{K}$s are not displayed here since there are no appreciable changes in the values.

Obviously, the higher initial simplex edge length brings more possibilities how to escape from the local minimum which can bee seen in Fig. 7 and it prevents the premature termination due to the simplex size.

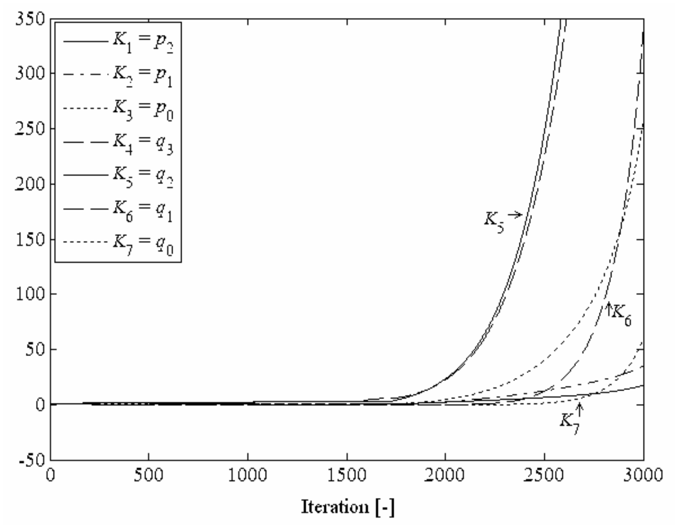

Fig. 5 Evolution of **K** for (16) and (17) using the QCSA

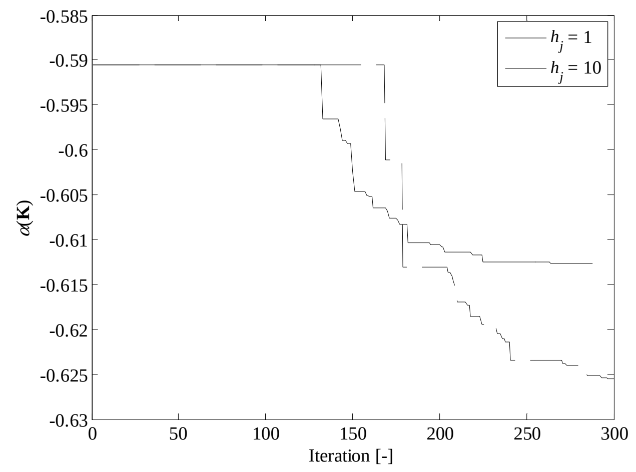

Fig. 7 Evolution of $\alpha(\mathbf{K})$ using the NM for $h_j = 1$ and $h_j = 10$, respectively, from $i = 2400$

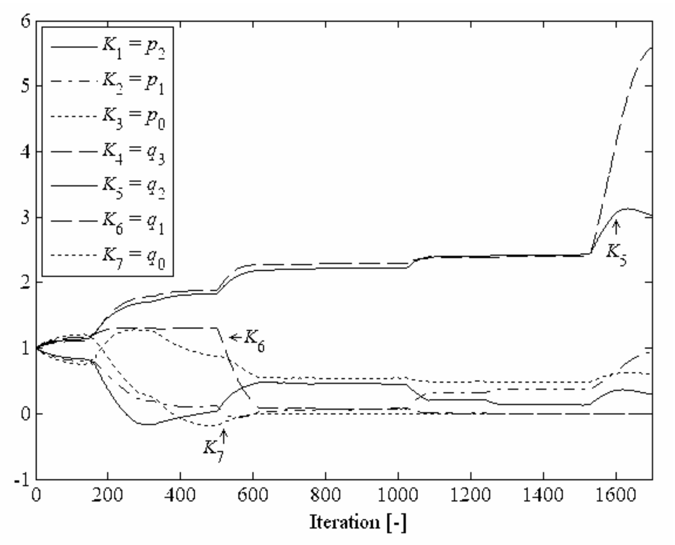

Fig. 6 Evolution of **K** for (16) and (17) using QCSA – a detail for the iterations range of i = 1-1700

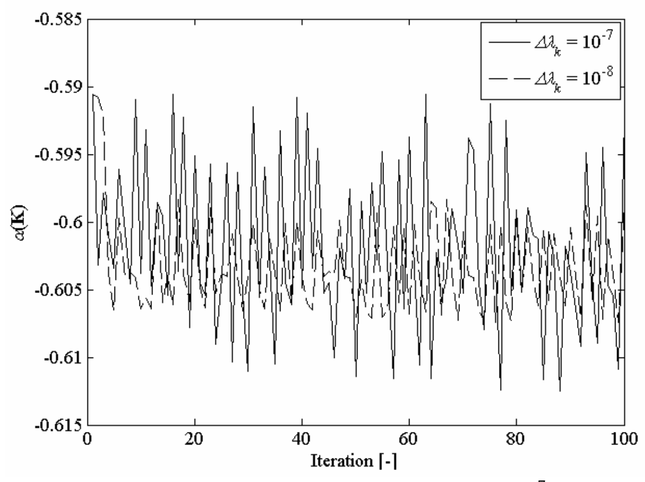

Fig. 8 Evolution of $\alpha(\mathbf{K})$ using the EGSA for $\Delta\lambda = 10^{-7}$ and $\Delta\lambda = 10^{-8}$, respectively, from $i = 2400$

The method is quite fast (≈ 8 s per iteration on Intel Core2 Duo CPU E8500@ 3.16 GHz, 4BG RAM) because of a low number of spectrum calculations using *qpmr* function in Matlab [21]. This operation is the most time consumptive command in a program code, especially when a high accuracy with a large searching region is required.

B. EGSA

Analogously to the previous subchapter, spectral abscissa evolutions via the EGSA for settings $\rho = 10^{-4}$, $l = r + 1 = 8$, $\varepsilon = 10^{-5}$, $\overline{\lambda_{\max}} = 2 \cdot 10^{-6}$, $\Delta\lambda = 10^{-7}$ and $\overline{\lambda_{\max}} = 2 \cdot 10^{-7}$, $\Delta\lambda = 10^{-8}$, respectively, are displayed in Fig. 8. The number of iteration steps equals 100. For details about the parameters, see [2] again.

It is apparent that the method does not bring a significant improvement (i.e. a sufficient decreasing) of $\alpha(\mathbf{K})$ mainly due to many cases when $\lambda_{\max} = k = 0$.

It is questionable whether the obstacle can be solved by decreasing of ρ or that of $\Delta\lambda$; however, this example shows that a variation in $\Delta\lambda$ within one order does not bring a satisfactory result. On the other hand, a lower value of ρ results in a higher gradient norm which means a numerical difficulties.

There are chosen 100 iterations since one iteration takes 65 seconds in average which is eight times more time consumptive then the NM, since the EGSA requires (in the worse case) $l + 1 + k \cdot (l + 1) = (l + 1)(k + 1)$ spectrum calculations per iteration.

C. SOMA

Evolutions of $\alpha(\mathbf{K})$ using the SOMA with settings $PopSize = 10$, $PathLength = 3$, $Step = 0.21$, $PRT = 0.6$, $M = 30$, $MinDiv = 10^{-4}$, $Rad = 1$ and $Rad = 10$, respectively are depicted in Fig. 9. Again, a higher value of Rad enables to scan the space of parameters more effectively resulting in a faster decrease of $\alpha(\mathbf{K})$. Parameters are defined in [2].

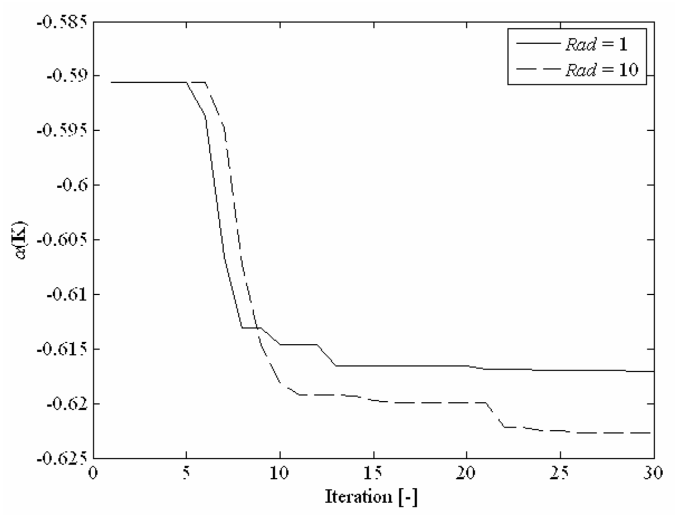

Fig. 9 Evolution of $\alpha(\mathbf{K})$ using the SOMA for *Rad* = 1 and *Rad* = 10, respectively, from i = 2400

Only 30 migration rounds (iterations) are chosen since every step is a rather time consumptive due to a high number of calling *qpmr* function. Meanwhile NM has approximately 2 or 3 spectrum calculations per iteration, the SOMA requires $\mathrm{round}\left(\left(PopSize-1\right)\cdot PathLength/Step\right)$ enumerations. An iteration step takes approximately 560 s here, i.e. 1 iteration step of SOMA lasts 70 iterations of the NM.

D. *Partial Summary*

Compare now best results for every of the approaches between each other. There are two possibilities how to cope with it – one can either draw a comparison for every iteration step, or (which is more impartial) to compare results w.r.t. the same calculation time. Both of the results are displayed in Fig. 10 and Fig. 11, respectively. Finally, minimization in time range using the NM, SOMA and the using the QCSA (one iteration step of the NM equals one iteration step of the quasi-continuous shifting algorithm which is a time unit) can be seen in Fig. 12.

From the figures above it can be evident that presented optimization algorithms can serve for searching the local minimum rather then the global optimum when solving the spectral abscissa of TDS. Simplex edge length and radius of the population are the crucial optional parameters in the NM and the SOMA, respectively. The higher values enable to scour the solution space more efficiently and these options reduce the risk of a premature termination due to small simplex size (as obvious from Fig. 11 where the simplex size reaches 10^{-8} at iteration step of i = 600) or a minimal divergence, respectively. The EGSA seems to be a rather useless because of troublesome gradient calculation and searching the step length. Moreover, this algorithm together with the SOMA are much slower then the NM since many spectrum calculations.

The three latter numerical approaches in comparison to the QCSA are useful if $\alpha(\mathbf{K})$ is nonsmooth at any point; however, to reach such points exactly is almost impossible.

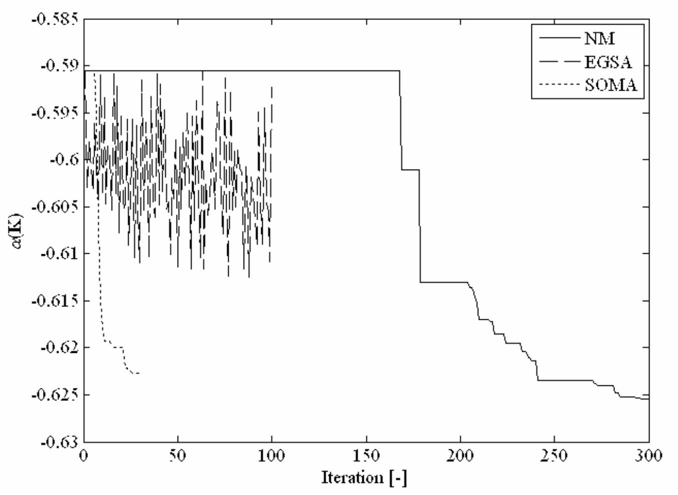

Fig. 10 Evolution of $\alpha(\mathbf{K})$ using the NM (d_j = 10), EGSA ($\Delta\lambda = 10^{-7}$) and SOMA (*Rad* = 10) in iterations range (starting from i = 2400)

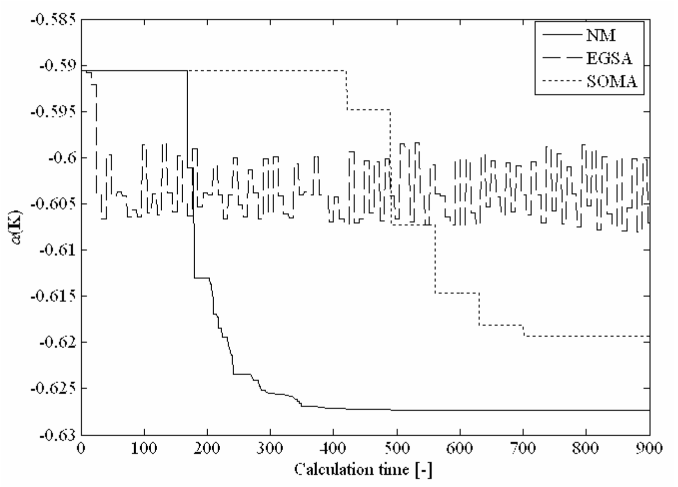

Fig. 11 Evolution of $\alpha(\mathbf{K})$ using the NM (d_j = 10), EGSA ($\Delta\lambda = 10^{-7}$) and SOMA (*Rad* = 10) in the calculation time range (starting from i = 2400)

Although the NM is the oldest algorithm from the ones presented here, it provides the best results together with the SOMA, yet it is much faster when implemented.

The final results for the best method here, i.e. the NM starting from i = 2400, are the following (compared to the QCSA in i = 2400)

$$\mathbf{K}_{2400,NM} = [5.442878, 9.838107,\ 27.592764,\ 148.615484,\ 161.172496,\ 5.246047,\ 0.513388]^T \tag{21}$$

$$\Omega_{2400} = \left\{ \begin{array}{l} -0.5906, -0.6329, -0.675, -0.7243, \\ -0.7664, -0.8151, -0.8661 \end{array} \right\} \tag{22}$$

$$\Omega_{2400,NM} = \left\{ \begin{array}{l} -0.6255 \pm \mathrm{j}0.0369, -0.6255 \pm \mathrm{j}0.0369, \\ -0.7946 \pm \mathrm{j}0.1609, -0.9842 \end{array} \right\} \tag{23}$$

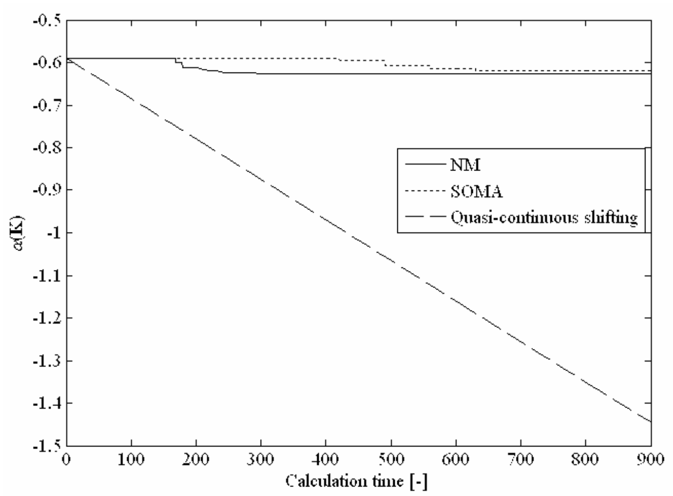

Fig. 12 Evolution of $\alpha(\mathbf{K})$ using the NM (d_j = 10), SOMA (Rad = 10) and the quasi-continuous shifting algorithm in calculation time range (starting from i = 2400)

Controller parameters set as in (21) give the control responses depicted in Fig. 13. Control action for $m_0 = 0.5$ is not unstable, yet, with an oscillatory response of a very high period since feedback transfer function zeros are located in the right-half plane and too close to the imaginary axis. The result is quite poor also because of the fact that system poles lie too close to the imaginary axis.

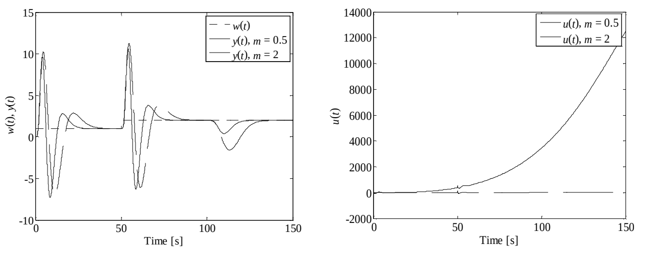

Fig. 13 Control responses ($y(t)$ - left, $u(t)$- right) for the setting as in (23) obtained by the NM, for m_0 = 0.5 and m_0 = 2

E. Improved Results

During the preparing of this paper, we try to perform a new test consisting in continuation of iterations, which have been started by the QCSA and displayed in Fig. 3. Results for the iteration range $i \in [3000, 3520]$ can be seen in Fig. 14.

It is clear that the improvement of the spectral abscissa terminates at $i = 3305$. The values of $\mathbf{K}$ and the corresponding spectrum (its dominant part, more precisely) in this iteration step read

$$\mathbf{K}_{3305} = [469418.2, 640264.2, 10560107, 8222650, \\ 106523133, 26247749, 5617613]^T \tag{24}$$

$$\Omega_{3305} = \left\{ \begin{matrix} -1.4454, -1.5056, -1.5617, -1.6187, \\ -1.6745, -1.7345, -1.802 \end{matrix} \right\} \tag{25}$$

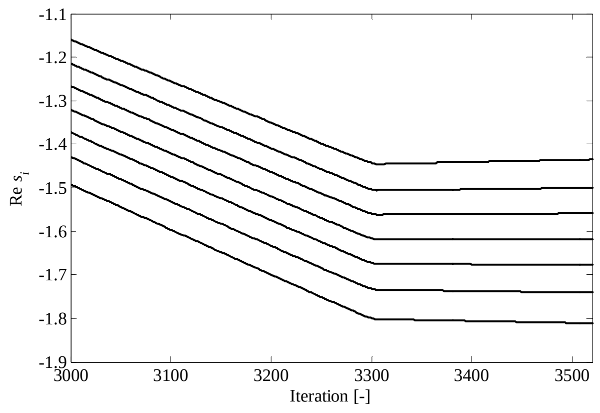

Fig. 14 Evolution of real parts of the rightmost roots of (16) using the QCSA within the iteration range $i \in [3000, 3520]$

The NM yields the development of $\alpha(\mathbf{K})$ as in Fig. 15.

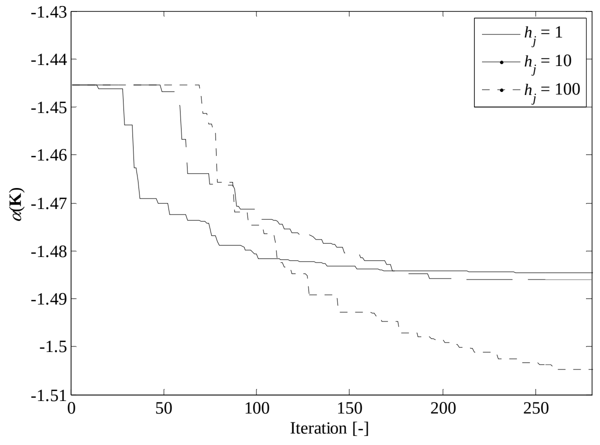

Fig. 15 Evolution of $\alpha(\mathbf{K})$ using the NM for h_j = 1, h_j = 10 and h_j = 100, respectively, from i = 3305

As can be seen from (24), values of $\mathbf{K}$ are very high and unusable in practice. However, we try to test the remaining algorithms starting in this local minimum, i.e. from $i = 3305$.

Again, the longer the initial simplex edge is, the slower but much better the minimization is obtained. It is substantial that the local minimum from the QCSA has been improved by the NM.

The EGSA gives results displayed in Fig. 16. Amazingly, the improvement of the spectral abscissa is much better then for i = 2400 (see Fig. 8 for the comparison) and, again, a new local minimum has been found.

Finally, the development of $\alpha(\mathbf{K})$ using the SOMA for two different initial population radii is shown in Fig. 17. The result is almost comparable with the NM, yet, the iteration process is much slower compared to this classical optimization method. This fact is clear from Fig. 18 where the best results from all the three methods are compared in the time range.

To sum up, the best result with the minimal value of $\alpha(\mathbf{K})$ obtained by the NM gives the following position of the right-most poles as in (26).

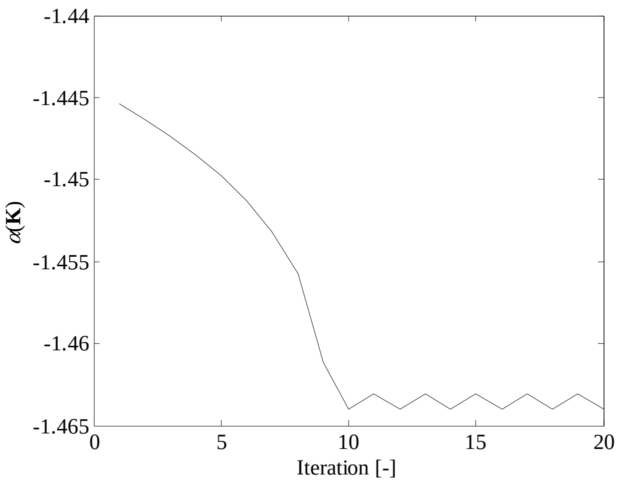

Fig. 16 Evolution of $\alpha(\mathbf{K})$ using the EGSA for $\Delta\lambda_k = 10^{-5}$, from i = 3305

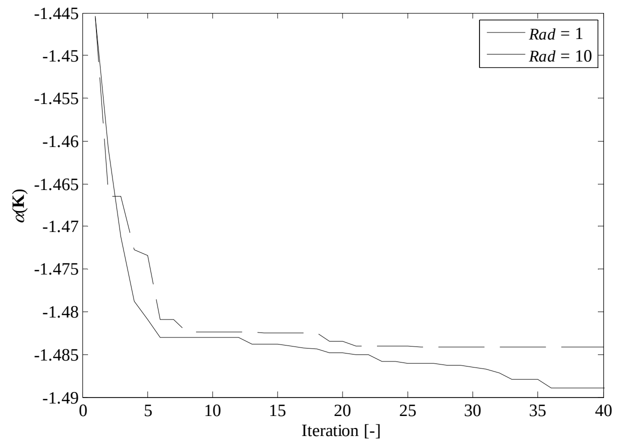

Fig. 17 Evolution of $\alpha(\mathbf{K})$ using the SOMA for *Rad* = 1 and *Rad* = 10, respectively, from i = 3305

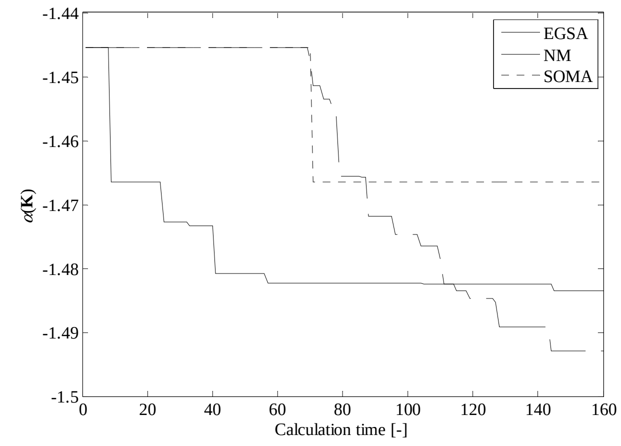

Fig. 18 Evolution of $\alpha(\mathbf{K})$ using the NM (d_j = 100), EGSA ($\Delta\lambda = 10^{-5}$) and SOMA (*Rad* = 10) in the calculation time range (starting from i = 3305)

$$\Omega_{3305,NM} = \begin{Bmatrix} -1.5048 \pm 0.0065\mathrm{j},\ -1.5048 \pm 0.0291\mathrm{j}, \\ -1.7203 \pm 0.1486\mathrm{j}, -1.8828 \end{Bmatrix} \quad (26)$$

The corresponding values of **K** do not differ significantly from (24). Simulated control responses are not displayed here due to the numerical problems with simulation program (caused by high values of controller parameters).

VI. Conclusion

The study case supported by simulation verifies the usability of a standard shifting algorithm (QCSA) as an initial procedure when minimizing the spectral abscissa. However, once the local minimum is reached, it is convenient to utilize another, more sophisticated, optimization algorithm which can deal with a non-smooth or non-Lipshitz functions.

We have shown that the classical Nelder-Mead algorithm gives very good results. Besides a significant improvement of the spectral abscissa, it is quite fast because of a very small number of spectrum evaluations.

Self-Organizing Migration Algorithm provides comparable results, yet, it is much slower here. If one does not care about the time, this approach is a good choice as well. The usability of the (Extended) Gradient Sampling Algorithm is debatable. In the first test, it has provided fast spectral abscissa improvement, yet next iterations have not brought a significantly better evolution. The second test has given much better score; however, the results have not been as satisfactory as in for other methods.

In the future research, we can extend the methods to neutral TDS, improve the controller structure design or to optimize control and termination parameters of presented optimization methods, i.e. to perform some kind of "meta-optimization".

References

[1] L. Pekař and R. Prokop, "Optimal pole placement for LTI-TDS via some advanced iterative algorithms – part I: theory," in *Recent Researches in Circuits and Systems, Proc. of the 16th WSEAS Int. Conf. on Systems*, Kos, Greece, 2012, pp. 196-201.

[2] L. Pekař, "On the optimal pole assignment for time-delay systems", *Int. J. Math. and Computers in Simulation*, submitted for publication.

[3] V. Volterra, "Sur le théorie mathématique des phénomenès héreditaires," *Journal des Mathématiques Pures et Appliquées*, vol. 7, pp. 249-298, 2003.

[4] J. E. Marshall, H. Górecki, A. Korytowski, and K. Walton, *Time Delay Systems, Stability and Performance Criteria with Applications*, Ellis Horwood, 1992.

[5] J. K. Hale and S. M. Verduyn Lunel, "Introduction to functional differential equations," vol. 99 of *Appl. Math. Scien.*, New York: Springer-Verlag, 1993.

[6] G. Mircea, M. Pirtea, M. Neamtu, and D. Opris, "Stochastic, fuzzy and hybrid monetary models with delay," *WSEAS Trans. Mathematics*, vol. 9, issue 7, pp. 571-580, 2010.

[7] M. Staněk, D. Maňas, M. Maňas, O. Šuba, "Optimization of Injection Molding Process," *Int. J. Math. and Computers in Simulation*, vol. 5, no. 5, pp. 413-421, 2011.

[8] J.-P. Richard, "Time-delay systems: an overview of some recent advances and open problems," *Automatica*, vol. 39, no. 10, pp. 1667-1694, 2003.

[9] P. Zítek and V. Kučera, “Algebraic design of anisochronic controllers for time delay systems,” *Int. J. Control*, vol. 76, no. 16, pp. 905-921, 2003.

[10] P. Zítek, V. Kučera, and T. Vyhlídal, „Meromorphic observer-based pole assignment in time delay systems,“ *Kybernetika*, vol. 44, no. 5, pp. 633-648, 2008.

[11] L. Pekař and R. Prokop, “On reference tracking and disturbance rejection for time delay systems,“ in *Proc. of the 31st IASTED International Conference on Modelling, Identification, and Control (MIC 2011)*, Innsbruck, Austria, 2011, pp. 327-333.

[12] W. Michiels, K. Engelborghs, P. Vansevenant, and D. Roose, “Continuous pole placement for delay equations,” *Automatica*, vol. 38, no. 6, pp. 747-761, 2002.

[13] W. Michiels and T.Vyhlídal, “An eigenvalue based approach for the stabilization of linear time-delay systems of neutral type,” *Automatica*, vol. 41, no. 6, pp. 991-998, 2005.

[14] J. Burke, A. Lewis, and M. Overton, “A robust gradient sampling algorithm for nonsmooth, nonconvex optimization,” *SIAM Journal of Optimization*, vol. 15, no. 3. pp. 751-779, 2005.

[15] T. Vanbiervliet, K. Verheyden, W. Michiels, and S. Vandewalle, “A nonsmooth optimization approach for the stabilization of time-delay systems,” *ESIAM. Control, Optimisation and Calculus of Variations*, vol. 14, no. 3, pp. 478-493, 2008.

[16] T. Vyhlídal, W. Michiels, and P. McGahan, “Synthesis of a strongly stable state-derivative controller for a time delay system using constrained nonsmooth optimization,” *IMA Journal of Mathematical Control and Information*, vol. 27, no. 4, pp. 437-455, 2010.

[17] J. A. Nelder and R. Mead, “A simplex method for function minimization,” *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.

[18] I. Zelinka, “SOMA-self organizing migrating algorithm,” in *New Optimization Techniques in Engineering*, G. C. Onwobolu and B. V. Babu, Eds. Berlin: Springer, 2004, pp. 167-217.

[19] L. Pekař and R. Prokop, “Implementation of a new quasi-optimal controller tuning algorithm for time-delay systems,” in *MATLAB for Engineers – Applications in Control, Electrical Engineering, IT and Robotics*, 1st ed., K. Perůtka, Ed. Rijeka, Croatia: InTech, 2011, pp 3-26.

[20] L. Pekař and R. Prokop, “A revised ring of stable and proper quasipolynomial meromorphic functions for LTI-TDS,” in *Proc. of the 14th WSEAS Int. Conf. on Math. and Computational Methods in Science and Engineering*, Sliema, Malta, 2012, to be published.

[21] J. K. Hale and S. M. Verduyn Lunel, “Strong stabilization of neutral functional differential equations,” *IMA J. Math. Control and Information*, vol. 19, pp. 5-23, 2002.

[22] T. Vyhlídal and P. Zítek, “Quasipolynomial mapping based rootfinder for analysis of time delay systems,” in *Proc. IFAC Workshop on Time-Delay systems* (*TDS’03*), Rocquencourt, France, 2003, pp. 227-232.

[23] L. Pekař and R. Prokop, “Some observations about the RMS ring for delayed systems,” in *Proc. of the 17th Int. Conf. on Process Control ‘09*, Štrbské Pleso, Slovakia, 2009, pp. 28-36.