ANALYSIS OF INFINITE SPECTRA OF TIME DELAY SYSTEMS USING QUASIPOLYNOMIAL ROOT FINDER IMPLEMENTED IN MATLAB

Tomáš Vyhlídal & Pavel Zítek

Czech Technical University in Prague,
Institute of Instrumentation and Control Engineering,
Centre for Applied Cybernetics

Abstract: Two original algorithms for computing the roots of low order quasipolynomials are introduced in the paper. The first algorithm is based on Weyl's construction combined with argument principle rule. The second algorithm is based on quasipolynomial function mapping in the complex plane. Both algorithms provide the approximate positions of the roots located inside a selected region. The accuracy of the roots is then increased by using Newton's iteration method. The quasipolynomial rootfinder using the second algorithm has been implemented in Matlab. The presented algorithms provide the significant contribution to the analysis of time delay systems, whose characteristic functions are of the quasipolynomial form.

Keywords: time delay system, quasipolynomial roots, system poles and zeros

1 INTRODUCTION

Description of plants by means of models with time delays has recently become common in control engineering. Unlike the classical system description, this kind of models allows to describe basic features of real systems, i.e., transport phenomenon, after-effects or distributed parameters without necessity of using higher order models, see e.g., Zítek (1998). The dynamics of both classical and time delay systems (TDS) are determined by positions of their poles and zeros, respectively. In order to explain concept of system poles and zeros, let us consider the classical linear SISO system

$$\begin{aligned} \frac{d\mathbf{x}(t)}{dt} &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t) \\ y(t) &= \mathbf{C}\mathbf{x}(t) \end{aligned} \tag{1}$$

where $\mathbf{x}$ is the vector of system state, u and y are the system input and output, respectively, and $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are the coefficient matrices of appropriate dimensions. Transforming state description (1) into input-output relation and performing Laplace transform (considering zero initial conditions), the model acquires the form of transfer function

$$G(s) = \frac{y(s)}{u(s)} = \mathbf{C}[s\mathbf{I} - \mathbf{A}]^{-1}\mathbf{B} = \frac{N(s)}{M(s)} \tag{2}$$

where $\mathbf{I}$ is the identity matrix and s is operator of Laplace transform. The poles of system (1) are the solutions of the characteristic equation

$$M(s) = \det(s\mathbf{I} - \mathbf{A}) = s^n + \sum_{i=0}^{n-1} a_i s^i = 0 \tag{3}$$

where n is system order and a_i, i=1..n are the coefficients of polynomial $M(s)$ while the system zeros are the solutions of the equation

$$N(s) = \mathbf{C}\,\mathrm{adj}\big[s\,\mathbf{I} - \mathbf{A}\big]\mathbf{B} = \sum_{i=0}^{m} b_i s^i = 0 \tag{4}$$

where $m \le n$ and b_i, i=1..m are the coefficients of polynomial $N(s)$. It should be noted that the system poles, i.e., the roots of $M(s)$ are identical with the eigenvalues of matrix $\mathbf{A}$. The position of system poles in the s-plane determines the basic features and modes of the system dynamics, i.e., stability,

natural frequency, damping ratio of the responses. The system zeros determines the proportion in which the system poles are combined in system input-output responses (Goodwin, 2001).

2 COMPUTING POLYNOMIAL ROOTS

As can be seen, both functions $M(s)$ and $N(s)$ in (2) are polynomials. There exists a great deal of methods for computing the polynomial roots, see, e.g., the overview in Pan, (1997). If the polynomial degree is 2, the root finding process is trivial, based on the formula for computing the roots of quadratic equation. The analogous formulas, however much more complicated, are available for polynomials of degrees 3 and 4 (of some specific form). The nonexistence of such formulas for the polynomials of degree $n>4$ has been proved already in the beginning of the nineteenth century. Thus, a numerical method has to be used for computing the roots of such polynomials. Actually, also the roots of polynomials of degrees 3 and 4 are usually solved numerically, since it is rather cumbersome to compute the roots in an analytic way.

Many of the existing numerical algorithms for computing polynomial roots are based on the fact that the polynomial roots are identical with the eigenvalues of the companion matrix of the polynomial. For example the companion matrix of polynomial $M(s)$ given by (3) can be of the following form

$$\mathbf{C}(M)=\begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ & \vdots & & \ddots & \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{bmatrix} \tag{5}$$

Thus the problem of computing the roots of $M(s)$ is changed into the problem of computing the eigenvalues of (5). Such an approach is also used in Matlab function *roots* in which matrix (5) is built from the polynomial coefficients and its eigenvalues are evaluated by means of function *eig* using the subroutines of LAPACK (Anderson, et. al., 1999). The theory of numerical computing the matrix eigenvalues is rather extensive, see, e.g., Wilkinson, (1965). There are available algorithms for solving the eigenvalue problem for sparse matrices and also for the matrices of considerably high order. A comprehensive practical guide for the methods of computing the matrix eigenvalues where the theory overview as well as the final code-written algorithms can be found is (Bai, et al, 2000). It should be noted that higher order polynomials are likely to be ill-conditioned, see Wilkinson (1984), i.e., the small changes of the polynomial coefficients (given, e.g., by truncation errors) can completely change the spectrum of the roots. Dealing with such ill-conditioned polynomials requires applying special iterative methods, e.g., Eigensolve, (Fortune, 2001). An alternative way of computing the polynomial roots is based on so-called Weyl's geometric construction, see, e.g., Wilf, (1978) or Pan, (1997) and the references therein. The method will be briefly explained in section 5.1.

3 POLES AND ZEROS OF TIME DELAY SYSTEMS

Let us consider the general form of linear SISO TDS using the Stieltjes integrals (this special form of integral allows to involve both lumped and distributed delays into one convolution formula)

$$\begin{aligned} &\frac{d}{dt}\left[\mathbf{x}(t)+\int_0^T d\mathbf{D}(\tau)\mathbf{x}(t-\tau)\right]=\int_0^T d\mathbf{A}(\tau)\mathbf{x}(t-\tau)+\int_0^T d\mathbf{B}(\tau)u(t-\tau), \\ &y(t)=\mathbf{C}\mathbf{x}(t) \end{aligned} \tag{6}$$

where the matrices $\mathbf{A}(\tau)$, $\mathbf{B}(\tau)$ and $\mathbf{D}(\tau)$ are functional matrices of delay distribution with the upper bound T. Unlike the state of the classical system (1) that is given by the actual values of the state variables, the state of system (6) is given by trajectories of the state variables on the interval $[t\text{-}T, t]$, see Zítek (1988), Diekman, et al, (1995).

If $\mathbf{D}(\tau)=\mathbf{0}$ the system is called retarded while if $\mathbf{D}(\tau)\neq\mathbf{0}$ the system is called neutral (Hale and Verduyn Lunel, 1993). The retarded systems are more common in control engineering than neutral systems since their theory is more developed and some of their features are similar to the features of

classical systems, e.g., the finite number of unstable poles, insensitivity of the poles to infinitesimal changes of the model parameters and so on. The retarded systems are used for describing the plants with after-effects, latencies, transportation phenomenon or distributed parameters, e.g., heat-transfer or chemical processes. Using the classical systems (1) for description of such plants results in high order model with many artificial state variables (unmeasured, often without physical meaning). On the other hand, involving the delays in the model results in considerably lower order model with the state variables often identical with the available system outputs. Neutral systems are largely used for describing lossless propagation phenomena, see Niculescu, (2001), which is encountered, e.g., in modelling of distributed networks.

In case of zero initial conditions the Laplace transform of (6) can be expressed in the following form

$$\begin{aligned} s\left[\mathbf{I}+\mathbf{D}(s)\right]\mathbf{x}(s) &= \mathbf{A}(s)\mathbf{x}(s)+\mathbf{B}(s)u(s), \\ y(s) &= \mathbf{C}\mathbf{x}(s) \end{aligned} \tag{7}$$

where

$$\mathbf{A}(s)=\int_0^T \exp(-s\tau)d\mathbf{A}(\tau), \quad \mathbf{B}(s)=\int_0^T \exp(-s\tau)d\mathbf{B}(\tau), \quad \mathbf{D}(s)=\int_0^T \exp(-s\tau)d\mathbf{D}(\tau) \tag{8}$$

Laplace form (7) is advantageous, since it allows to use equivalent matrix operations like in the case of classical system description. Analogously to this case of classical system (1), the formulation (7) can be transformed into input-output transfer function

$$G(s)=\frac{y(s)}{u(s)}=\frac{N(s)d(s)}{M(s)} \tag{9}$$

where

$$N(s)=\frac{1}{d(s)}\mathbf{C}\,\mathrm{adj}\left\{s[\mathbf{I}+\mathbf{D}(s)]-\mathbf{A}(s)\right\}\mathbf{B}(s) \tag{10}$$

$$M(s)=\det\left\{s[\mathbf{I}+\mathbf{D}(s)]-\mathbf{A}(s)\right\} \tag{11}$$

are quasipolynomials as a rule and term d(s) represents the distribution of system input delay. Analogously to the case of classical systems the poles and zeros of TDS determine the features of the system dynamics. The system poles are the solutions of the characteristic equation

$$M(s)=\sum_{i=0}^{n} a_i s^i Q_i(s)=0 \tag{12}$$

where n is system order, a_i are the coefficients and $Q_i(s)$, i=1..n are the corresponding delay distribution terms of quasipolynomial M(s). The system zeros are the solutions of the equation

$$N(s)=\sum_{i=0}^{m} b_i s^i P_i(s)=0 \tag{13}$$

where $m\le n$, b_i are the coefficients and $P_i(s)$, i=1..m are the corresponding delay distribution terms of quasipolynomial N(s), respectively. Unlike the classical system (1), TDS has infinitely many poles because the characteristic equation (12) is transcendental. The number of zeros depends on the form of N(s), which does not have to be quasipolynomial as a rule (depending on the matrices $\mathbf{B}(\tau)$ and $\mathbf{C}$). If at least one of the terms $P_i(s)$ is not equal to zero, equation (13) is also transcendental with infinitely many roots.

The physical meaning of poles and zeros of TDS is equivalent to the meaning of poles and zeros of classical system, at least in case of retarded systems. Provided that no zero-pole cancellation occurs, every pole generates a natural mode in the system responses. Even though the number of poles and zeros is infinite, only few of them are decisive in the system dynamics. As regards the system

poles, their roles in the system dynamics depend on the distances of the poles from the imaginary axis and from the *s*-plane origin. The dominant poles are those located in the closest positions to the *s*-plane origin and to the stability boundary. Nevertheless, it is difficult to claim either the distance from the origin or the distance from the imaginary axis is more important in determining the particular pole significance. An alternative method for evaluating pole dominance based on residue evaluation has been proposed in Zítek and Vyhlídal, (2002). Anyway, if at least one pole is located to the right from the stability boundary, the system is unstable. As regards the system zeros, the analogous evaluation to system poles can be done, i.e., the most important zeros from the infinite set are those located in the closest positions to the stability boundary. The sign of the zeros is also important, however, not from the stability point of view. The zeros with plus sign are called nonminimum-phase zeros and they are responsible for the undershoots in the system responses.

The distribution of system poles differs significantly with respect to the character of TDS. If the system is retarded, the number of poles located to the left from the vertical line drawn in any real α is always finite, see Hale and Verduyn Lunel, (1993). The poles of a retarded system are usually distributed as a finite number of chains asymptotically departing to the upper-left direction with increasing distance of the poles from the *s*-plane origin. The poles of neutral systems are distributed within the lines drawn in some specific real α and β. The poles of a neutral system are also distributed as a finite number of asymptotic chains. However, these chains asymptote to the chains of essential spectrum, see e.g. Avelar and Hale, (1980). The essential spectrum corresponds to the spectrum of difference equation

$$\mathbf{x}(t) + \int_0^T d\mathbf{D}(\tau)\mathbf{x}(t-\tau) = 0 \tag{14}$$

which is given by the solutions of the following equation

$$M_e(s) = \det\{\mathbf{I} + \mathbf{D}(s)\} = 0 \tag{15}$$

where $M_e(s)$ is called exponential polynomial

Since the stability of some difference equations is very sensitive to the infinitesimal changes in delays, the concept of so-called strong stability has been introduced in analysis of neutral systems, see Hale and Verduyn Lunel, (2002). The difference equation is strongly stable if it is stable independently on the changes in delays. The analysis of stability of difference equation (14) is very important, since the essential spectrum asymptotically determines the spectrum of the neutral system. If the difference equation (14) is unstable, it means that the neutral system is not only unstable but it is unstable with infinitely many unstable poles.

4 COMPUTING THE POLES OF RETARDED TDS

The problem of computing poles of TDS has remained unsolved untill the nineties of the last century when two algorithms appeared both based on discretization of the continuous TDS. The first algorithm is based on discretization of the solution operator (Engelborghs and Roose, 1999, 2002) in which the discretization is performed by means of linear multi-step methods. The second algorithm is based on discretization of the infinitesimal generator of the semigroup, (Ford and Wulf, 1998), (Wulf and Ford, 2000) where the Euler explicit method and trapezoidal rule is used for the discretization. The solution operator and the infinitesimal generator of the semigroup are the concepts used in functional theory of TDS, in which the functional state of the system is directly involved, see Hale and Verduyn Lunel, (1993) or Diekman, et al, (1995). The method based on on discretization of the solution operator has been implemented in the Matlab package DDE-BIFTOOL (Engelborghs, 2001). The modifications of both methods consisting in using stiff accurate Runge-Kutta method Radau IIA can be found in Breda, Maset and Vermiglio, (2002).

As has been mentioned in section 2, many of the numerical methods for computing the polynomial roots are based on transforming the problem into computing the eigenvalues of the companion matrix of the polynomial. Such an approach cannot be directly used for computing the roots of quasipolynomials because the terms corresponding to the particular powers of *s*, see (12), are of functional form. The problem of computing the roots of the quasipolynomials corresponding to

retarded TDS can be solved by approximating the continuous system by a discrete system and applying, e.g., δ-transform, see Zítek and Petrová, (2002) and the references therein. Consequently, the system characteristic functions $M_d(\delta)$ and $N_d(\delta)$ are of the polynomial forms and the classical approach for computing polynomial roots can be used. It should be recalled that the roots of these polynomials are located in δ-domain. The stability boundary of δ-domain corresponding to the imaginary axis of s-plane is a circle located in the left-hand side of the complex plane touching the imaginary axis in the origin. The radius of the stability boundary circle depends on the chosen step h of the discretization method. In order to obtain the approximations of the poles and zeros of TDS, the computed roots δ_i should be transformed into the s-plane using the appropriate formula $s_i=f(\delta_i, h)$. Analogously to the methods mentioned in section 4, only the right-most poles and zeros of TDS can be obtained using this method. The drawback of this method is given by the fact that the polynomials $M_d(\delta)$ and $N_d(\delta)$ are of very high degree if $h<<\tau_{\max}$, where $\tau_{\max}$ is the delay with the maximum length. The coefficients of high degree $M_d(\delta)$ and $N_d(\delta)$ are influenced by truncation errors which is likely to result in a numerical failure providing incorrect results. Consequently, step h should not be chosen too small which is contradictory requirement to the requirement to achieve good approximation of the rightmost roots (the smaller h, the closer δ to s).

The indispensable drawback of the mentioned methods based on discretization is given by the fact that only the rightmost poles (zeros) of the system are approximated. The accuracy of the approximation depends on the chosen numerical method and particularly on the chosen step of the discretization. On the other hand the rightmost poles are the most important ones in the system dynamics, therefore the methods often provide satisfactory results for evaluation of the dominant modes of retarded TDS. However, to the best of the authors knowledge, the analogous methods for computing the poles of neutral systems has not appeared yet. The objective of the paper is to introduce two algorithms that can be used for computing both poles and zeros of continuous TDS located inside a selected region of the complex plane both based on computing the quasipolynomial roots.

5 COMPUTING THE QUASIPOLYNOMIAL ROOTS

5.1 *Weyl's construction with argument principle based test*

The basic idea of Weyl's construction used for computing the polynomial roots is as follows: On the complex plane, the search for the roots starts with an initial suspect region $\mathcal{D}=[\beta_{\min},\beta_{\max}]\times[\omega_{\min},\omega_{\max}]$, preferably square, containing the polynomial roots. Then the region is partitioned into four congruent subregions, see Fig. 3. At the centre of each of them, the proximity test is performed (Henrici, 1974), i.e., a distance of the closest root from the centre is estimated. If this distance exceeds the length of the diagonal of the subregion then the square does not contain any roots and is discarded. If the result is converse, the subregion is called suspect and undergoes the same recursive process of partitioning into four subregions. Subsequently the proximity tests are performed at the centers of those smaller subregions. The polynomial roots lying in each suspect region are approximated by its centre with errors bounded by the half-length of its diagonal. Thus, in k iteration steps, the approximation errors do not exceed $0.5\mathrm{diag}(\mathcal{D}_k)/2^k$, where $\mathcal{D}_k$ represents the actual suspect region.

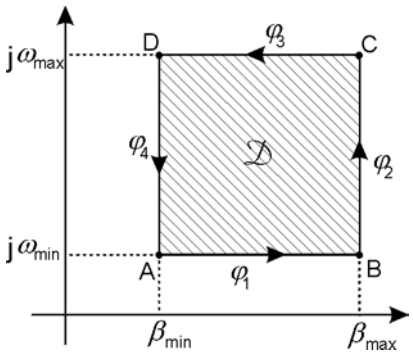

Fig. 1 Region $\mathcal{D}$ in the s-plane

Let us use the Weyl's construction for locating the roots of qusipolynomials. Instead of the proximity test, let us use the argument principle, which holds for any analytic function including quasi-polynomials, see ,e.g., (El'sgol'ts and Norkin, 1973). Considering a quasipolynomial, e.g., in the form of $M(s)$, see (12), the number of its roots located inside the region $\mathcal{D}$ with the boundary φ is given by the following formula

$$N_{\mathcal{D}}=\frac{1}{2\pi}\Delta_{\varphi}\arg M(s)=\frac{1}{2\pi\mathrm{j}}\int_{\varphi}\frac{M'(s)}{M(s)}ds \tag{16}$$

where $M'(s) = dM(s)/ds$. As can be seen, the number of roots $N_{\mathfrak{D}}$ can be computed directly by evaluating the integral in (16), as it has been used in Zítek and Vyhlídal, (2000). However, the numerical evaluation of the integral becomes unacceptably time and memory demanding in case that M'/M acquires complicated form. It usually happens if $M(s)$ is of higher order with the delays of different distributions.

The other possibility to evaluate $N_{\mathfrak{D}}$ is based on a graphical evaluation of formula (16) which claims that $N_{\mathfrak{D}}$ is equal $1/2\pi$ times of the variation of the argument

$$\Phi_M(s) = \arg(M(s)) = \arctan\frac{\mathrm{Im}(M(s))}{\mathrm{Re}(M(s))} \tag{17}$$

as s moves once around φ in the counter-clockwise sense. Considering the features of the trigonometric function arc-tangent, the argument Φ_M results in the form of $\mathrm{mod}(\pi/2)$. Thus the argument varies between -π/2 and π/2. The algorithm of evaluating the change of $\Phi_M(s)$ as s moves around the boundary of the region $\mathfrak{D}$ is quite simple and easy to implement. Starting from a certain arbitrarily chosen point on φ, e.g. A, see Fig. 1, and varying s continuously in the counter-clockwise sense, $\Phi_M(s)$ changes piecewise continuously until it reaches either π/2 or -π/2. Then $\Phi_M(s)$ changes discontinuously to ±π/2 (the sign is opposite to the sign of previous π/2) and it changes piecewise continuously again until it reaches one of the boundary values ±π/2. In this way $\Phi_M(s)$ is evaluated around φ, passing the points B, C and D and coming back to the starting point A, see Fig. 1. Since $M(\varphi(s))$ is closed contour in the complex plane, the starting and ending values of $\Phi_M(s)$ has to be equal. As the result of the argument evaluation a set of curves starting and ending at ±π/2 is obtained providing the argument change π, -π or 0, see Fig. 2. In order to obtain the final $\Delta_\varphi\Phi(s)$ the changes in arguments on these curves, i.e., π, -π or 0, are to be summed.

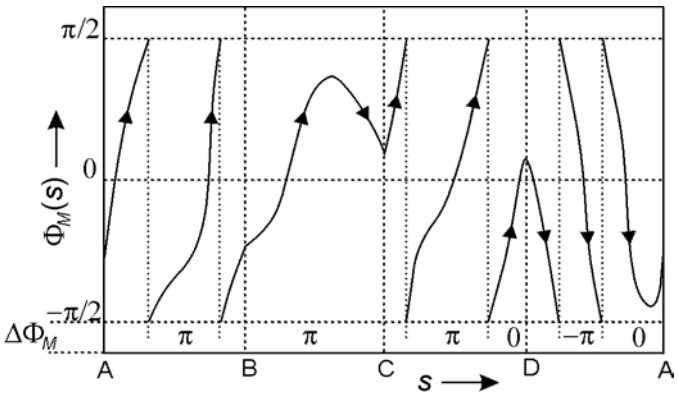

Fig. 2 Variation of $\Phi_M(s)$ along the closed contour φ, see Fig. 1

Obviously, the final change of the argument, whose variation along the contour φ is shown in Fig. 2, is $\Delta_\varphi\Phi(s) = 2\pi$, which means that there is one root of $M(s)$ in the region $\mathfrak{D}$., $N_{\mathfrak{D}}$=1. The presented algorithm for computing $N_{\mathfrak{D}}$, which has been used in Luzyanina and Roose, (1996) to detect a bifurcation points for delay differential equations, is more suitable for computer implementation than the algorithm based on the evaluation of the integral in (16) due to its distinctly lower number of numerical operations to be performed.

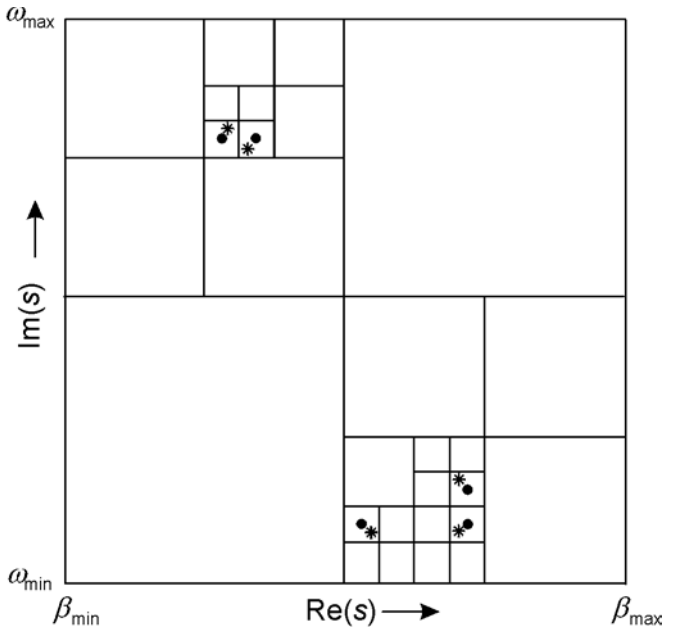

Fig. 3 Locating the roots by Weyl's algorithm
asterisks - quasipolynomial roots,
black dots - approximations of the roots

Using the argument principle in the particular suspect subregion, the result is not only whether or not there are some roots in the subregion, but also the number of the roots in the subregion is obtained. The procedure of recursive partitioning of the suspect regions is shown in Fig. 3. There are 5 roots in the region $\mathfrak{D}$, marked by asterisks and their approximations obtained using Weyl's construction are marked by the black dots. The first step of the algorithm consists in computing the number of roots in $\mathfrak{D}=[\beta_{\min},\beta_{\max}]\times[\omega_{\min},\omega_{\max}]$ using the described argument principle rule. Then Weyl's construction with the argument principle based test is applied recursively until the root approximation with required accuracy is achieved

The accuracy of the final root approximations, which is given by the half of the final (smallest) subregion, can be enhanced by means of applying, e.g., Newton's iteration method

$$s_{i,k+1} = s_{i,k} - \frac{M(s_{i,k})}{M'(s_{i,k})} \tag{18}$$

where s_i, $i=1, 2, \ldots$ are the root of $M(s)$ and $k=0,1,\ldots$ is the step of the Newton's iteration. Suppose that the root approximations $s_{i,0}$ resulting from the algorithm based on Weil's construction are close to the roots s_i then the approximations $s_{i,k}$ are likely to converge much faster to the roots s_i than in case of further carrying on Weil's algorithm.

5.2 *Algorithm based on quasipolynomial function mapping in complex plane*

The quasipolynomial $M(s)$ as a function of the complex variable $s=\beta+\mathrm{j}\omega$ can be split up into real and imaginary parts

$$M(\beta,\omega) = R(\beta,\omega) + \mathrm{j}I(\beta,\omega) \tag{19}$$

where $R(\beta,\omega)=\mathrm{Re}\{M(\beta,\omega)\}$ and $I(\beta,\omega)=\mathrm{Im}\{M(\beta,\omega)\}$. Consequently, equation (12) can be split up into

$$\begin{aligned} R(\beta,\omega) &= 0 \\ I(\beta,\omega) &= 0 \end{aligned} \tag{20}$$

From the geometric point of view, the roots of $M(s)$ are the intersection points of the curves described by the implicit functions $R(\beta,\omega)=0$ and $I(\beta,\omega)=0$. Mapping the surfaces $R(\beta,\omega)$ and $I(\beta,\omega)$ over the region $\mathfrak{D}=[\beta_{\min},\beta_{\max}]\times[\omega_{\min},\omega_{\max}]$ the equipotential contours are given by the intersections of the surfaces with the s-plane. Taking into account this geometric interpretation of equation (20) the algorithm for locating the roots of $M(s)$ can be summarized as follows:

1 The region $\mathfrak{D}=[\beta_{\min},\beta_{\max}]\times[\omega_{\min},\omega_{\max}]$, where the roots of $M(s)$ are to be computed, is defined.
2 With a chosen increment Δ, the region $\mathfrak{D}$ is covered by the grid of nodes $d_{ij} = \beta_j\times\omega_i$, $i=1..\,|\omega_{\max}-\omega_{\min}|/\Delta$, $j=1..\,|\beta_{\max}-\beta_{\min}|/\Delta$ with stepwise incrementing co-ordinates β and ω.
3 For each node d_{ij}, the values of $R(\beta_i,\omega_j)$ and $I(\beta_i,\omega_j)$ are evaluated.
4 Using an interpolation method, the intersections of the surfaces R and I with the s-plane, i.e., contours $R(\beta,\omega)=0$ and $I(\beta,\omega)=0$, are mapped.
5 The intersection points of the contours indicate the approximate positions of the poles.
6 The accuracy of the root approximations is enhanced by means of Newton's iteration method.

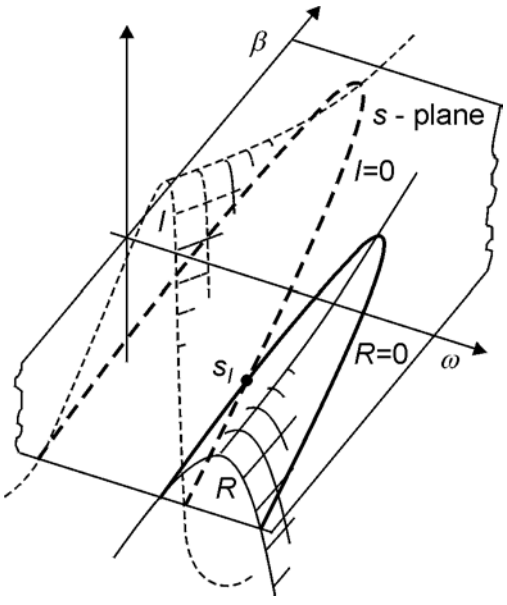

Fig. 4 The principle of locating $M(s)$ roots
R=Re(M)=0 - solid,
I=Im(M) = 0 - dashed

Regarding the computational effort, the algorithm is rather demanding. A lot of calculation has to be done, especially if the region $\mathcal{D}$ is chosen too large and there are many roots located inside the region. However, using the Matlab functions defined for the matrices and 3D graph functions, the approximate locations of the roots given as the intersections of the contours can be found relatively fast. The crucial role in the length of the root finding process is played by the increment Δ, which should be chosen according to the expected frequency range of the contours f and particularly to the size of the region $\mathcal{D}$. The smaller Δ is, the more precise is the estimate of the positions of roots. On the other hand, too small Δ results in too long duration of the rootfinding process. To solve the problem of choosing Δ, it is necessary to balance these two contradictory requirements. With respect to the size of the region $\mathcal{D}$ the increment Δ should not be less than $10^{-5}\left|\beta_{\max}-\beta_{\min}\right|\left|\omega_{\max}-\omega_{\min}\right|$. If the increment is chosen too large, the contours are not approximated well which may result in omitting some of the roots. In such a case, the increment should be decreased. It is advisable also to divide the original region into smaller ones and to carry out the computation separately for each new region with the decreased increment.

5.3 *Implementation of the quasipolynomial rootfinding algorithm in Matlab*

The algorithm based on Weyl's construction is suitable to be applied if the number of roots in the region $\mathcal{D}$ is not too large. Although the algorithm of recursive dividing of the suspect regions is quite simple, its computer implementation requires using an elaborate approach that would guarantee that none piece of the region with some roots remains unsearched. On the other hand, the second quasipolynomial rootfinding algorithm based on $M(s)$ mapping can be easily implemented by means of available Matlab functions. Therefore this algorithm has been chosen for computer implementation. Using the Symbolic Math Toolbox, quasipolynomial $M(s)$ can be written directly in the operator s, as it results from performing Laplace transform of TDS, and no special form of $M(s)$ is required. The quasipolynomial rootfinder has been implemented in Matlab and is available as the function with the following syntax

P=qroots(M,D,Δ,ε) (21)

where M is the quasipolynomial of the symbolic variable s, D is the suspect region, Δ is the increment of the grid, ε is desired maximal error of the root approximation and P is the vector of computed roots. The algorithm implementation will be shown in the following examples.

5.4 *The limitations of the algorithm use*

Also the quasipolynomials, analogously to the polynomials, may be ill conditioned. The danger of this numerical risky form increases with the of quasipolynomial degree is high. If a quasipolynomial is ill conditioned, the contours described by $R(\beta,\omega) = 0$ and $I(\beta,\omega) = 0$ are not likely to be compact and smooth (at least on a part of the region $\mathcal{D}$) and the rootfinding process fails. On the other hand, as has been mentioned, using models with various distribution of delays result in low order models as a rule, for which such a numerical risk is low.

Another problem of the algorithm is given by the use of Newton's method. If two roots are close to each other, the Newton iteration can incorrectly result in a double root. If such a failure is suspected, the computation should be run again with the region defined in the vicinity of these poles with distinctly smaller Δ.

5.5 *Example 1 - poles of a retarded system*

Let us consider a retarded system with the following functional matrix

$$\mathbf{A}(s)=\frac{1}{15}\begin{bmatrix} -\mathrm{e}^{-9s} & \mathrm{e}^{-4s} & \mathrm{e}^{-6s} \\ \dfrac{\mathrm{e}^{-5s}-\mathrm{e}^{-12s}}{7s} & -\mathrm{e}^{-4s} & \mathrm{e}^{-3s} \\ \mathrm{e}^{-7s} & \dfrac{\mathrm{e}^{-6s}-\mathrm{e}^{-18s}}{12s} & \mathrm{e}^{-5s} \end{bmatrix} \tag{22}$$

It can be seen that there are encountered both lumped and linear distributed delays in matrix $\mathbf{A}(s)$. In order to analyse the system dynamics the system poles, i.e., the roots of quasipolynomial $M(s) = \det\{s\mathbf{I} - \mathbf{A}(s)\}$ are to be found. Using the function *qroots*, according to its syntax (21) the task can be performed in Matlab as follows

```
> syms s
>A=[-exp(-9*s)  exp(-4*s)  exp(-6*s); (exp(-5*s)-exp(-12*s))/7/s  -exp(-4*s)  exp(-3*s);
  exp(-7*s)  (exp(-6*s)-exp(-18*s))/12/s  exp(-5*s)]
>P=qroots(det(s*eye(3)-A), [-0.8 0.4 0 3], 0.01, 0.001)
```

where the command *syms s* produces symbolic variable *s*.

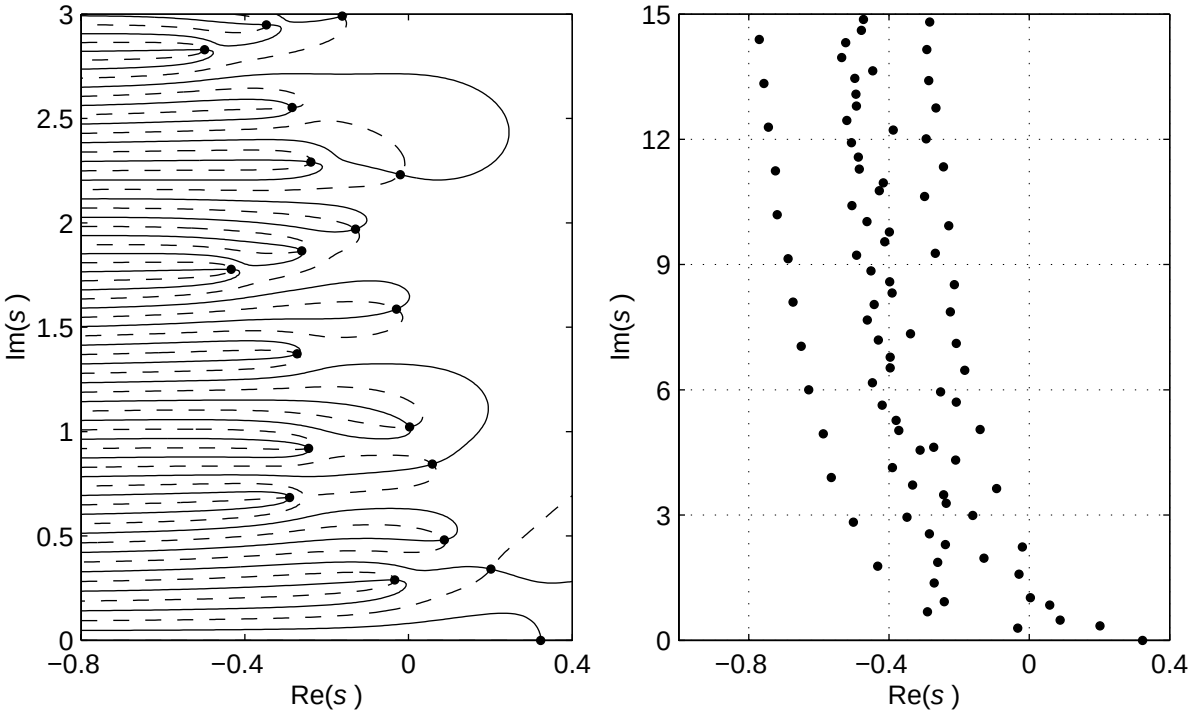

Fig. 5 Distribution of the poles of the retarded system with matrix $\mathbf{A}(s)$ given by (22).
left part - $R(\beta,\omega) = 0$ (solid), $I(\beta,\omega) = 0$ (dashed)
right part - distribution of the poles in the enlarged region $\mathfrak{D}$

The resulting distribution of the system poles can be seen in the left part of Fig. 5. As can be seen the poles are given by the intersections of the contours described by the functions $R(\beta,\omega) = 0$ (solid) and $I(\beta,\omega) = 0$ (dashed). In the right part of Fig. 5, the distribution of poles is shown located in the enlarged region $\mathfrak{D}$. Obviously, the poles tend to depart to the left from the imaginary axis, which is the basic feature of retarded systems. Since there are 9 poles located to the right from the imaginary axis, the system is unstable. It should be noted that the complex plane is symmetric in real axis, thus the complex poles are conjugate. Therefore it is advisable to define the region $\mathfrak{D}$ only on a half plane.

5.4 *Example 2 - poles of a neutral system*

Let us consider a neutral system with the matrices $\mathbf{A}(s)$, $\mathbf{D}(s)=\mathbf{A}(s)$, where $\mathbf{A}(s)$ is given by (22). Analogously to the Example 1, the system poles, i.e. the roots of quasipolynomial $M(s) = \det\{s[\mathbf{I} + \mathbf{A}(s)] - \mathbf{A}(s)\}$, result from the use of the function *qroots*. The result of the procedure can be seen in the left part of Fig. 6. In the right part of the figure, the distribution of the poles is shown in the enlarged region. Besides the spectrum of the neutral system poles (marked with black dots) also the essential spectrum of the system is show (marked with asterisks), given as the roots of the exponential polynomial $M_e(s) = \det\{\mathbf{I} + \mathbf{A}(s)\}$. It can be seen that the poles of neutral system close to the *s*-plane origin have the similar distribution to the poles of retarded system from Example 1, compare Fig.5 and Fig.6. However, with increasing distance of the poles from the *s*-plane

origin, the poles of neutral system asymptote to the values of essential spectrum, which is the basic feature of the neutral systems. Since a part of the essential spectrum is located to the right from the stability boundary, the neutral system is not only unstable, but it is unstable with infinitely many unstable poles.

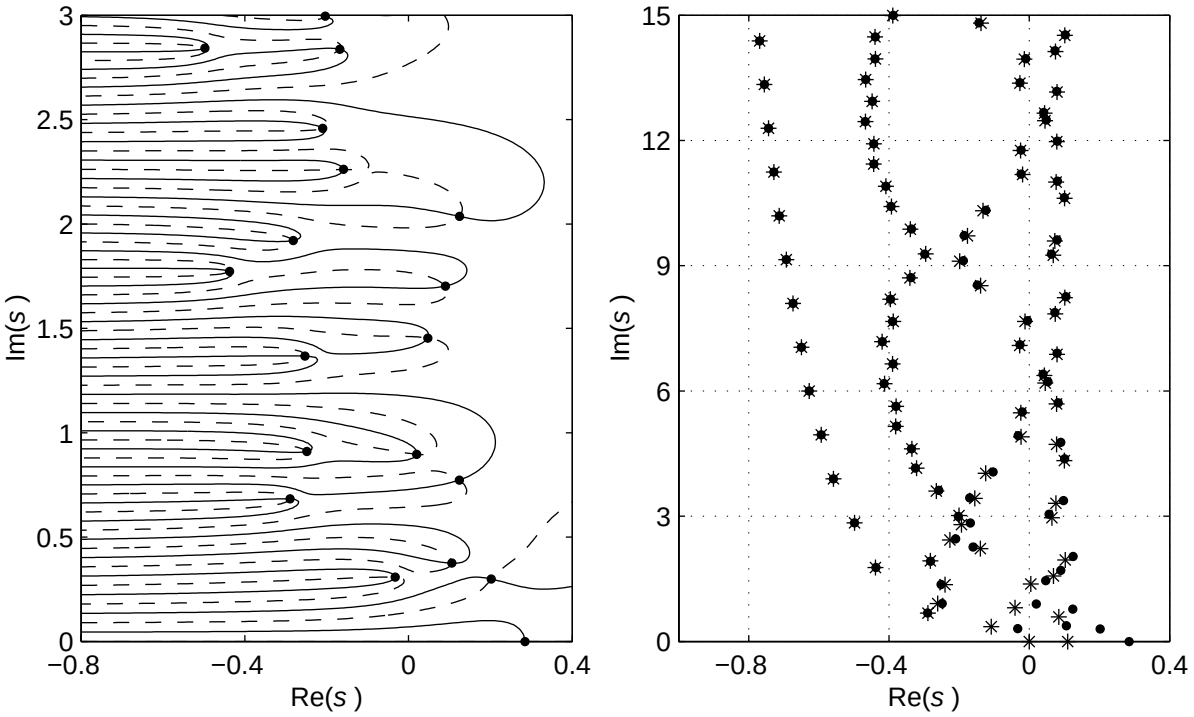

Fig. 6 Distribution of the poles of the neutral system with the matrices $\mathbf{A}(s)$ and $\mathbf{D}(s)=\mathbf{A}(s)$, where $\mathbf{A}(s)$ given by (22), left part - $R(\beta,\omega) = 0$ (solid), $I(\beta,\omega) = 0$ (dashed)
right part - distribution of the poles on enlarged region $\mathfrak{D}$, dots - spectrum of neutral system, asterisks - essential spectrum

6 CONCLUSIONS

Two original algorithms for computing the quasipolynomial roots located inside a selected region in complex plane have been introduced. The first algorithm is based on application of Weyl's construction and argument principle. The second algorithm is based on the geometric interpretation of the quasipolynomial function in the complex plane. The second algorithm has been implemented in Matlab and is available as the function *qroots* with the syntax given by (21). Symbolic calculations available via Symbolic Math Toolbox has been used in the function *qroots* as well as the Matlab 3D-graph functions. The designed quasipolynomial rootfinder implemented in Matlab provides the significant contribution to the analysis of time delay systems where the characteristic functions are of quasipolynomial forms. By means of the rootfinder *qroots*, poles as well as zeros of both retarded and neutral time delay systems can be computed, which simplifies the analysis of the system dynamics. It also allows further studying of the features of time delay systems resulting in deeper understanding of their fundamentals.

Acknowledgement: The presented research was supported by the Ministry of Education of the Czech Republic under Project LN00B096

REFERENCES

Anderson, E., Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, (1999) *LAPACK User's Guide*, http://www.netlib.org/lapack/lug/lapack_lug.html Third Edition, SIAM, Philadelphia.

Bai, Z., Demmel, J., Dongarra, J., Ruhe, A. and van der Vorst, H. - editors, (2000), *Templates for the solution of Algebraic Eigenvalue Problems: A Practical Guide,* SIAM, Philadelphia, (@Book, http://www.cs.utk.edu/~dongarra/etemplates/index.html)

Diekmann, O., S.A. Van Gils, S.M. Verduyn Lunel and H.O. Walther (1995) *Delay Equations Functional, Complex and Nonlinear Analysis*. Springer Verlag, AMS Series, No. 110

Engelborghs, K. (2000). *DDE-BIFTOOL: a Matlab package for bifurcation analysis of delay differential equations*. TW Report 305, Department of Computer Science, Katholike Universiteit Leuven, Belgium. Available from http://www.cs.kuleuven.ac.be/~koen/delay/ddebiftool.shtml.

Engelborghs, K. and Roose, D. (1999). Numerical computation of stability and detection of hopf bifurcations of steady state solutions of delay differential equations. *Advanced in Computational Mathematics,* 10(3-4), pp. 271-289

Engelborghs, K., Roose, D. (2002), On stability of LMS-methods and characteristic roots of delay differential equations, *SIAM Journal of Numerical Analysis*, Vol. 40, number 2, pp. 629-650

Ford, N. J. and Wulf, V, (1998), Embedding of a numerical solution of a DDE into the numerical solution of a system of ODEs, *Technical Report*, Manchester Centre for Computational Mathematics, University of Manchester

Fortune, S., (2001), Polynomial root finding using iterated eigenvalue computation, *ISSAC 2001,* pp. 121-128.

Goodwin, G. C., Graebe, S. F., Salgado, M. E., (2001), *Control System Design*, Prentice-Hall, Inc., New Jersey

Luzyanina, T., Engelborghs, K., Lust, K. and Roose, D. (1997), Computation, continuation and bifurcation analysis of periodic solutions of delay differential equations *International Journal of Bifurcation and Chaos*, vol. 7, number 11, pp. 2547-2560

Niculescu, S. I., (2001), *Delay Effects on Stability, A Robust Control Approach*. Springer-Verlag London Limeted.

Pan, V, (1997), Solving a polynomial equation: some history and recent progress, *SIAM Rewiev* 39:2, pp.187-220

Wilf, H. S., (1978), A Global Bisection Algorithm for Computing the Zeros of Polynomials in the Complex Plane, *Journal of the ACM*, vol. 25, pp. 415-420

Wilkinson, J. H. (1984). The perfidious polynomial, in *Studies in numerical analysis*,vol. 24 of MAA Stud. Math., Math. Assoc. America, Washington, DC.

Wilkinson, J. H., (1965), *The Algebraic Eigenvalue Problem*, Oxford University Press

Wulf, V. and Ford, N. J., (2000), *Numerical Hopf Bifurcation for a class of delay differential equations*, Journal of Computational and Applied Mathematics, vol. 115, pp. 601-616

Zítek, P. and Petrová, R., (2002), Discrete Approximation of Anisochronic Systems Using Delta Transform, In *Proc. of 5th International Conference Process Control 2002*, Kouty nad Desnou, Czech Republic

Zítek, P. and Vyhlídal, T. (2002), Dominant eigenvalue placement for Time Delay Systems, In *Proc. of Controlo 2002, 5th Portuguese Conference on Automatic Control, University of Aveiro, Portugal*

Zítek, P., (1998). Time Delay Control System Design Using Functional State Models. *CTU Reports* No.1/1998**,** CTU Prague, 93p.

Zítek, P., Vyhlídal, T., (2000), State Feedback Control of Time Delay System: Conformal Mapping Aided Design In: *2nd IFAC Workshop on Linear Time Delay Systems*. Ancona, Universita di Ancona, pp. 146-151

Contact:

Tomáš Vyhlídal
Center for Applied Cybernetics
Faculty of Mechanical Eng.
Czech Technical University in Prague
166 07 Praha 6
Czech Republic
vyhlidal@fsid.cvut.cz

Pavel Zítek
Institute of Instrumentation and Control Eng.
Faculty of Mechanical Eng.
Czech Technical University in Prague
166 07 Praha 6
Czech Republic
zitek@fsid.cvut.cz