

Using Orthogonal Rational Functions for System Identification

P. Van gucht A. Bultheel

Report TW 314, September 2000



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Using Orthogonal Rational Functions for System Identification

P. Van gucht A. Bultheel

Report TW 314, September 2000

Department of Computer Science, K.U.Leuven

We establish two different algorithms that return an approximation of a system transfer function if input and output data are given. The model we use, is written as a linear combination of orthogonal rational functions (ORF), with respect to a weight function which depends on the input data.

In the first algorithm a continuous inner product leads to a least squares problem with a well conditioned matrix. Due to the fact that the data is finite we cannot solve the least squares problem recursively. This problem can be solved if we use a discrete inner product. For this approach however we need the z -transform of the input and output in some points.

Abstract

Keywords : system identification, orthogonal rational functions, least squares.
AMS(MOS) Classification : Primary : 93B30, Secondary : 42C05, 93E24

Using Orthogonal Rational Functions for System Identification

P. Van gucht and A. Bultheel

Dept. Computerscience, Celestijnenlaan 200A, 3001 Leuven.

`{patrick.vangucht–adhemar.bultheel}@cs.kuleuven.ac.be`

Abstract

We establish two different algorithms that return an approximation of a system transfer function if input and output data are given. The model we use, is written as a linear combination of orthogonal rational functions (ORF), with respect to a weight function which depends on the input data.

In the first algorithm a continuous inner product leads to a least squares problem with a well conditioned matrix. Due to the fact that the data is finite we cannot solve the least squares problem recursively. This problem can be solved if we use a discrete inner product. For this approach however we need the z -transform of the input and output in some points.

1 Introduction

In the area of system identification, people are looking for a model that fits the input and output data in the best possible way and with the least possible complexity. Many different solutions are given and a good reference is [6].

One of the most elegant models that gives nice results is the FIR-model, where the Fourier coefficients of the transfer function are approximated. Problems can occur when the unknown system has a slightly damped pole, because the number of coefficients grows very fast if we like to handle the dynamics of this pole in our model. This is the main reason why the knowledge of one or more (approximations of the) poles of the unknown system should be incorporated in the basisfunctions we use to build up the approximating model. This will lead to a considerable reduction in the coefficients needed to give a *good* approximation. This motivates the use of rational functions with prescribed poles in system identification.

The reason we use orthogonal rational functions (ORF) is numerical stability.

The most simple case is where one pole is incorporated, the so-called Laguerre model (see, e.g., [20]). The step to two complex conjugated poles, the Kautz basis, is easy (see, e.g., [21]).

The case where a number of poles is cyclically repeated, is extensively investigated by Van den Hof, Bokor and co-workers [5, 12, 14]. They were the first to describe the ORF by a state space realization, which makes the MIMO-case easier to handle. Ninness et al. [1, 2, 8, 9, 11, 10, 7] generalized this cyclic case to the more general case of an arbitrary sequence of poles which results in the Malmquist basis [22, pp. 224]. The state space approach of Van den Hof et al. [5, 14], is already generalized to the case of an arbitrary sequence of poles [15].

All these results are about ORF with respect to the Lebesgue measure. However the least squares uses a weight depending on the input data. So it would be interesting to incorporate this weight in our ORF-basis so that the least squares can easily be computed. Therefore we need some theory about orthogonal rational functions with respect to a given measure.

In [3] the abstract theory of orthogonal rational functions (ORF) with respect to a general measure and with prescribed poles, lying outside the unit disc, is well developed. These functions are in fact a generalization of the Szegő polynomials (see [13]), in the sense that they reduce to the Szegő polynomials if we place all the poles at infinity.

In system identification, it is however necessary to have poles inside the unit disc for stability reasons. This case can be found in [17], which is a slight adaptation of the theory found in [3] to the case where all the poles lie inside the unit disc.

This paper is built up as follows: in the next section we give a summary of [17] with the results of the theory of ORF with given poles inside the unit disc. The third section deals with system identification using these ORF. In a fourth section a numerical example is given, which illustrates that this approach is satisfactory for the case where we have knowledge about the poles. Some conclusions are made in section five.

2 Orthogonal rational functions, analytic outside the unit disc

2.1 Summary

We only state some results that can be found in [17]. In the next paragraph some notation and definitions are given. Paragraph three contains a recurrence relation for the ORF. If the moments are finite and known, a simple algorithm is given in paragraph four to compute the reflection coefficients.

2.2 Preliminaries

The field of real and complex numbers is denoted as \mathbb{R} and \mathbb{C} respectively. The integers are denoted by \mathbb{Z} and the non-negative integers by \mathbb{N} .

The unit circle, its exterior and interior are denoted by

$$\mathbb{T} = \{z \in \mathbb{C} : |z| = 1\}, \quad \mathbb{E} = \{z \in \mathbb{C} : |z| > 1\} \text{ and } \mathbb{D} = \{z \in \mathbb{C} : |z| < 1\},$$

respectively. The closed unit disc is defined as $\overline{\mathbb{D}} = \mathbb{T} \cup \mathbb{D}$.

Let μ be a positive measure on \mathbb{T} . The inner product is denoted by

$$\langle f, g \rangle_{\mu} = \int_{\mathbb{T}} f(z)g_*(z)d\mu(z), \quad f, g \in L^2(\mu). \quad (1)$$

We define the moments as

$$c_k = \langle 1, z^k \rangle_{\mu}, \quad k \in \mathbb{Z}. \quad (2)$$

The set of functions analytic outside the unit disc is denoted by $\mathcal{H}(\mathbb{E})$.

We define the Riesz-Herglotz kernel

$$D(t, z) = \frac{t + z}{t - z}.$$

We can associate a positive real function $\Omega(z)$ with the measure μ as follows

$$\Omega(z) = \langle D(t, z), 1 \rangle_\mu, \quad z \in \mathbb{D}.$$

This function is analytic in \mathbb{D} and the relation between Ω and μ is one-to-one. If $\Omega \in H_1(\mathbb{D})$, the Hardy-space of analytic, absolutely integrable functions, then we have

$$\Omega(z) = c_0 + 2 \sum_{k=1}^{\infty} c_k z^k, \quad (3)$$

which converges uniformly in \mathbb{D} .

By $\alpha = \{\alpha_1, \alpha_2, \dots\} \subset \mathbb{D}$, we denote a given sequence of points inside the unit disc. These points will be the poles of the orthogonal rational functions. We set $\alpha_0 = 0$ for notational reasons.

Now we can define the Blaschke factors ζ_k as

$$\zeta_k(z) = \frac{1 - \overline{\alpha_k}z}{z - \alpha_k}, \quad k \in \mathbb{N}, \quad (4)$$

and the Blaschke products B_k as

$$B_0(z) = 1, \quad B_k(z) = B_{k-1}(z)\zeta_k(z), \quad k = 1, 2, \dots \quad (5)$$

If we incorporate ζ_0 , we use the notation

$$\tilde{B}_{-1}(z) = 1, \quad \tilde{B}_k(z) = \zeta_0(z)B_k(z) = \frac{B_k(z)}{z}, \quad k \in \mathbb{N}. \quad (6)$$

The space of all rational functions with poles in $\{\alpha_1, \alpha_2, \dots, \alpha_n\}$ is denoted by

$$\mathcal{L}_n = \text{span}\{B_0(z), B_1(z), \dots, B_n(z)\}. \quad (7)$$

The space \mathcal{L}_n can also be defined as

$$\mathcal{L}_n = \left\{ \frac{p_n(z)}{\pi_n(z)} : p_n \in \Pi_n, \pi_n(z) = \prod_{k=1}^n (z - \alpha_k) \right\},$$

where Π_n denotes the space of polynomials of degree equal to or less than n . The space of all rational functions with poles in α is denoted by

$$\mathcal{L} = \bigcup_{k=0}^{\infty} \mathcal{L}_k.$$

Note that if all $\alpha_k = 0$, the space \mathcal{L}_n reduces to Π_{-n} , the space of all polynomials in z^{-1} of degree equal to or less than n .

We also introduce the notation

$$B_{n/k}(z) = B_n(z)/B_k(z), \quad n, k \in \mathbb{N}.$$

Taking the complex conjugate of a function on the unit circle is extended to the whole complex plane by the substar transform

$$f_*(z) = \overline{f(1/\bar{z})}, \quad z \in \mathbb{C}.$$

The superstar transform is defined for functions $f \in \mathcal{L}_n \setminus \mathcal{L}_{n-1}$ as

$$f^*(z) = B_n(z)f_*(z), \quad z \in \mathbb{C}.$$

Note that this operation depends on the degree n of the rational function. We do not incorporate this in the notation because it will be clear from the context. For polynomials of exact degree n the superstar transform is defined as above by replacing $B_n(z)$ by z^n .

2.3 Recurrence for the orthogonal rational functions

With the Gram-Schmidt procedure we can orthonormalize the basis $\{B_k\}_{k \in \mathbb{N}}$ to find a set of orthonormal rational functions $\phi_k \in \mathcal{L}_k$ with respect to the measure μ . This implies

$$\langle \phi_k, \phi_l \rangle_\mu = \delta_{kl} \quad \forall k, l \in \mathbb{N},$$

where δ_{kl} denotes the Kronecker delta.

The following recurrence relations for these orthonormal rational functions $\{\phi_k\}_{k=0}^\infty$ can be found in [17].

Theorem 2.3.1 *With the notation above the following recursion for the orthonormal rational functions $\phi_n \in \mathcal{L}_n$ is valid:*

$$\begin{bmatrix} \phi_n(z) \\ \phi_n^*(z) \end{bmatrix} = e_n \frac{z - \alpha_{n-1}}{z - \bar{\alpha}_n} \begin{bmatrix} 1 & \bar{L}_n \\ L_n & 1 \end{bmatrix} \begin{bmatrix} \zeta_{n-1}(z) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \phi_{n-1}(z) \\ \phi_{n-1}^*(z) \end{bmatrix},$$

where

$$L_n = -\frac{\left\langle \phi_k, \frac{1 - \bar{\alpha}_{n-1}z}{z - \alpha_n} \phi_{n-1} \right\rangle_\mu}{\left\langle \phi_k, \frac{z - \alpha_{n-1}}{z - \alpha_n} \phi_{n-1}^* \right\rangle_\mu}, \quad \forall k \leq n-1, \quad (8)$$

$$e_n = \left(\frac{1 - |\alpha_n|^2}{1 - |\alpha_{n-1}|^2} \frac{1}{1 - |L_n|^2} \right)^{1/2}. \quad (9)$$

The initial conditions are $\phi_0(z) \equiv 1/\sqrt{c_0} \equiv \phi_0^*(z)$.

We also mention here that the space \mathcal{L} is dense in $H^2(\mathbb{E})$, the Hardy space of all functions, which are square integrable on \mathbb{T} and analytic in \mathbb{E} , iff the poles do not approach the boundary \mathbb{T} too fast. This is reflected in the Blaschke condition (for a proof with the poles outside \mathbb{T} , see [3])

$$\sum_{k=0}^{\infty} (1 - |\alpha_k|^2) = \infty. \quad (10)$$

2.4 An algorithm to compute the reflection coefficients

In [17] we developed an algorithm if the sequence of moments c_k (2) is finite

$$c_{-N}, c_{1-N}, \dots, c_{N-1}, c_N.$$

We will use the following notation

$$\begin{aligned} C^N &= [c_0 \dots c_N]; \\ \alpha^n &= \{\alpha_0, \alpha_1, \dots, \alpha_n\}, \text{ the poles of the ORF}; \\ \mathbf{z} &= \text{the points where we want to evaluate the ORF}; \\ \Phi &= \text{the matrix whose } (i, j)\text{-th element is } \phi_{i-1}(\mathbf{z}(j)). \end{aligned}$$

We use the MATLAB notation. With $*$ and $/$, we denote matrix multiplication and division. The notations $.*$ and $./$ denote elementwise multiplication and division. The transpose of a matrix A is denoted as A^T . The left- and right-shifted vector of $\mathbf{x} = [x_1 \dots x_m]$ is denoted as $\mathbf{x}| = [x_2 \dots x_m \ 0]$ and $|\mathbf{x} = [0 \ x_1 \dots x_{m-1}]$, respectively. The algorithm to calculate the ORF that we gave in [17] is given next.

Algorithm 1

ORFcalcul \langle $IN: C^N, \alpha^n, \mathbf{z};$
 $OUT: \Phi \rangle$

1. $\Phi(1, :) = 1/\sqrt{c_0} = \Phi^*(1, :);$
 2. $\gamma^{Num} = C^N|;$
 $\gamma^{Den} = C^N;$
 3. for $k = 1 : n,$
 - (a) $\mathbf{a} = [1 \ \alpha_k \ \alpha_k^2 \ \dots \ \alpha_k^{N-1}]^T;$
 - (b) $L_k = -\overline{\gamma^{Num} * \mathbf{a} / \gamma^{Den} * \mathbf{a}};$
 - (c) $e_k = \sqrt{(1 - |\alpha_k|^2) / ((1 - |\alpha_{k-1}^2|) * (1 - |L_k|^2))};$
 - (d) $\Phi(k+1, :) = e_k * ((1 - \overline{\alpha_{k-1}} * \mathbf{z}). * \Phi(k, :) + \overline{L_k} * (\mathbf{z} - \alpha_{k-1}). * \Phi^*(k, :)) ./ (\mathbf{z} - \alpha_k);$
 - (e) $\Phi^*(k+1, :) = e_k * (L_k * (1 - \overline{\alpha_{k-1}} * \mathbf{z}). * \Phi(k, :) + (\mathbf{z} - \alpha_{k-1}). * \Phi^*(k, :)) ./ (\mathbf{z} - \alpha_k);$
 - (f) if $k < n,$
 - i. $\beta = \overline{L_k} * \gamma^{Den} | + \gamma^{Num} |;$
 - ii. for $l = N : -1 : 1, \beta(l) = \beta(l) + \alpha_k \beta(l+1);$ end for;
 - iii. $\gamma^{Den} = \gamma^{Den} + L_k * \gamma^{Num};$
 - iv. $\gamma^{Num} = \beta - \overline{\alpha_k} * |\beta|;$
- end for;

3 System identification using ORF

3.1 Summary

In the next paragraph some preliminaries and the problem setting are given. The third paragraph deals with the question 'Which measure μ should we take?'. We describe two different choices for the inner product $\langle \cdot, \cdot \rangle_\mu$: the continuous, leading to a time domain identification algorithm in paragraph four and the discrete one, leading to a frequency domain identification algorithm in paragraph five.

3.2 Preliminaries and problem setting

It is our goal to identify a system by means of its input and output data. Suppose the system we want to identify is given by

$$y(t) = G(z)u(t) + v(t), \quad t \in \mathbb{Z}, \quad (11)$$

where z denotes the shift operator ($zu(t) = u(t+1)$) and $v(t) = H(z)e(t)$, where H is a stable, rational and monic transfer function and e is a unit-variance, zero mean white noise process. This setting is standard [6].

We call G the transfer function of the system, which we suppose to be in $H^2(\mathbb{E})$ ¹. Furthermore, we suppose throughout the rest of the article that the input signal is strictly causal. This means $u(t) = 0$ for $t \leq 0$. We also assume that the Blaschke condition (10) is satisfied². This assumption implies that the transfer function G can be written as a unique series expansion

$$G(z) = \sum_{k=0}^{\infty} G_k \phi_k(z), \quad z \in \mathbb{E}, \quad (12)$$

where the non-tangential limit for $z \rightarrow \mathbb{T}$ exists a.e.

Note that if all poles $\alpha_k = 0$ and the measure μ is the Lebesgue measure, we are in the FIR case and the expansion (12) reduces to a Fourier series expansion where the G_k are nothing else but the Fourier coefficients or Markov parameters of G . Some convergence results for these general rational Fourier series can be found in [16].

The model structure, defined in (12), allows us to incorporate knowledge of the poles of the system in the ORF ϕ_k , which results in a much faster convergence than the classical methods, like e.g. FIR. This means that we achieve a considerable data reduction in the number of coefficients we need to compute. However we must compute the ORF ϕ_k , but this can be done very fast and accurate as we explained in section 2.

As mentioned before, we try to identify the system (11), by means of input and output data. We look for an approximation of the transfer function that is an approximation of a partial sum of the series expansion (12) of G

$$\hat{G}_n(z, \theta) = \sum_{k=0}^n \theta_k \phi_k(z), \quad (13)$$

where n is the degree of the approximation and the parameter $\theta = [\theta_0 \ \dots \ \theta_n]^T$ is varying over an appropriate set. We will estimate this parameter θ in a least squares setting.

Thus, we use a linear model structure defined by

$$\hat{y}(t, \theta) = \hat{G}_n(z, \theta)u(t). \quad (14)$$

Note that the approximation \hat{G}_n of the transfer function G is proper, but not strictly proper. A strictly proper transfer function can be modeled by redefining \mathcal{L}_n as the span of the Blaschke products \tilde{B}_k and some other small changes to the theory of section 2, but we are not going into detail here.

Suppose we are given some data $\{u(t), y(t)\}_{t=1}^N$, taken from experiments on the system (11). The one-step-ahead prediction error is given by

$$\varepsilon_n(t, \theta) = y(t) - \hat{y}(t, \theta) = y(t) - \sum_{k=0}^n \theta_k \phi_k(z)u(t). \quad (15)$$

In the next paragraph we will go into some more detail to find a good choice for the measure μ .

¹This assumption implies that we are only interested in identifying stable systems

²This assumption is reasonable, since we are interested in computing a finite degree approximation. We can assume all the other poles to be zero, which implies that the Blaschke condition is satisfied.

3.3 The measure μ

We have derived an expression for the one-step-ahead prediction error (15) in the previous paragraph. We would like this error to become small in a least squares sense. In this paragraph we will further analyse this error and come to a choice for the measure μ such that the computations become easy and fast.

First of all, we mention the case where we take the measure μ to be the Lebesgue measure. The ORF are analytically known (see [9])

$$\phi_k(z) = \frac{\sqrt{1 - |\alpha_k|^2} z}{z - \alpha_k} B_{k-1}(z), \quad k = 1, 2, \dots$$

The orthonormality is easy to check, but a nice state space based proof is given in [18]. We have already explored this case in [19].

If we take the z -transform³ of (15), we find (when we neglect the noise V)

$$\varepsilon_n(z, \theta) = Y(z) - \hat{Y}(z, \theta) = \left(G(z) - \hat{G}_n(z, \theta) \right) U(z), \quad (16)$$

where Y and U denote the z -transform of y and u respectively.

We want to determine the optimal parameter θ in a least squares sense. Thus we want to minimize

$$\|\varepsilon_n(z, \theta)\|^2 = \left\langle \left(G - \hat{G}_n(\cdot, \theta) \right) U, \left(G - \hat{G}_n(\cdot, \theta) \right) U \right\rangle_\lambda,$$

where λ is the normalized Lebesgue measure. If we choose

$$d\mu(z) = |U(z)|^2 d\lambda, \quad (17)$$

then the minimizer is easily found to be

$$\hat{G}_n(z, \theta) = G_n(z) = \sum_{k=0}^n G_k \phi_k(z).$$

This is quite interesting, because if we want to add one degree we only need to compute one coefficient or if we see that the last term does not give much better results, we can throw it away and take a new pole α_n . This can give rise to a powerfull interactive tool in system identification.

There will however be a problem if only a finite number of input and output data is available $\{u(t), y(t)\}_{t=1}^N$. When the input signal $u(t) = 0$ for $t > N$, then the output $y(t)$ need not be zero for $t > N$. This means that the z -transform $Y(z)$ cannot be computed exactly using only the data $\{u(t), y(t)\}_{t=1}^N$. This will give rise to two approaches: the *continuous* approach, where we will not be able to recursively update our model and the *discrete* approach, where we suppose that the output becomes zero after time $t = N$.

3.4 The continuous approach

In this paragraph we will develop a time-domain system identification algorithm that is based on the choice of a continuous inner product

$$\langle f, g \rangle_\mu = \int_{\mathbb{T}} f(z) g_*(z) d\mu(z), \quad f, g \in L^2(\mu), \quad (18)$$

³The z -transform of $\{x(t)\}_{t=1}^N$ is defined as $X(k) = \sum_{t=1}^N x(t) z^{1-k}$.

with the measure μ defined as in (17). With this choice, $U(z)$ is a polynomial of degree less than or equal to N and hence $|U(z)|^2$ is a trigonometric polynomial of the same degree so that the sequence of moments c_k is finite, which is what we have in paragraph 2.4. Thus we can calculate the orthogonal rational functions $\phi_n(z)$ recursively.

As identification model we find using (14)

$$\begin{aligned}\hat{y}(t, \theta) &= \hat{G}_n(z, \theta)u(t) = \sum_{k=0}^n \theta_k \phi_k(z)u(t) \\ &= \sum_{k=0}^n \theta_k x_k(t), \quad \text{with } x_k(t) = \phi_k(z)u(t).\end{aligned}$$

We are able to compute $x_k(t)$, but it is actually an infinite sequence. We will cut it off, which results in the loss of orthogonality. The least squares computations will however be numerically stable due to *nearly* orthogonality of the matrix Ψ (see (20)).

When we use the following notation

$$\begin{aligned}\mathbf{E} &= [\varepsilon_n(1, \theta) \ \varepsilon_n(2, \theta) \ \cdots \ \varepsilon_n(N, \theta)]^T, \\ \mathbf{Y} &= [y(1) \ y(2) \ \cdots \ y(N)]^T, \\ \theta &= [\theta_0 \ \theta_1 \ \cdots \ \theta_n]^T, \\ x_k(t) &= \phi_k(z)u(t), \\ \mathbf{x}_k &= [x_k(1) \ x_k(2) \ \cdots \ x_k(N)]^T, \\ \Psi &= [\mathbf{x}_0 \ \mathbf{x}_1 \ \cdots \ \mathbf{x}_n],\end{aligned}$$

then we find the formula

$$\mathbf{E} = \mathbf{Y} - \Psi\theta. \quad (19)$$

The identification problem is to determine θ in a least square sense, which can be done by solving the normal equations

$$\theta = (\Psi^H \Psi)^{-1} \Psi^H \mathbf{Y}, \quad (20)$$

where $(\cdot)^H$ denotes the Hermitian conjugate of a matrix.

The matrix Ψ is *nearly* orthogonal as mentioned before. This implies that the condition number of the systemmatrix $\Psi^H \Psi$ in (20) is small. So this problem can be accurately computed. The Matlab-file `ORFsysidC.m` can be found in the appendix.

3.5 The discrete approach

In this paragraph we will develop a frequency domain identification algorithm, based on the choice of a discrete inner product. Suppose the input signal has length N . We take N equally spaced points on the unit circle

$$t_k = \exp(i2\pi(k-1)/N), \quad k \in \{1, 2, \dots, N\}. \quad (21)$$

We define the inner product

$$\langle f, g \rangle_{|U|^2} = \sum_{k=1}^N |U(t_k)|^2 f(t_k) \overline{g(t_k)}, \quad f, g \in L^2(|U|^2), \quad (22)$$

where $U(z)$ denotes the z -transform of the input signal u . We denote $w_k = |U(t_k)|^2$. With this inner product the sequence of moments c_k is not finite, but cyclic. We cannot use Algorithm 1 of paragraph 2.4 anymore, but the reflection coefficients can easily be computed by expression (8). For $k = 0$, this gives

$$L_n = -\frac{\sum_{k=1}^N w_k \frac{t_k - \alpha_{n-1}}{1 - \overline{\alpha_n} t_k} \overline{\phi_{n-1}(t_k)}}{\sum_{k=1}^N w_k \frac{1 - \overline{\alpha_{n-1}} t_k}{1 - \overline{\alpha_n} t_k} \phi_{n-1}^*(t_k)}. \quad (23)$$

We could also choose any other $k \leq n - 1$, but these expressions need more calculations, which imply less numerical accuracy.

Suppose we know the z -transform of u and y in the points t_k : $\{U(t_k), Y(t_k)\}_{k=1}^N$. If we only have $\{u(t), y(t)\}_{t=1}^N$, we can compute $U(t_k)$ exactly, but only an approximation of $Y(t_k)$, since the output sequence is infinite as discussed above. This will imply that we do not identify the exact transfer function G , but one that is close to it.

We have

$$\begin{aligned} \hat{Y}(t_k, \theta) &= \hat{G}_n(t_k, \theta)U(t_k) = \sum_{l=0}^n \theta_l \phi_l(t_k)U(t_k) \\ &= \sum_{l=0}^n \theta_l X_l(t_k), \quad \text{with } X_l(t_k) = \phi_l(t_k)U(t_k). \end{aligned}$$

We introduce the following notation

$$\begin{aligned} \mathbf{E} &= [\varepsilon_n(t_1, \theta) \ \varepsilon_n(t_2, \theta) \ \cdots \ \varepsilon_n(t_N, \theta)]^T, \\ \mathbf{Y} &= [Y(t_1) \ Y(t_2) \ \cdots \ Y(t_N)]^T, \\ \theta &= [\theta_0 \ \theta_1 \ \cdots \ \theta_n]^T, \\ X_l(t_k) &= \phi_k(t_k)U(t_k), \\ \mathbf{X}_1 &= [X_l(t_1) \ X_l(t_2) \ \cdots \ X_l(t_N)]^T, \\ \Psi &= [\mathbf{X}_0 \ \mathbf{X}_1 \ \cdots \ \mathbf{X}_n]. \end{aligned}$$

Again we find the formula

$$\mathbf{E} = \mathbf{Y} - \Psi\theta. \quad (24)$$

The identification problem is to determine θ in a least square sense, which can be done by solving the normal equations

$$\theta = (\Psi^H \Psi)^{-1} \Psi^H \mathbf{Y}, \quad (25)$$

where $(\cdot)^H$ denotes the Hermitian conjugate of a matrix. Here the matrix Ψ is orthogonal. This implies that (25) becomes

$$\theta = \Psi^H \mathbf{Y}. \quad (26)$$

So $\theta_k = \langle G, \phi_k \rangle_\mu$, which gives us a powerfull interactive identification tool.

The reason for the choice of the points t_k (21) is the fact that the weights are then easy to compute with the discrete Fourier transform. We could choose other points on the unit circle \mathbb{T} , as long as the number of points is larger than the degree of the approximation we want. This can give a substantial data reduction.

For frequency domain identification where one uses ORF with respect to the Lebesgue measure and the poles are repeated cyclically, see [4].

We wrote 2 Matlab-files `ORFsysidD1.m`, with input $\{u(t), y(t)\}_{t=1}^N$ and `ORFsysidD2.m`, with input $\{U(t_k), Y(t_k)\}_{k=1}^N$. They can be found in the appendix.

Table 1: Relative errors

$\ y - \hat{y}\ /\ y\ $	ORFsysidC	ORFsysidD1	Lebesgue
$\alpha 1$.2029	.2154	.2033
$\alpha 2$.0655	.1072	.0655
$\alpha 3$.1360	.1601	.1360
$\alpha 4$.1125	.1393	.1127

Table 2: CPU-time

cpu-time	ORFsysidC	ORFsysidD1	Lebesgue
$\alpha 1$	2.9169	0.4227	10.183
$\alpha 2$	2.2641	0.4720	7.8914
$\alpha 3$	2.2553	0.5990	7.9747
$\alpha 4$	2.2666	0.3831	7.9484

4 A numerical example

We consider the following transfer function

$$\begin{aligned}
 G(z) &= \frac{z^5 + 4z^4 + 27z^3 - 38z^2 + 170z}{z^5 - .7z^4 - 1.08z^3 + .59z^2 + .72z - .36} \\
 &= \frac{z(z - 1 + 2i)(z - 1 - 2i)(z + 3 + 5i)(z + 3 - 5i)}{(z - .9 + .3i)(z - .9 - .3i)(z + .8 + .4i)(z + .8 - .4i)(z - .5)}.
 \end{aligned}$$

This system has poles: $.9 \pm .3i$, $-.8 \pm .4i$ and $.5$. We consider an input u of length $N = 1000$ uniformly spread in $[0..1)$ and no output error.

We use the following sets of poles:

$$\begin{aligned}
 \alpha_1 &= [.95 \pm .1i \quad -.7 \pm .2i], \text{ repeated 5 times;} \\
 \alpha_2 &= [.9 \quad -.8 \quad .4 \quad .6], \text{ repeated 5 times;} \\
 \alpha_3 &= \textit{zeros}(1, 20), \text{ the polynomial case;} \\
 \alpha_4 &= -.95 : .1 : .95;
 \end{aligned}$$

In Table 1 the relative errors are noted for the 2 different approaches. Also the case where the Lebesgue measure is used (see [19]) is denoted in Table 1. Table 2 gives the cpu-time of the different approaches and Table 3 the number of floating point operations.

The relative error for the continuous case and the Lebesgue-case is of course the same, since we are looking for the best solution in the same space \mathcal{L}_{20} . Due to the fact that we are not approximating the exact transfer function in the discrete case, the relative error is a little worse. But the cpu-time is in this case much better, due to the fact that we do not have to solve a system due to the orthogonality of the systemmatrix.

The number of floating point operations is not so representative, because in the discrete case every operation deals with complex numbers and in the Lebesgue case we do not have to calculate the reflection coefficients.

Table 3: Floating point operations

flops	ORFsysidC	ORFsysidD1	Lebesgue
$\alpha 1$	12509946	11589269	11365643
$\alpha 2$	5866561	11589113	2967771
$\alpha 3$	5866561	11400113	2969317
$\alpha 4$	5866561	11589113	2968700

5 Conclusions

We have given two different identification algorithms. The time domain identification algorithm gives a better fit to the original system, where the frequency domain identification algorithm is better in execution time.

References

- [1] H. Akçay and B. Ninness. Rational basis functions for robust identification from frequency and time domain measurements. *Automatica*, 34(9):1101–1117, 1998.
- [2] H. Akçay and B. Ninness. Orthonormal basis functions for continuous-time systems and L_p convergence. *Math. Control Signals Systems*, 12:295–305, 1999.
- [3] A. Bultheel, P. González-Vera, E. Hendriksen, and O. Njåstad. *Orthogonal rational functions*, volume 5 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, 1999.
- [4] D. K. de Vries and P. van den Hof. Frequency domain identification with generalized orthonormal basis functions. *IEEE Trans. Automat. Control*, AC-43(5):656–669, 1998.
- [5] P.S.C. Heuberger, P.M.J. Van den Hof, and O. Bosgra. A generalized orthogonal basis for linear dynamical systems. *IEEE Trans. Automat. Control*, 40:451–465, 1995.
- [6] L. Ljung. *System identification: Theory for the user*. Information and System Sciences Series. Prentice Hall, 2nd edition, 1999. First edition 1987.
- [7] B. Ninness. Frequency domain estimation using orthonormal bases. In *Proceedings of the 13th IFAC World Congress, San Francisco 1996, Volume 8*, pages 381–386, 1996.
- [8] B. Ninness and J.C. Gómez. Asymptotic analysis of MIMO system estimates by the use of orthonormal bases. In *Proceedings of the 13th IFAC World Congress, San Francisco 1996, Volume 8*, pages 363–368, 1996.
- [9] B. Ninness and F. Gustafsson. A unified construction of orthogonal bases for system identification. *IEEE Trans. Automat. Control*, 42:515–522, 1997.
- [10] B. Ninness, H. Hjalmarsson, and F. Gustafsson. The fundamental role of general orthogonal bases in system identification. *IEEE Trans. Automat. Control*, 44(7):1384–1407, 1999.

- [11] B. Ninness, H. Hjalmarsson, and F. Gustafsson. Generalised Fourier and Toeplitz results for rational orthonormal bases. *SIAM J. Control Optim.*, 37:429–460, 1999. <http://murray.newcastle.edu.au/users/staff/brett/ee9740.ps.gz>.
- [12] Z. Szabó, J. Bokor, and F. Schipp. Identification of rational approximate models in H_∞ using generalized orthogonal basis. Manuscript, 1998.
- [13] G. Szegő. *Orthogonal polynomials*, volume 33 of *Amer. Math. Soc. Colloq. Publ.* Amer. Math. Soc., Providence, Rhode Island, 4th edition, 1975. First edition 1939.
- [14] P.M.J. Van den Hof, P.S.C. Heuberger, and J. Bokor. System identification and with generalized orthogonal basis functions. *Automatica*, 31:1821–1834, 1995.
- [15] P. Van gucht and A. Bultheel. State-space realization and orthogonal rational functions. Technical Report TW292, Department of Computer Science, K.U. Leuven, September 1999.
- [16] P. Van gucht and A. Bultheel. Bernstein equiconvergence and Fejér type theorems for general rational Fourier series. *J. Comput. Appl. Math.*, 2000. Accepted.
- [17] P. Van gucht and A. Bultheel. Computing orthogonal rational functions analytic outside the unit disc. Technical Report TW312, Department of Computer Science, K.U. Leuven, August 2000.
- [18] P. Van gucht and A. Bultheel. A relation between orthogonal rational functions on the unit circle and the interval $[-1, 1]$. Technical Report TW299, Department of Computer Science, K.U. Leuven, January 2000.
- [19] P. Van gucht and A. Bultheel. SISO system identification using orthogonal rational functions. unpublished, 2000.
- [20] B. Wahlberg. System identification using Laguerre models. *IEEE Trans. Automat. Control*, AC-36(5):551–562, 1991.
- [21] B. Wahlberg. System identification using Kautz models. *IEEE Trans. Automat. Control*, AC-39(6):1276–1282, 1994.
- [22] J.L. Walsh. *Interpolation and approximation*, volume 20 of *Amer. Math. Soc. Colloq. Publ.* Amer. Math. Soc., Providence, Rhode Island, 3rd edition, 1960. First edition 1935.

A The continuous case

Here we give the MATLAB file, which does the identification of a system as in paragraph 3.4. The file `ORFsysidC.m` computes the parameter θ and also returns the reflection coefficients of the orthogonal rational functions ϕ_k and the relative identification error.

This m-file is freely available at

<http://www.cs.kuleuven.ac.be/~nalag/research/public-software.html>

```

function [theta,L,err,coef] = ORFsysidC(u,y,alpha)

% ORFSYSIDC =
%   SISO System identification using orthogonal rational
%   functions phi_k, with user defined poles and
%   with respect to a weight defined by the input.
%   The identification is based on a continuous inner product.
%
% use: [theta,L,err,coef] = ORFsysidC(u,y,alpha)
% output:
% theta = coefficients in expansion: G_hat = sum_k theta_k phi_k
% L      = reflection coefficients of the ORF
% err    = relative error of the approximation
% coef   = Fourier coefficients of the weight
% input:
% u      = input signal (column vector !)
% y      = output signal (column vector !)
% alpha  = the poles of the ORF
%
% See also: ORFsysidD1, ORFsysidD2, interactiveORFsysid, ORFcalcul

% Patrick Van gucht and Adhemar Bultheel, september, 11, 2000.

%initializations
if (size(u,1)<size(u,2)), % row vector --> column vector
    u=transpose(u);
end
if (size(y,1)<size(y,2)), % row vector --> column vector
    y=transpose(y);
end

N = min(length(u),length(y)); % Length of the signal
n = length(alpha);
u = u(1:N);
y = y(1:N);

hulp = conv(conj(u),flipud(u));
coef = hulp(N:2*N-1); % Fourier coefficients of the weight
omega0 = coef(1)

x(:,1)=u/sqrt(omega0);
xstar(:,1)=u/sqrt(omega0);
L = zeros(n,1);

alpha = [0 alpha];
k = 2;
gamma_num = [coef(2:N); 0];

```

```

gamma_den = coef;
beta = zeros(N-1,1);

for k=1:n,
    z = alpha(k+1).^[0:N-1];
    L(k) = -conj((z*gamma_num)/(z*gamma_den)); % reflectioncoefficient
    hulp = sqrt((1-abs(alpha(k+1))^2)/(1-abs(alpha(k))^2)/(1-abs(L(k))^2));
    v1 = filter([-alpha(k)' 1],[1 -alpha(k+1)],x(:,k));
    v2 = filter([1 -alpha(k)],[1 -alpha(k+1)],xstar(:,k));
    x(:,k+1) = hulp*(v1 + L(k)'*v2);
    xstar(:,k+1) = hulp*(L(k)*v1+v2);
    if (k<n),
        beta(N-1) = L(k)'*gamma_den(N) + gamma_num(N);
        for l=N-2:-1:1,
            beta(l) = L(k)'*gamma_den(l+1) + gamma_num(l+1) + alpha(k+1)*beta(l+1);
        end
        gamma_den = gamma_den + L(k)*gamma_num;
        gamma_num(1) = beta(1);
        gamma_num(2:N-1) = beta(2:N-1)-alpha(k+1)'*beta(1:N-2);
        gamma_num(N) = -alpha(k+1)'*beta(N-1);
    end
end

% IDENTIFICATION
theta = x\y;
yhat = x*theta;
err = norm(y-yhat)/norm(y);

```

B The discrete case

Here we give the 2 m-files ORFsysidD1.m and ORFsysidD2.m, described in paragraph 3.5. In the first one we must give the input and output data $\{u(t), y(t)\}_{t=1}^N$. The output y is presumed to be zero if $t > N$. For the second, the z -transform of u and y need to be given in the points $t_k = \exp(i2\pi(k-1)/N)$.

These files are not interactive, since you have to give the poles α^n as input. An interactive file `interactiveORFsysid.m` is freely available at

<http://www.cs.kuleuven.ac.be/~nalag/research/public-software.html>

together with these two m-files.

```

function [theta,L,err,moments]=ORFsysidD1(u,y,alpha)

% ORFSYSIDD1 =
%   SISO System identification using orthogonal rational
%   functions phi_k, with user defined poles and
%   with respect to a weight defined by the input.

```

```

% The identification is based on a discrete inner product.
%
% use: [theta,L,err,moments] = ORFsysidD1(u,y,alpha)
% output:
% theta = coefficients in expansion: G_hat = sum_k theta_k phi_k
% L      = reflection coefficients of the ORF
% err    = relative error of the approximation
% moments= the moments
% input:
% u      = input signal (column vector !)
% y      = output signal (column vector !)
% alpha = the poles of the ORF
%
% See also: ORFsysidC, ORFsysidD2, interactiveORFsysid, ORFcalcul

% Patrick Van gucht and Adhemar Bultheel, september, 11, 2000.

%initialization
if (size(u,1)<size(u,2)), % row vector --> column vector
    u=transpose(u);
end
if (size(y,1)<size(y,2)), % row vector --> column vector
    y=transpose(y);
end

N = length(u); % Length of the signal
n = length(alpha); % number of ORF
u = u(1:N);
y = y(1:N);

moments = conv(conj(u),flipud(u)); %to compute the weights
moments = moments(N:2*N-1);
I=sqrt(-1);
t=exp(I*2*pi/N*transpose([0:N-1]));
hulp = fft(moments);
weight = transpose(hulp + conj(hulp) - moments(1));
W=sum(weight)

U = fft(u);
Y = fft(y);

phi(:,1)=ones(N,1)/sqrt(W);
phistar(:,1)=ones(N,1)/sqrt(W);
x(:,1)=phi(:,1).*U;
xstar(:,1)=phistar(:,1).*U;

L = zeros(n,1);

```

```

alpha = [0 alpha];
for k=2:n+1,
    hulp1 = (t-alpha(k-1))./(1-alpha(k)')*t).*(abs(phi(:,k-1)).^2);
    hulp2 = (1-alpha(k-1)')*t)./(1-alpha(k)')*t).*phi(:,k-1).*conj(phistar(:,k-1));
    L(k-1) = -(weight*hulp1)/(weight*hulp2); % reflectioncoefficient
    e_k = sqrt((1-abs(alpha(k))^2)/((1-abs(alpha(k-1))^2)*(1-abs(L(k-1))^2)))
        ./(t-alpha(k));
    hulp = (1-alpha(k-1)')*t).*phi(:,k-1);
    hulpstar = (t-alpha(k-1)).*phistar(:,k-1);
    phi(:,k) = e_k.*(hulp + L(k-1)')*hulpstar);
    phistar(:,k) = e_k.*(L(k-1)*hulp + hulpstar);
    x(:,k) = phi(:,k).*U;
    xstar(:,k) = phistar(:,k).*U;
end

```

```

% IDENTIFICATION
theta = x'*Y;
Yhat = x*theta;
err = norm(Y-Yhat)/norm(Y);

```

When the z -transform of input and output is known, the following m-file can be used.

```

function [theta,L,err]=ORFsysidD2(U,Y,alpha)

% ORFSYSIDD2 =
%   SISO System identification using orthogonal rational
%   functions phi_k, with user defined poles and
%   with respect to a weight defined by the input.
%   The identification is based on a discrete inner product.
%
% use: [theta,L,err,moments] = ORFsysidD2(U,Y,alpha)
% output:
%   theta = coefficients in expansion: G_hat = sum_k theta_k phi_k
%   L      = reflection coefficients of the ORF
%   err    = relative error of the approximation
% input:
%   U/Y    = z-transform of the in/out-put signal in the points
%           exp(I*2*pi*(k-1)/length(U))
%   alpha  = the poles of the ORF
%
% See also: ORFsysidC, ORFsysidD1, ORFcalcul

% Patrick Van gucht and Adhemar Bultheel, september, 11, 2000.

%initialization
if (size(U,1)<size(U,2)), % row vector --> column vector
    U=transpose(U);

```

```

end
if (size(Y,1)<size(Y,2)), % row vector --> column vector
    Y=transpose(Y);
end

N = length(U); % Length of the signal
n = length(alpha); % number of ORF
U = U(1:N);
Y = Y(1:N);

I=sqrt(-1);
t=exp(I*2*pi/N*transpose([0:N-1]));
weight = abs(transpose(U)).^2;
W=sum(weight)

phi(:,1)=ones(N,1)/sqrt(W);
phistar(:,1)=ones(N,1)/sqrt(W);
x(:,1)=phi(:,1).*U;
xstar(:,1)=phistar(:,1).*U;

L = zeros(n,1);
alpha = [0 alpha];
for k=2:n+1,
    hulp1 = (t-alpha(k-1))./(1-alpha(k)'*t).*(abs(phi(:,k-1)).^2);
    hulp2 = (1-alpha(k-1)'*t)./(1-alpha(k)'*t).*phi(:,k-1).*conj(phistar(:,k-1));
    L(k-1) = -(weight*hulp1)/(weight*hulp2); % reflectioncoefficient
    e_k = sqrt((1-abs(alpha(k))^2)/((1-abs(alpha(k-1))^2)*(1-abs(L(k-1))^2)))
        ./(t-alpha(k));
    hulp = (1-alpha(k-1)'*t).*phi(:,k-1);
    hulpstar = (t-alpha(k-1)).*phistar(:,k-1);
    phi(:,k) = e_k.*(hulp + L(k-1)'*hulpstar);
    phistar(:,k) = e_k.*(L(k-1)*hulp + hulpstar);
    x(:,k) = phi(:,k).*U;
    xstar(:,k) = phistar(:,k).*U;
end

% IDENTIFICATION
theta = x'*Y;
Yhat = x*theta;
err = norm(Y-Yhat)/norm(Y)

```

C Recover the approximation \hat{G}_n

When we have computed the identification parameter θ , the reflection coefficients L and the moments C^N , we can write the approximation \hat{G}_n as a quotient of two polynomials of degree n

$$\hat{G}_n(z) = \frac{p_n(z)}{\pi_n(z)}.$$

We wrote the m-file `Ghat.m`, which returns the coefficients of the numerator polynomial p_n and denominator polynomial π_n . Note that the input `c0` is different for the continuous or the discrete case. If we look at the output from `ORFsysidC.m` and `ORFsysidD1.m`, we find for the continuous case that `c0 = coef(1)`, while in the discrete case `c0 = sum(fft(moments) + conj(fft(moments)) - moments(1))`.

```
function [num,den] = Ghat(theta,L,alpha,c0),

% GHAT = calculate the approximation of the
%         transfer function as a linear combination
%         of orthogonal rational functions (ORF).
%
% USE: [num,den] = Ghat(theta,L,alpha,c0);
% output:
% num = coefficients of numerator of Ghat
% den = coefficients of denominator of Ghat
% input:
% theta = identification coefficients
% L      = reflection coefficients of the ORF
% alpha = the poles of the ORF
% c0     = the zero-th moment
%
% See also: ORFsysidC, ORFsysidD1, ORFsysidD2, ORFcalcul, interactiveORFsysid
%

% Patrick Van gucht, september 18, 2000.

% INITIALIZATION
n = length(alpha);

numphi = 1/sqrt(c0);
numphistar = numphi;

num = theta(1)*numphi;
den = 1;

alpha = [0 alpha];

% RECURRENCE
for k=1:n,
```

```
e_k = sqrt((1-abs(alpha(k+1))^2)/((1-abs(alpha(k))^2)*(1-abs(L(k))^2)));
helpnumphi = [0 numphi] - alpha(k)'*[numphi 0];
helpnumphistar = [numphistar 0] - alpha(k)*[0 numphistar];
numphi = e_k*(helpnumphi + L(k)'*helpnumphistar);
numphistar = e_k*(L(k)*helpnumphi + helpnumphistar);
num = [num 0] - alpha(k+1)*[0 num] + theta(k+1)*numphi;
den = [den 0] - alpha(k+1)*[0 den];
end
```