Metrol. Meas. Syst., Vol. XVIII (2011), No. 4, pp. 505-528.

METROLOGY AND MEASUREMENT SYSTEMS
Index 330930, ISSN 0860-8229
www.metrology.pg.gda.pl

FREQUENCY AND DAMPING ESTIMATION METHODS – AN OVERVIEW

Tomasz Piotr Zieliński$^{1)}$, Krzysztof Duda$^{2)}$

1) AGH University of Science and Technology, Department of Telecommunications, Al. Mickiewicza 30, 30-059 Kraków, Poland (✉ tzielin@agh.edu.pl)
2) AGH University of Science and Technology, Department of Measurement and Instrumentation, Al. Mickiewicza 30, 30-059 Kraków, Poland (kduda@agh.edu.pl)

Abstract

This overview paper presents and compares different methods traditionally used for estimating damped sinusoid parameters. Firstly, direct nonlinear least squares fitting the signal model in the time and frequency domains are described. Next, possible applications of the Hilbert transform for signal *demodulation* are presented. Then, a wide range of autoregressive modelling methods, valid for damped sinusoids, are discussed, in which frequency and damping are estimated from calculated signal linear self-prediction coefficients. These methods aim at solving, directly or using least squares, a matrix linear equation in which signal or its autocorrelation function samples are used. The Prony, Steiglitz-McBride, Kumaresan-Tufts, Total Least Squares, Matrix Pencil, Yule-Walker and Pisarenko methods are taken into account. Finally, the interpolated discrete Fourier transform is presented with examples of Bertocco, Yoshida, and Agrež algorithms. The Matlab codes of all the discussed methods are given. The second part of the paper presents simulation results, compared with the Cramér-Rao lower bound and commented. All tested methods are compared with respect to their accuracy (systematic errors), noise robustness, required signal length, and computational complexity.

Keywords: damped sinusoids, frequency estimation, damping estimation, linear prediction, subspace methods, interpolated DFT.

1. Introduction

The need for frequency and damping measurement appears in many fields [1]: electrical (e.g. power quality measurement [2-3]), mechanical, electromechanical, optical, biological, human, economical and social, geophysical, astrophysical, chemical. The literature on this subject is extensive and a significant amount of time is required to get a complete and consistent view in this field based on numerous different sources. There are many specialized computational environments with implemented standard estimation methods. Matlab is a prime example of such a sophisticated program. However, using toolbox functions for frequency and damping estimation is not always straightforward.

This paper aims to provide a 'reader's digest' of joint frequency and damping estimation methods. The paper is constructed as a practical overview and tutorial, and should be useful in selecting a suitable method for the case investigated by the reader. For this reason Matlab implementations of all the described and tested estimation methods are given in [4]. Additionally, the most efficient methods are presented in Appendix B and some basics of Matlab calculations are given in the text.

The paper is structured as follows. In section 2, the measurement model of interest is formulated. In section 3, direct signal model fitting methods (nonlinear least squares minimization) in time and frequency are introduced. In sections 4 to 6, the application of the Hilbert

Article history: received on Aug. 29, 2011; received in revised form on Sept. 11, 2011; accepted on Nov. 14, 2011; available online on Dec. 12, 2011.

transform, linear prediction (auto-regressive) parametric modelling and the discrete Fourier transform (DFT) for frequency and damping estimation are presented respectively. In section 7, the Cramér-Rao lower bound (CRLB) for statistically-efficient estimation of frequency and damping is given. The obtained simulation results of the tested methods are given in section 8. The paper ends with final conclusions, references and two appendices.

2. Signal model

The assumed signal model is

$$x_n = Ae^{-Dn}\sin\left(\Omega_x n+\varphi\right) = Ae^{-Dn}\cos\left(\Omega_x n+\varphi\right), \quad n=0,1,2,...,N-1, \tag{1}$$

where: $A > 0$ - signal amplitude, $0 < \Omega_x$-$<\pi$, - signal angular frequency in radians ($\Omega_x = 2\pi f_x/f_s$ and $\Omega_x = 2\pi$ rad corresponds to half of the sampling frequency f_s in hertz), $-\pi < \varphi \le \pi$ - phase angle in radians, n - index of the sample, N - the number of samples, $D \ge 0$ – damping of the digital signal.

We assume that the signal is embedded in the possible measurement disturbance ε_n (e.g. noise, drift, interference, etc.), i.e.

$$y_n = x_n + \varepsilon_n, \quad n=0,1,2,...,N-1. \tag{2}$$

The remaining part is devoted to estimating Ω_x, D, A and φ based on the measured signal y_n. We will concentrate only on the damped signal, as an undamped signal is a special case with D=0. When Ω_x and D are known, finding values of A and φ is straightforward (see Appendix A), therefore in most methods we only find values of Ω_x and D. In turn, in parametric modelling methods Ω_x and D can be calculated from coefficients of the linear self-prediction model, therefore further discussion is terminated once those coefficient are calculated.

3. Estimation based on direct model fitting

3.1. Time domain

Minimization of cost functions defined as [5]:

$$C_T(\Omega_x, A, \varphi, DC) = \sum_{n=0}^{N-1}(y_n - A\cos(\Omega_x n+\varphi) - DC)^2 , \tag{3}$$

$$C_T(\Omega_x, D, A, \varphi, DC) = \sum_{n=0}^{N-1}(y_n - A\cos(\Omega_x n+\varphi)e^{-Dn} - DC)^2 , \tag{4}$$

represents a straightforward approach to estimating parameters of the (un)damped sinusoidal signals given by (1); DC stands for a constant value that may be present in the measurement signal. Important drawbacks of this approach are as follows:

1) The above cost functions are difficult to minimize as they have many local minima, and a good starting point (close enough to the correct minimum) has to be selected.
2) It is difficult to choose a suitable signal model, as the measurement signal could be disturbed by different signals (e.g. drift, other sinusoids, etc.).
3) The method has a high computational complexity.

A practical example of the time domain optimization-based approach represents an OMI algorithm [6], reported to be very successful in mechanical spectroscopy for estimating resonant frequency and logarithmic decrement.

3.2. Frequency domain

The cost function for estimating signal parameters can also be defined in the domain of DFT coefficients as [5]:

$$C_F(\Omega_x, A, \varphi) = \sum_k [V_k - V(e^{j\Omega_k})]^2 , \tag{5}$$

$$C_F(\Omega_x, D, A, \varphi) = \sum_k [V_k - \overline{V}(e^{j\overline{\Omega}_k})]^2 , \tag{6}$$

where V_k denotes the DFT spectrum of the measured signal y_n analyzed with the window w_n (i.e. $v_n = y_n w_n$), $V(e^{j\Omega_k})$ is the theoretical spectrum of the windowed undamped sinusoidal signal ($D = 0$), and $\overline{V}(e^{j\overline{\Omega}_k})$ - of the windowed damped sinusoidal signal ($D > 0$). These spectra are given by:

$$V(e^{j\Omega}) = \frac{A_0}{2} e^{j\varphi} W(e^{j(\Omega-\Omega_x)}) + \frac{A_0}{2} e^{-j\varphi} W(e^{j(\Omega+\Omega_x)}) , \tag{7}$$

$$\overline{V}(e^{j\overline{\Omega}}) = \frac{A_0}{2} e^{j\varphi} \overline{W}(e^{j(\overline{\Omega}-\Omega_x)}) + \frac{A_0}{2} e^{-j\varphi} \overline{W}(e^{j(\overline{\Omega}+\Omega_x)}) , \tag{8}$$

where $W(e^{j\Omega})$ is a spectrum of the window w_n, $\overline{W}(e^{j\overline{\Omega}})$ is a spectrum of the damped window $\overline{w}_n = w_n e^{-Dn}$, introduced in [7], and $\overline{\Omega} = \Omega - jD$. For example, spectra of the rectangular and Hanning windows are as follows:

$$W_R(e^{j\Omega}) = e^{-j\Omega(N-1)/2} \frac{\sin(\Omega N/2)}{\sin(\Omega/2)} , \tag{9}$$

$$W_H(e^{j\Omega}) = -0.25 W_R(e^{j(\Omega-\Omega_1)}) + 0.5 W_R(e^{j\Omega}) - 0.25 W_R(e^{j(\Omega+\Omega_1)}), \quad \Omega_1 = 2\pi/N , \tag{10}$$

while for damped windows one has:

$$\overline{W}_R(e^{j\overline{\Omega}}) = e^{-j\overline{\Omega}(N-1)/2} \frac{\sin(\overline{\Omega} N/2)}{\sin(\overline{\Omega}/2)} , \tag{11}$$

$$\overline{W}_H(e^{j\overline{\Omega}}) = -0.25 \overline{W}_R(e^{j(\overline{\Omega}-\Omega_1)}) + 0.5 \overline{W}_R(e^{j\overline{\Omega}}) - 0.25 \overline{W}_R(e^{j(\overline{\Omega}+\Omega_1)}), \quad \Omega_1 = 2\pi/N . \tag{12}$$

For a rectangular window and $k = 0, 1, 2, ..., N/2$, minimization of (5-6) (i.e. fitting the signal spectrum model) is equivalent to minimizing (3-4) [5]. In (5-6) only a few DFT bins with the highest magnitude can be used, e.g. 3 out of 1024. In such a case we gain robustness against disturbances not included in the signal model at the cost of higher noise sensitivity.

The cost function (5) can be simplified by excluding the signal amplitude A [8] or amplitude A and phase φ [9]. In the last case the function depends only on frequency Ω_x.

Matlab implementation of the frequency domain optimization (6), based on [9], is given in the program **`fOptyDFT()`** [4]. The Rife-Vincent class I windows of order 0 to 6 [7-11] and Hamming and Blackman windows are implemented in the function **`window_cos()`**. In the optimization, three DFT bins with the highest magnitudes are used. The starting point is computed by the interpolated DFT algorithm described in [7].

4. Discrete Hilbert transform methods

Applying the Hilbert transform (HT) [12-14] to estimation of frequency and damping has been reported in many publications; however, the HT was mainly used for measuring damping [15-19]. Its usage in estimating frequency is discussed in [19]. The HT shifts the signal in phase by $-\pi/2$ rad, e.g. $\cos(\Omega_x n)$ becomes $\sin(\Omega_x n)$. An analytic signal (AS) is defined as a complex signal with the input samples in its real part and their HT in the imaginary part, e.g. for input $\cos(\Omega_x n)$ we have $\exp(j\Omega_x n) = \cos(\Omega_x n)+j\sin(\Omega_x n)$. For a discrete time signal of the form (1), its AS version is

$$x_n^{(a)} = Ae^{-Dn}e^{j\Omega_x n+\varphi} \tag{13}$$

and damping and frequency are given by (approximation of derivative):

$$D_n = -\operatorname{diff}(\ln(|x_n^{(a)}|)), \quad D = \operatorname{mean}(D_n), \tag{14}$$

$$\Omega_n = \operatorname{diff}(\angle x_n^{(a)}), \quad \Omega_x = \operatorname{mean}(\Omega_n), \tag{15}$$

where "diff" is the difference between two consecutive values as implemented in the Matlab function **`diff`** and |.|, $\angle$ denote the magnitude and angle of a complex number respectively. In computer implementation, an analytic signal can be calculated by convolution or modification of DFT coefficients [13, 20]. When Ω_x and D are known, we find values of A and φ using the method described in Appendix A.

Matlab implementation of the AS (or Hilbert) estimation method of frequency and damping is given in the program **`fHilbert()`** [4]. The calculated instantaneous frequency and damping have strong oscillations at the beginning and the end of the observation interval. Therefore 1/4 of samples is discarded at the beginning and the end of the computed data. In the last program line variable **`temp`** is used for damping estimation via line fitting, which is an option for computing the mean value. A summary of Matlab calculations is presented below.

```
N = length(x); ind = round( N/2-N/4 : N/2+N/4 ); win = hanning(N)';
Xh = hilbert(win.*x)./win;
Om = diff( unwrap( angle(Xh) ) );  Om = mean(Om(ind));
D = -diff( log( abs(Xh)) );        D  = mean(D(ind));
```

5. Parametric modeling methods – solving linear equations

5.1. Initial problem setup

The signal x_n (1) is an impulse response h_n of the following linear digital filter (u – input, v – output):

$$v_n = b_1 u_{n-1} - a_1 v_{n-1} - a_2 v_{n-2}, \tag{16a}$$

$$b_1 = Ae^{-D}\sin(\Omega_x), \ a_1 = -2e^{-D}\cos(\Omega_x), \ a_2 = e^{-2D}, \tag{16b}$$

having a transfer function $H(z)$ with a complex pole z_1 (zero of denominator) as follows ($a_0 = 1$):

$$H(z) = \frac{b_1 z^{-1}}{a_0 + a_1 z^{-1} + a_2 z^{-2}} = \frac{b_1 z^{-1}}{(1 - z_1 z^{-1})(1 - z_1^* z^{-1})}, \ z_1 = e^{p_1} = e^{-D}e^{j\Omega_x}. \tag{17}$$

Therefore the following linear self-prediction (auto regression) is valid for x_n:

$$x_n = -a_1 x_{n-1} - a_2 x_{n-2}, \quad x_{-1} = 0, \quad x_{-2} = -Ae^{-D}\sin(\Omega_x). \tag{18}$$

In order to estimate frequency Ω_x and damping D, one should find such values of coefficients a_1 and a_2 for given N noisy measurements $\{y_0, y_1, \ldots, y_{N-1}\}$, where $y_n = x_n + \varepsilon_n$, that the linear prediction model $y_n = -a_1 y_{n-1} - a_2 y_{n-2}$ fits best the measurement data y_n.

Many solutions for this optimization task are briefly summarized below. In general, the typical calculation path for algorithms covered in this section is as follows:

1) calculate coefficients a_1, a_2 of the signal self-prediction model ($a_0 = 1$),
2) find complex roots (zeros) $z_1 = e^{p1} = e^{-D} e^{j\Omega x}$, $z_2 = z_1^*$ of the polynomial $a_0 + a_1 z^{-1} + a_2 z^{-2}$ (or $a_0 z^2 + a_1 z + a_2$) describing the denominator of the transfer function of the second order digital system (17); in Matlab function **`roots`**([a_0, a_1, a_2]);
3) calculate Ω_x and D of the discrete-time signal:

$$\Omega_x = \angle(z_1), \quad D = -\ln(|z_1|), \tag{19}$$

$$\Omega_x = \left|\mathrm{Im}(\ln(z_1))\right|, \quad D = -\mathrm{Re}(\ln(z_1)), \tag{20}$$

4) estimate signal amplitude and phase (see appendix).

Having the transfer function $H(z)$ one can calculate the frequency response setting $z = \exp(j\Omega)$, $0 \le \Omega < 2\pi$, and find its greatest magnitude corresponding to the angular frequency Ω_x of the analyzed signal. However, estimation of damping is not possible using just the magnitude of the frequency response.

5.2. Classical solution to the linear prediction problem using signal samples (covariance methods)

There are several methods used in digital filter design, e.g. the Pade approximation procedure and the Prony least squares autoregressive model fitting [21] for finding coefficients $\{b_k, a_k\}$ of the digital filter difference equation (and thus the transfer function $H(z)$) for the desired/given filter impulse response h_n. In our case $h_n = x_n$ is a damped sinusoid, and the calculated coefficients $\{a_k\}$ are used to estimate frequency and damping, as stated above in (19)(20). Therefore, using the input-output relationship (16) leads to the well-known Prony digital filter design method [21].

It follows that the linear equation to be solved for a_1 and a_2 is:

$$\begin{bmatrix} y_0 & y_1 \\ y_1 & y_2 \\ \vdots & \vdots \\ y_{N-3} & y_{N-2} \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \end{bmatrix} = -\begin{bmatrix} y_2 \\ y_3 \\ \vdots \\ y_{N-1} \end{bmatrix}, \quad \mathbf{Ya} = -\mathbf{y}, \tag{21}$$

where $\mathbf{Y}$ is a $(N-2)\times 2$ matrix with the Hankel structure (the lower row is the upper row shifted one position to the left). In practice, estimation is based on a large number of samples (as many as thousands) with the hope that measurement noise will be averaged and cancelled this way and thus the obtained result will be more correct.

For $N = 4$ we have four measurements y_0, y_1, y_2, y_3, (21) consists of two equations with two unknowns and the solution is (`in Matlab`):

$$\mathbf{a} = -\mathbf{Y}^{-1}\mathbf{y}, \quad \det(\mathbf{Y}) \neq 0, \quad \texttt{a = -inv(Y)*y}, \tag{22}$$

where $\mathbf{Y}^{-1}$ is an inverse of the square matrix $\mathbf{Y}$. In turn, for $N > 4$ the matrix $\mathbf{Y}$ is not square, its inverse $\mathbf{Y}^{-1}$ does not exist, therefore both sides of (21) are multiplied by $\mathbf{Y}^T$ ("T" denotes transposition):

$$(\mathbf{Y}^T\mathbf{Y})\mathbf{a} = -\mathbf{Y}^T\mathbf{y} \tag{23}$$

Now, the matrix $\mathbf{Y}^T\mathbf{Y}$ is square and a solution of (23) is:

$$\mathbf{a} = -(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathtt{y}, \quad \mathtt{a = -pinv(Y)*y,} \tag{24}$$

in which the pseudo inverse of $\mathbf{Y}$:

$$\mathbf{Y}^I = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T, \tag{25}$$

replaces $\mathbf{Y}^{-1}$ used in (22).
In Matlab the matrix $\mathbf{Y}^I$ can be computed directly from its definition or using the built-in function `pinv(Y)`. Knowing the vector **a** we can calculate Ω_x and D from (19-20).

In Matlab the optimal ordinary least squares solution of (21) that takes into account features of the matrix **Y** is given by `a = -Y\y`. A summary of Matlab computations is given below (note that Matlab starts indexing from 1 not 0).

```
N = length(y); Y = hankel(y(1:N-2), y(N-2:N-1)); y = y(3:N).';
a = -Y\y; % a = -pinv(Y)*y;
z = roots( [1 fliplr(a')] ); p = log(z);
D = -real(p(1)); Om = imag(p(1));
```

Equation (21) can be solved using the following Matlab functions: **`arcov()`**, **`lscov(), prony(), stmb()`**. In the last two, transfer function coefficients of the digital linear system having the given (pre-defined) impulse response are to be found. The measured signal **y**, input to both functions, is treated as an impulse response **h** of the system, while coefficients **a** of the transfer function denominator, returned from the functions, are the solution of (21). Different methods belonging to this group are closely related to algorithms of matrix algebra and numerical analysis [22].

The simulations have shown that the Steiglitz-McBride method [23] (Matlab **`stmb()`** function), being equivalent to the iterative quadratic maximum likelihood approach [24]), gives very good results for estimating single damped sinusoids embedded in white Gaussian noise. Firstly, initial estimation of coefficients $\mathbf{a}_0$ is found using the LS Prony solution (24) of (21). Then, the unitary input impulse-excitation (Kronecker delta function) and the response to it (analyzed h_n in our case) are both filtered using the recursive digital IIR filter with calculated weights $\mathbf{a}_0$, and the filtered signals are used in the input-output LS model fitting the results with new coefficients $\mathbf{a}_1$. Now weights $\mathbf{a}_1$ are used for filtering the original input and output of the digital system, coefficients $\mathbf{a}_2$ are found, etc. After a few iterations, the processing loop is stopped and $\mathbf{a}_K$ is the final result.

The Matlab implementation of the selected covariance methods is given in programs **`fLPsig()`** and **`fSTMCB()`** [4]. In the first, three algorithms are employed for solving (21). Note that the **`tls()`** function is not taken from Matlab libraries but from the book [25]. In the second program, given in appendix B, the Prony method can be used instead of the Steiglitz-McBride method. The Burg estimation method using simultaneous forward and backward linear prediction is not appropriate for damped sinusoids and has not been tested (despite the presence of the Matlab function **`arburg()`**).

5.3. SVD-based solutions of linear prediction problems using signal samples (SVD-based covariance methods)

In this approach, a principal component approximation of the pseudo-inverse matrix $\mathbf{Y}^I$ (25), more robust to noise, is used. The selected linear prediction order is greater than 2 and equals P, such that $\min(P, N–P) \geq 2K$ where K – assumed number of sinusoids present in the analyzed signal (in our case $K = 1$). Additionally, the backward (not forward) signal self-prediction is exploited [26], i.e. $y_n = -b_1 y_{n+1} - b_2 y_{n+2} - \ldots - b_P y_{n+P}$ instead of $y_n = -a_1 y_{n-1} - a_2 y_{n-2}$. Therefore (21) is now replaced by:

$$\begin{bmatrix} y_1 & \cdots & y_{P-1} & y_P \\ y_2 & \cdots & y_P & y_{P+1} \\ \vdots & \ddots & \vdots & \vdots \\ y_{N-P} & \cdots & y_{N-2} & y_{N-1} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_P \end{bmatrix} = -\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-P-1} \end{bmatrix}, \quad \mathbf{Y}_{N-P,P}\mathbf{b}_P = -\mathbf{y}_{N-P} \tag{26}$$

and as before in (24):

$$\mathbf{b} = -\left[\left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T\right]\mathbf{y} = -\mathbf{Y}^I\mathbf{y}. \tag{27}$$

In this approach the singular value decomposition (SVD) [25] of the Hankel-type matrix **Y** is performed:

$$\mathbf{Y} = \sum_{k=1}^{\min\{P,N-P\}} \sigma_k \mathbf{u}_k \mathbf{v}_k^H, \tag{28}$$

where σ_k – singular values of **Y** ($\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots$), and $\mathbf{u}_k$, $\mathbf{v}_k$ – left and right singular vectors of **Y.** The SVD of matrix **Y** produces a diagonal matrix **S** of the same dimension as **X**, with non-negative diagonal elements in non-increasing order, and unitary matrices **U** and **V** so that $\mathbf{Y} = \mathbf{U}\cdot\mathbf{S}\cdot\mathbf{V}^T$. The SVD results are the ones used for computation of $\mathbf{Y}^I$ [25]:

$$\mathbf{Y}^I = \sum_{k=1}^{\min\{P,N-P\}} \frac{1}{\sigma_k}\mathbf{v}_k\mathbf{u}_k^H. \tag{29}$$

The exact solution (27) is given by:

$$\mathbf{b} = -\mathbf{Y}^I\mathbf{y} = -\sum_{k=1}^{\min\{P,N-P\}} \frac{1}{\sigma_k}\mathbf{v}_k\left[\mathbf{u}_k^H\mathbf{y}\right]. \tag{30}$$

In the Kumaresan-Tuft (KT) method [26] $P = 3N/4$ and the summation (30) is limited to $2K$ ($<P$) terms, i.e. to the doubled number of expected real sinusoidal signals, completing the principal component approximation of the pseudo inverse matrix $\mathbf{Y}^I$ (in our case $K = 1$). Next, zeros (roots) z_k of the polynomial $b_0 z^P + b_1 z^{P-1} + \ldots + b_{P-1} z^1 + b_P$ (b_0=1) are found (in Matlab: `roots([b0,b1,…,bP])`), and only the $2K$ that lie outside the unit circle are considered (in our case just 2 poles: z_1 and its conjugate companion z_1^*; see proof in [26]). Finally, as before, equations (19-20) are used to calculate Ω_x and D, but the value of D has to be negated. In turn, when the polynomial $b_P z^P + b_{P-1} z^{P-1} + \ldots + b_1 z^1 + b_0$ is used, we are looking for zeros inside the unit circle, and the value of D calculated from (19-20) is not negated. In the program **`fKT()`** [4] in appendix B, the Matlab code of the Kumaresan-Tufts algorithm is given. It is a modified version of the **`lpsvd.m`** function taken from matNMR software [27] that works with real not complex signals.

The total least squares (TLS) solution of the discussed problem was proposed in [28] and its mathematical foundations are presented in [25, chapter 7.7, program **`tls.m`**]. In this method, solving (26) is replaced with the minimization task in respect of $\mathbf{b}_P$:

$$\left\|\mathbf{Y}_{N-P,P}\mathbf{b}_P + \mathbf{y}_{N-P}\right\|_2 \rightarrow \min, \tag{31}$$

in which the noisy character of not just matrix $\mathbf{Y}_{N-P,P}$ but also of vector $\mathbf{y}_{N-P}$ is taken into account. Here the SVD is performed on the augmented matrix given below (with added last column $-\mathbf{y}_{N-P}$):

$$\left[\mathbf{Y}_{N-P,P} \,|\, -\mathbf{y}_{N-P}\right] = \mathbf{U} \cdot diag(\sigma_1 ... \sigma_{P+1}) \cdot \mathbf{V}^T = \sum_{k=1}^{\min(N-P,P+1)} \sigma_k \mathbf{u}_k \mathbf{v}_k^H \,, \tag{32}$$

For $\sigma_1 \geq ... \geq \sigma_P > \sigma_{P+1} > 0$ the solution is simply computed as [25]:

$$\mathbf{b}_P = \frac{\mathbf{v}_{P+1}(1:P)}{\mathbf{v}_{P+1}(P+1)}. \tag{33}$$

The SVD-based TLS approach presented above is applied in Matlab functions **`fLPsig()`**and **`fLPcor()`** [4] to the forward (not backward) self-prediction formulation, and the prediction order P equals $2K$, i.e. double the number of real signal components. The function **`tls.m`** from [25] is used there.

In a more general case it is assumed that the smallest significant value is repeated $\sigma_1 \geq ... \geq \sigma_k > \sigma_{k+1} = \sigma_{k+2} = ... = \sigma_{P+1}$ and a different (more difficult) formula for $\mathbf{b}_P$ calculation is used [25]. While $\mathbf{b}_P$ is known, the subsequent steps are the same as in the KT algorithm described above. This approach is implemented in function **`fTLS.m`** [4], which is a modification of **`tls.m`** from [25], where the exact known number of damped sinusoids is taken into account.

The constraint total least squares (CTLS) solution, proposed by Abatzoglou and Medel (1987), is not discussed here, since it is related to the IQML method [24], which is equivalent to the Steiglitz-McBride algorithm [23] presented in the previous section.

In turn, in the Matrix Pencil method [29, 30] $P = \lfloor N/3 \rfloor$, the matrix $\mathbf{Y}_{N-P,P}$ in (26) is denoted as $\mathbf{Y}_1$ and a new matrix $\mathbf{Y}_0$ is introduced:

$$\mathbf{Y}_0 = \begin{bmatrix} y_0 & \cdots & y_{P-2} & y_{P-1} \\ y_1 & \cdots & y_{P-1} & y_P \\ \vdots & \ddots & \vdots & \vdots \\ y_{N-P-1} & \cdots & y_{N-3} & y_{N-2} \end{bmatrix}. \tag{34}$$

Next, the reduced rank pseudo-inverse of $\mathbf{Y}_1$ is calculated (K – number of signal sinusoidal components, in our case K=1) and used for finding a new matrix $\mathbf{Z}$:

$$\mathbf{Z} = \tilde{\mathbf{Y}}_1^I \cdot \mathbf{Y}_0 = \left[\sum_{k=1}^{2K} \frac{1}{\sigma_k} \mathbf{v}_k \mathbf{u}_k^H\right] \mathbf{Y}_0 \,. \tag{35}$$

In the first $2K$ (we have K=1) the biggest singular-values of matrix $\mathbf{Z}$ are equal to roots z_k of the polynomial with coefficients $\mathbf{a}$; therefore, as before, for K=1, we use equations (19-20) to estimate Ω_x and D. Let us remember that the linear eigenvalue Matrix Pencil problem is defined for two squares matrices $\mathbf{A}$ and $\mathbf{B}$ as solving the equation $(\mathbf{A}-\lambda\mathbf{B})\mathbf{u}=0$, $\mathbf{u} \neq 0$ [25]. In the program **`fMatPen()`** [4] in appendix B Matlab code of the Matrix Pencil algorithm is given based on Fortran program from [30].

Further improvements of the Kumaresan-Tufts and Matrix Pencil methods were proposed in [31] and [32] respectively, while the application of higher order statistics was introduced in [33]. A simple description of the KT method is given in [34].

5.4. Classical solution of the linear prediction problem using the autocorrelation function

For $N = 4$, equation (23) can also be interpreted as:

$$(\mathbf{Y}^T\mathbf{Y})\mathbf{a} = -(\mathbf{Y}^T\mathbf{y}) \rightarrow \begin{bmatrix} r_0 & r_1 \\ r_1 & r_0 \end{bmatrix}\begin{bmatrix} a_2 \\ a_1 \end{bmatrix} = -\begin{bmatrix} r_2 \\ r_1 \end{bmatrix} \rightarrow \begin{bmatrix} r_0 & r_1 \\ r_1 & r_0 \end{bmatrix}\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -\begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \rightarrow \mathbf{R}_{yy}\mathbf{a} = -\mathbf{r}_{yy}, \quad (36)$$

where

$$r_k = \frac{1}{N-2}\sum_{n=0}^{N-3} y_n y_{n+k}, \quad k = 0, 1, 2. \quad (37)$$

Vector **a**, which is the solution of (36), minimizes the mean square prediction error (MSE). The generalization of (36) is known as the Yule-Walker equations and can be solved by the Matlab functions `aryule()` and `lpc()`. An estimate of the autocorrelation function r_k (37) of y_n needs to be found first, $\mathbf{R}_{yy}$ and $\mathbf{r}_{yy}$ are calculated next, and finally the equation $\mathbf{R}_{yy}\mathbf{a} = -\mathbf{r}_{yy}$ (36) is solved using one of the many existing methods, e.g. the one presented previously for solving (21). However, in this case, the matrix of the linear equation ($\mathbf{R}_{yy}$) is square and symmetric in contrast to **Y**, and new possibilities exist, e.g. iteratively solving the equation (36) for increasing dimension of $\mathbf{R}_{yy}$ using the Levinson-Durbin algorithm (function `levinson()` in Matlab). For example, such a method is used for fast calculations of vocal tract filter coefficients in digital speech coders in GSM telephony. Algorithms belonging to this group represent well-known classical ARMA techniques widely presented in [20, 35-37].

After multiplying both sides of (21) by y_0 and taking an expected value in the statistical sense, the following equation is obtained:

$$\begin{bmatrix} r_0 & r_1 \\ r_1 & r_2 \\ \vdots & \vdots \\ r_{N-3} & r_{N-2} \end{bmatrix}\begin{bmatrix} a_2 \\ a_1 \end{bmatrix} = -\begin{bmatrix} r_2 \\ r_3 \\ \vdots \\ r_{N-1} \end{bmatrix}, \quad \mathbf{R}_{yy}\mathbf{a} = -\mathbf{r}_{yy} \quad (38)$$

which offers new computational possibilities for $N > 4$ in comparison to (36), since in this case the matrix $\mathbf{R}_{yy}$ is not square and **a** can be calculated, for example, from equation $\mathbf{a} = -(\mathbf{R}_{yy}^T\mathbf{R}_{yy})^{-1}\mathbf{R}_{yy}^T\mathbf{r}_{yy}$, similar to (24), and not from equation $\mathbf{a} = -\mathbf{R}_{yy}^{-1}\mathbf{r}_{rr}$, similar to (22), which is a solution of (36).

Matlab implementation of autocorrelation-based estimation methods (36)(38) is given in the program `fLPcor()` [4].

5.5. EVD-based (signal subspace) solution of the linear prediction problem using autocorrelation

Setting $x_n = y_n - \varepsilon_n$ in (18) we obtain:

$$\mathbf{y}^T\mathbf{a} = \boldsymbol{\varepsilon}^T\mathbf{a}, \quad (39a)$$

where ($a_0 = 1$)

$$\mathbf{y}^T = [y_n\ y_{n-1}\ y_{n-2}], \quad \mathbf{x}^T = [x_n\ x_{n-1}\ x_{n-2}], \quad \boldsymbol{\varepsilon}^T = [\varepsilon_n\ \varepsilon_{n-1}\ \varepsilon_{n-2}], \quad \mathbf{a}^T = [a_0\ a_1\ a_2] \quad (39b)$$

After left-multiplication (39) by **y** and calculation of expected values of both sides, we have:

$$E[\mathbf{y}\mathbf{y}^T]\mathbf{a} = E[\mathbf{y}\boldsymbol{\varepsilon}^T]\mathbf{a}, \quad (40)$$

since signal x_n is uncorrelated with noise ε_n. Finally:

$$\mathbf{R}_{yy}\mathbf{a} = \sigma_\varepsilon^2\mathbf{a}, \tag{41}$$

so the vector **a** of interest is an eigenvector of the square, symmetric autocorrelation matrix $\mathbf{R}_{yy}$ associated with eigenvalue σ_ε^2 (it should be noted that the eigenvalue problem for the given matrix **A** is solving the equation $\mathbf{Av}=\lambda\mathbf{v}$; $\mathbf{v} \neq 0$ – eigenvector of **A**, λ- corresponding eigenvalue of **A**). The eigenvector **a** lies in the noise subspace and is orthogonal to eigenvectors lying in the signal subspace. Therefore, it is necessary to perform the following steps:

1) calculate the estimate of the autocorrelation function $\mathbf{r}_{yy}$ of noisy measurement signal y_n, for example using (37) as before, then build the autocorrelation matrix R_{yy}:

$$\mathbf{R}_{yy} = \begin{bmatrix} r_0 & r_1 & r_2 \\ r_1 & r_0 & r_1 \\ r_2 & r_1 & r_0 \end{bmatrix}, \tag{42}$$

2) compute its eigenvalue decomposition (EVD), e.g. using function `eig()` in Matlab:

$$\mathbf{R}_{yy} = \sum_{k=1}^{3} \lambda_k \mathbf{v}_k \mathbf{v}_k^T, \quad \lambda_1 \geq \lambda_2 \geq \lambda_3, \tag{43}$$

3) find eigenvector $\mathbf{v}_3$ associated with the smallest eigenvalue $\lambda_3 = \sigma_\varepsilon^2$ and set $\mathbf{a} = \mathbf{v}_3$. However, due to scaling incorporated in EVD, only the signal frequency can be found from roots $\{z, z^*\}$ of the polynomial **a**: $\Omega_x = |\text{imag}(\ln(z))$.

The signal subspace methods (Pisarenko, EV, Min-Norm, MUSIC, ESPRIT) are very well presented in [38] and [28]. Their brief description can also be found in [20]. In program **`fPisarenko()`** [4], only the Matlab code of the Pisarenko method is given. The other methods are not discussed since performing this algorithmic family is significantly inferior to the SVD-based methods, making direct use of signal samples (section 5.3: Kumaresan-Tufts, TLS, Matrix Pencil).

For example, in [39] the Min-Norm EVD-based method, the extension of the Pisarenko approach is compared with the Prony and FFT techniques in application to power system analysis.

6. DFT methods

The Fourier transform (FT) of an infinite discrete sequence is a continuous function of angular frequency Ω, defined as

$$X(e^{j\Omega}) = \sum_{n=-\infty}^{\infty} x_n e^{-j\Omega n}. \tag{44}$$

In (44) we intentionally use notation $X(e^{j\Omega})$ instead of $X(\Omega)$ in order to stress the connection between the FT and the Z transform [13]. For finite length sequences, DFT (discrete Fourier transform) is defined as:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j(2\pi/N)kn}, \; k = 0, 1, 2, ..., N-1. \tag{45}$$

The DFT can be efficiently calculated by FFT (fast Fourier algorithms) proposed in [40], or by recursive algorithms [41, 42] when the calculation of spectra of consecutive signal fragments (shifted by 1 sample) is required. Comparing (44) and (45), it can be observed that in the DFT (45) the Fourier transform (44) is computed only for frequencies $\Omega_k=(2\pi/N)k$, thus the DFT samples the continuous Fourier spectrum of discrete-time signal.

The DFT is derived for infinite periodic signals [13]; as a consequence the results of frequency estimation are correct only for measurement signals containing an integer number of cycles (periods for sinusoidal signals (1), $D = 0$). It means that the frequency of the signal must be equal to the DFT frequency

$$\Omega_x = \Omega_k = \frac{2\pi}{N}k, \quad k = 0,1,2,...,N-1. \tag{46}$$

A sinusoidal signal with frequency $\Omega_x = \Omega_k$ is known as synchronously or coherently sampled. In practice, the condition (46) for pure sinusoidal signals is closely fulfilled at the stage of acquisition by the PLL (Phase Locked Loop) that keeps the integer ratio between signal frequency f_x and sampling frequency f_s. It is also possible to coherently resample a non-coherently acquired signal [43].

For non-coherently sampled signals, DFT analysis is affected by spectral leakage and sampling of continuous spectrum [13, 44]. Both phenomena can be the source of unacceptable estimation errors. The influence of spectral leakage is reduced by appropriate time windows, and the impact of sampling of the continuous spectrum is mitigated by interpolated DFT (IpDFT) algorithms.

Let us now rewrite (1) in the form:

$$x_n = Ae^{-Dn}\cos(\Omega_x n+\varphi) = \frac{A}{2}\left(e^{-Dn}e^{j(\Omega_x n+\varphi)} + e^{-Dn}e^{-j(\Omega_x n+\varphi)}\right). \tag{47}$$

From the definition (45), the DFT of the first term of the sum in (47) is

$$\mathrm{DFT}\left(e^{-Dn}e^{j(\Omega_x n+\varphi)}\right) = \sum_{n=0}^{N-1} e^{-Dn}e^{j(\Omega_x n+\varphi)}e^{-j(2\pi/N)kn} = e^{j\varphi}\sum_{n=0}^{N-1} e^{(j\Omega_x - j\Omega_k - D)n}, \tag{48}$$

where $\Omega_k = (2\pi/N)k$. By using the sum of the geometrical sequence from (48) we obtain

$$X_k = \frac{A}{2}\left(e^{j\varphi}\frac{1-\lambda^N}{1-\lambda e^{-j\Omega_k}} + e^{-j\varphi}\frac{1-\lambda^{*N}}{1-\lambda^* e^{-j\Omega_k}}\right), \tag{49}$$

where $\lambda = e^{-D+j\Omega_x}$ [5, 45].
Equation (49) always holds, except for $D = 0$ and $\Omega_x = \Omega_k$; in this case we have

$$X_k = A\frac{N}{2}e^{j\varphi}, \quad \textit{for} \quad D = 0 \quad \textit{and} \quad \Omega_x = \Omega_k. \tag{50}$$

The frequency of the coherently sampled sinusoidal signal is equal to the frequency of the only nonzero DFT bin with index k. Amplitude and phase are given from (50) by

$$A = 2|X_k|/N \text{ and } \varphi = \angle X_k. \tag{51}$$

For a coherently sampled damped signal (1), its frequency equals the frequency of the DFT bin with the highest modulus, although the neighboring bins do not equal zero. As we have three unknowns (damping, amplitude and phase), we have to use at least two DFT bins (which are complex numbers) do determine them.

Let us approximate the DFT spectrum for positive frequencies by the first term of (49) and define the following ratio of DFT bins:

$$R = \frac{X_{k+1}}{X_k} \approx \frac{A}{2}\left(e^{j\varphi} \frac{1-\lambda^N}{1-\lambda e^{-j\Omega_{k+1}}} \right) / \frac{A}{2}\left(e^{j\varphi} \frac{1-\lambda^N}{1-\lambda e^{-j\Omega_k}} \right) = \frac{1-\lambda e^{-j\Omega_k}}{1-\lambda e^{-j\Omega_{k+1}}}. \tag{52}$$

From (52) we get

$$\lambda = e^{j\Omega_k} \frac{1-R}{1-Re^{-j2\pi/N}} \tag{53}$$

and finally

$$D = -\mathrm{Re}\{\ln(\lambda)\} \text{ and } \Omega_x = \mathrm{Im}\{\ln(\lambda)\}. \tag{54}$$

Solution (54) is known as the Bertocco algorithm [45].

The DFT analysis is only correct when a signal contains an integer number of cycles, that is when (47) holds. In practice it is more likely that $\Omega_x \neq \Omega_k$. The objective of interpolated DFT (IpDFT) algorithms is to interpolate the spectrum in the neighborhood of the k-th DFT bin of the highest magnitude. The Bertocco algorithm (54) presented above is an example of such an algorithm and it is a zero-order member of the Bertocco-Yoshida (BY) family of algorithms, while the Yoshida algorithm [46] is a second-order member of this group. Definitions and properties of BY algorithms are given in [7, 9].

Taking into account more than two DFT bins, derivation of alternative IpDFT methods is possible having smaller systematic errors. In this part we present the Yoshida algorithm only, since in general it performs well in comparison to other BY methods which are rectangular-window-based. In the Yoshida algorithm, the following ratio R between DFT frequency bins is computed [46]:

$$R = \frac{X_{k-2} - 2X_{k-1} + X_k}{X_{k-1} - 2X_k + X_{k+1}}. \tag{55}$$

Four successive DFT bins in (55) have the greatest magnitudes (depending on the signal's frequency, the magnitude of X_k or X_{k-1} is the highest). The frequency and damping are then calculated from the formulas:

$$\Omega_x = \frac{2\pi}{N}\mathrm{Re}\{k - 3/(R-1)\},\ D = \frac{2\pi}{N}\mathrm{Im}\{-3/(R-1)\}, \tag{56}$$

where k is the index of the DFT bin with the greatest magnitude. Matlab implementation of the Yoshida algorithm is given in the program **`fYoshida()`** [4] presented in appendix B.

In the BY family of estimation methods [7, 9], the rectangular window is used, i.e. the DFT is computed from a fragment of the signal. In order to describe some other IpDFT algorithms let us assume that the measurement signal x_n, before the DFT computation, is multiplied by the sequence w_n known as window function [13, 44]. This operation reduces the spectral leakage in the DFT spectrum by the cost of lower spectral resolution. Let us write the signal frequency in the form

$$\Omega_x = (k \pm \delta)\frac{2\pi}{N}, \quad 0 < \delta \leq 0.5, \tag{57}$$

where, as before, k is the index of the DFT bin with the greatest magnitude. The DFT interpolation task can then be stated as follows [10, 47-50]: based on the DFT spectrum V_k=DFT$\{x_n w_n\}$ of the signal x_n, analyzed with the known window w_n, determine the signal frequency correction δ. Analytical solution of this problem is only possible for Rife-Vincent class I windows (RVCI), defined in [7, 9-11]. The rectangular window is a 0-order RVCI window and the Hanning (Hann) window is a 1-order RVCI window.

An IpDFT polynomial approximation-based algorithm, valid for sinusoids without damping and for any window, even non-cosine ones, is proposed in [11].

In [51, 52], an algorithm combining linear algebra and an IpDFT concept known as the LIDFT-(The DFT Linear Interpolation Method) is proposed and discussed.

In the following part we restrict ourselves to using the Hanning window, as in general it gives reasonable tradeoffs between systematic errors and noise immunity. Let us define the following ratios of squared DFT bins of the damped signal analyzed with the Hanning window:

$$R_1 = |V_{k+1}|^2 / |V_k|^2 \text{ and } R_2 = |V_{k-1}|^2 / |V_k|^2, \tag{58}$$

where k is the index of the DFT bin with the greatest magnitude. Then the frequency correction and damping are:

$$\delta = -\frac{3}{2}\frac{R_1 - R_2}{4R_1R_2 - R_1 - R_2 - 2}, \tag{59}$$

$$D = \frac{2\pi}{N}\sqrt{\frac{(\delta+1)^2 - R_1(\delta-2)^2}{R_1 - 1}},\quad \delta \neq 0.5,\quad D = \frac{2\pi}{N}\sqrt{\frac{(\delta-1)^2 - R_2(\delta+2)^2}{R_2 - 1}},\quad \delta \neq -0.5 \qquad (60)(61)$$

Equation (59) for estimating the frequency was derived in [7], while solutions (60-61) for damping were first given in [53].

Program **`fIpDFTd()`** [4], presented in Appendix B, implements the IpDFT algorithm defined by (59-61). Estimating amplitude and phase is also possible using this method [9]. The implementation allows the use of RVC1 windows of orders ranging between M=0 and M=6.

For estimating the frequency of a pure sinusoidal signal analyzed with the Hanning window, the following IpDFT formula can be used [49]:

$$\delta = 2\frac{|V_{k+1}| - |V_{k-1}|}{2|V_k| + |V_{k-1}| + |V_{k+1}|}, \tag{62}$$

which is based on 3 maximum DFT bins (k as before). Implementation of a three-point IpDFT algorithm for sinusoidal signals is given in program **`fIpDFT()`** [4] for RVCI windows of order between 0 -6.

7. Statistical efficiency

Statistical efficiency of estimators is determined by comparison with the Cramér-Rao Lower Bound (CRLB). An unbiased estimator that reaches the CRLB is the optimal MVU (Minimum Variance Unbiased) estimator [54]. For the sinusoidal signal (1) $D = 0$ with disturbance ε_n being a zero-mean Gaussian noise with variance σ^2, the CRLB for frequency estimation (denoted as "(E)") is given by [54]

$$\operatorname{var}\left(\Omega_x^{(E)}\right) \geq \frac{12}{\eta N(N^2 - 1)}, \tag{63}$$

where η is the signal noise ratio

$$\eta = A^2/(2\sigma^2),\ S/N = 10\log_{10}(\eta),\ \text{(dB)}. \tag{64}$$

For a damped sinusoidal signal (1) with disturbance ε_n being a zero-mean Gaussian noise with variance σ^2 the CRLB is [55]

$$\operatorname{var}\left(\Omega_x^{(E)}\right) = \operatorname{var}\left(D^{(E)}\right) \approx \frac{(1-z^2)^3(1-z^{2N})}{\eta[-N^2 z^{2N}(1-z^2)^2 + z^2(1-z^{2N})^2]},\ z = |e^{-D+j\Omega}|. \qquad (65)$$

For signals analyzed with a rectangular window, (3-4) and (5-6) are the optimal MVU estimators. The reader should note the strong dependence of the variance on *N*, motivating using high values of *N*, which means high sampling rate and long observation time.

8. Experimental results and practical method comparison

In this section we present comparative results of testing the methods described above for frequency and damping estimation. We begin with systematic errors, as they give insight into whether the method is biased, followed by a discussion on the robustness against additive zero-mean Gaussian noise.

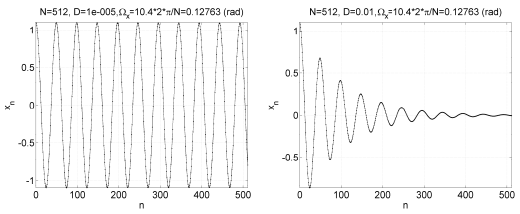

Fig. 1. Example model test signals.

Fig. 1 depicts test signals for minimum and maximum values of damping considered in the simulations, that is for $D = 10^{-5}$ and $D = 10^{-2}$. The length of the signals was set to N=512 samples and the frequency equals $\Omega_x = 10.4(2\pi)/N = 0.12763$ rad, which means that the signal contains 10.4 cycles and its frequency lies between DFT bins with indices k=10 and k=11, that is the signal is not coherently sampled. We assume that the test signal can begin with some non-zero values, because signal acquisition can start after time $t = 0$ when the impulse response begins.

In the following part, estimation methods are distinguished in figure legends by the names of the Matlab functions presented in the paper. All the functions are summarized in Table 1.

Table 1. Estimation methods considered in our final simulation (Matlab source code in [4]).

Estimation method	Description
`fOptyDFT(x,0)`	Direct model fitting in the DFT domain. Signal with rectangular window (M=0). The three DFT bins with the highest amplitudes are used for optimization.
`fHilbert(x,1)`	Hilbert transform (analytic signal) method. W=1: Hanning window. Similar to [19].
`fLPsig(x,1,3)`	Solving linear-prediction matrix equations using signal samples (different covariance methods). Assumed one damped sinusoid (1) and usage of `tls.m` program (3) from [25].
`fSTMCB(x,1,2)`	Iterative Steiglitz-McBride method of linear system identification implemented in the Matlab Signal Processing Toolbox (after a small change – the non-iterative Prony method). 1, 2 – orders of transfer function numerator and denominator.
`fKT(x,1)`	Linear prediction SVD-based Kumaresan-Tufts method with high prediction order based on function `lpsvd()` from the matNMR Matlab toolbox [27]. Assumed one damped sinusoid.
`fTLS(x,1)`	Linear prediction SVD-based TLS method with high prediction order based on function `tls.m` [25]. Assumed one damped sinusoid.
`fMatPen(x,1)`	Linear prediction SVD-based Matrix Pencil method with high prediction order based on [30]. Assumed one damped sinusoid.
`fLPcor(x,1,1,4)`	Solving linear-prediction matrix equations using signal autocorrelation samples (different autocorrelation methods). Assumed one damped sinusoid (1), biased autocorrelation estimation (1) and `tls.m` (4) function from [25].
`fPisarenko(x,1,1)`	Pisarenko method of EVD for biased signal autocorrelation estimation.
`fYoshida(x)`	Yoshida IpDFT algorithm.
`fIpDFTd(x,1)`	IpDFT algorithm for damped sinusoids with Hanning (Hann) window.
`fIpDFTd(x,6)`	IpDFT algorithm for damped sinusoids with Rife-Vincent class I order 6 window.
`fIpDFT(x,1)`	IpDFT algorithm for pure sinusoids with Hanning (Hann) window.

8.1. Systematic errors

Systematic errors of frequency and damping estimation are presented in Figs. 2-4. The errors are defined for ideal signals without any distortion. For the given value of frequency and damping, test signals were generated with the **phase being changed from $-\pi$ to π with the step $\pi/20$**, and systematic error was defined as the maximum absolute difference between the estimated and true value of frequency or damping.

Fig. 2 presents results for the signal frequency swept between the 8th and 12th DFT bin with the step 0.25. For frequencies $\Omega_x=\{8, 9, 10, 11, 12\}2\pi/N$ rad the signal is coherently sampled, which results in local minima of errors for DFT-based methods. The best precision in disturbance-free conditions is obtained by the optimization-based model fitting in the DFT domain. In practice, for the correct signal model the estimation error is only bound by the numerical precision of computer calculations. Systematic errors for DFT-based methods are a few orders higher than those of parametric methods, except **`fLPcor(x,1,1,4)`** and **`fPisarenko(x, 1,1)`** that perform poorly in the given comparison. For damping estimation, **`fLPsig(x,1,3)`**, **`fSTMCB(x,1,2)`**, **`fKT(x,1)`** and **`fTLS(x,1)`** are on a similar level of 10^{-14}-10^{-15}, and **`fMatPen(x,1)`** is slightly better, closer to 10^{-15}.

Fig. 3 depicts systematic errors for signal frequency swept between frequencies $2.1(2\pi)N$ rad and $50(2\pi)N$ rad with the step $2.5(2\pi)N$ rad. As a result, the test signals contain from 2.1 to 50 cycles. The signal was never coherently sampled in the above setup.

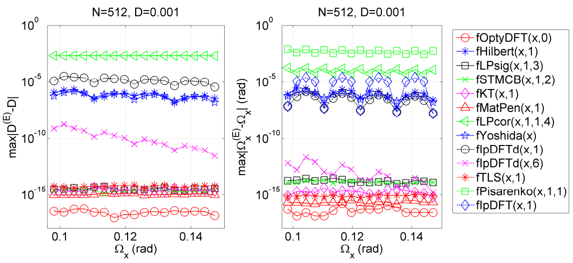

Fig. 2. Systematic errors of damping and frequency estimation as a function of frequency change in a small interval: Ω_x=(8:0.25:12)$2\pi/N$ rad.

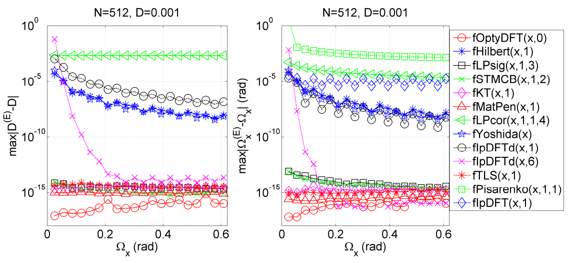

Fig. 3. Systematic errors of damping and frequency estimation as a function of frequency change in a large interval: Ω_x=(2.1:2.5:50)$2\pi/N$ rad.

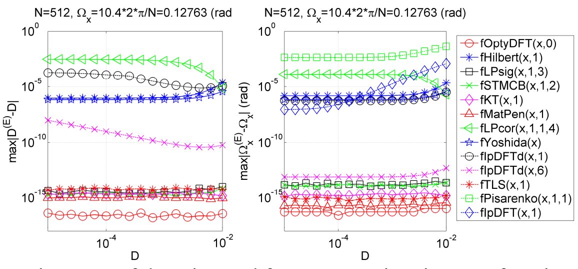

Fig. 4. Systematic errors of damping and frequency estimation as a function of damping.

Fig. 3 clearly shows that by using enough signal cycles and an adequate window, systematic errors for IpDFT methods can be negligible, as is the case for **`fIpDFTd(x,6)`**. For a small number of cycles, IpDFT methods can by strongly biased by spectral leakage, caused by two closely located poles that shift towards each other. This is the main disadvantage of IpDFT methods as compared to parametric ones, which give correct estimates even for a small number of signal cycles.

Fig. 4 shows estimation errors for increasing damping. As expected, errors for the IpDFT algorithm for sinusoidal signals, i.e. **`fIpDFT(x,1)`**, increase with damping, since the effect of the damped window is not taken into account in the derivation of this method.

8.2. Noise

Noise robustness of the considered frequency and damping estimators is shown in Figs. 5-7 using mean values and standard deviations of the observed estimation errors. For the given value of frequency and damping, the test signal was generated with the **phase being a random variable with uniform distribution in the interval from –π to π rad**, and additionally embedded in Gaussian noise with standard deviation corresponding to the assumed *S*/*N* ratio, defined by (64). For each frequency and damping, **200 realizations were generated**, and the mean value and standard deviation (std) of estimation errors were computed.

As seen from Fig. 6, the DFT-based methods **`fIpDFTd(x,1)`**, **`fIpDFTd(x,6)`** and **`fHilbert(x,1)`** are poor damping estimators, whereas the Yoshida technique performs well and is comparable with parametric methods.

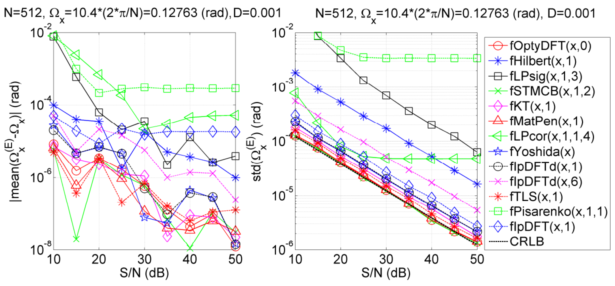

Fig. 5. Mean value and standard deviation of frequency estimation.

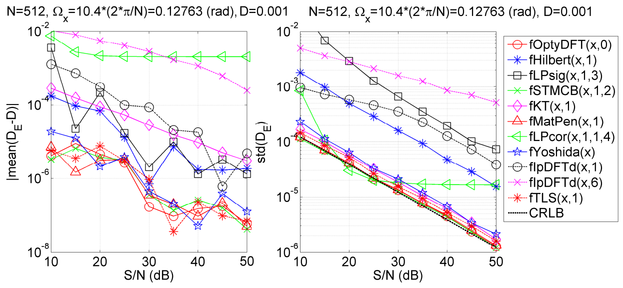

Fig. 6. Mean value and standard deviation of damping estimation.

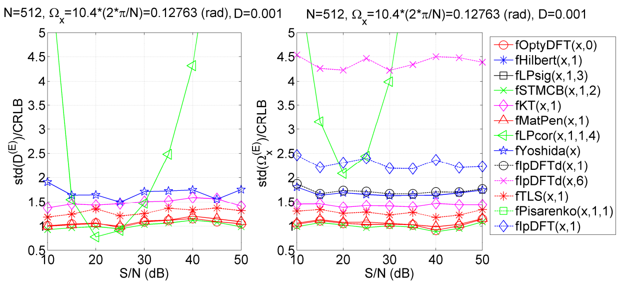

Fig. 7. Standard deviation of damping and frequency estimation with respect to CRLB.

The best results are clearly shown in Fig. 7. Algorithms **`fSTMCB(x,1,2)`**, **`fOptyDFT(x,0)`**, and **`fMatPen(x,1)`** perform similarly well and are all optimal, as they practically reach the CRLB. Slightly worse results were obtained for **`fTLS(x,1)`**, **`fKT(x,1)`** and **`fYoshida(x)`**, which is the only DFT-based method amongst the best performers. Note that **`fLPcor(x,1,1,4)`** which seems to behave well in Fig. 7 has strong bias, as seen from mean value errors shown in Figs. 5-6.

8.3. Signal length and computational complexity

Figs. 8-9 present results for frequency and damping estimation for S/N=30 dB, Ω_x=0.1 rad and changing signal length $N = \{100, 158, 251, 398, 631, 1000\}$ samples, that is for $N\Omega_x/(2\pi) \approx \{1.59, 2.51, 3.99, 6.33, 10.04, 15.92\}$ cycles. Mean values and std were computed from 200 realizations. Note that for fixed Ω_x=0.1 rad and D=10^{-3}, the signal contains relatively more noise when N increases. For this reason, after some value of N the std can start increasing.

The comparison of the best methods with respect to the CRLB is shown in Fig. 10. It is shown that the Yoshida algorithm requires a sufficient number of signal cycles for accurate performance, whereas parametric methods give good results even when the signal contains about 1.6 cycles.

Fig. 11 shows the normalized computational time estimated during simulations performed in the Matlab environment. All times were divided by the maximum time obtained for a single estimation. Computational complexity of DFT-based methods approximately equals the complexity of the FFT algorithm, as additional computing in IpDFT is negligible. It is shown in Fig. 11 that for N=1000, computational time of **`fKT(x,1)`** is about 4 orders higher than for **`fYoshida(x)`**. For N=4096 (not shown in Fig. 11), this time is approx. 6 orders higher.

For **`fOptyDFT(x,0)`**, computational time depends on the performed number of iterations during the optimization search. The DFT of the signal is computed only once and the model is fitted to the three frequency bins. Note that in time domain optimization the computational complexity is higher because in each iteration a new signal is generated (synthesized) and compared, in the sense of the chosen metric, to the measurement.

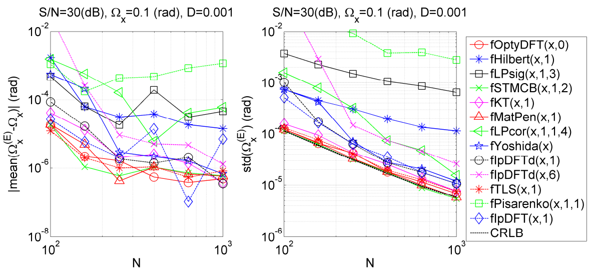

Fig. 8. Mean value and standard deviation of frequency estimation.

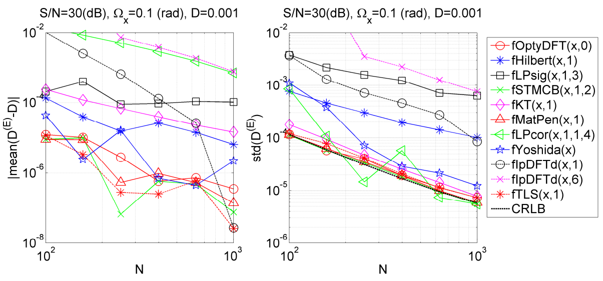

Fig. 9. Mean value and standard deviation of damping estimation.

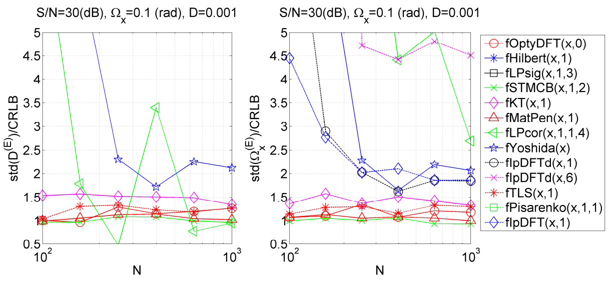

Fig. 10. Standard deviation of damping and frequency estimation with respect to CRLB.

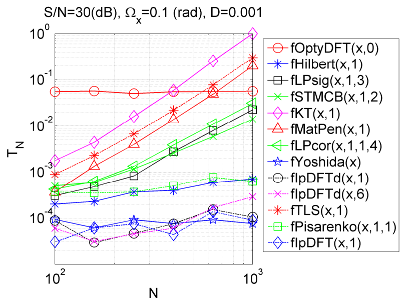

Fig. 11. Normalized computational time.

9. Conclusions

Damped sinusoidal signals are a solution of second order differential equations describing many real-world natural and technical phenomena. There are many widely used methods for estimating their parameters, especially the frequency and damping [4]. This paper briefly highlights and compares their metrological features.

The performed simulations show that for damping and frequency estimation, direct fitting in the DFT domain (**`fOptyDFT()`**) should be preferred, although for a reasonably small number of samples some parametric modeling methods can also be used, in particular the Steiglitz-McBride (**`fSTMCB()`**) and Matrix Pencil algorithms (**`FMatPen()`**), which are very accurate and robust to noise. For higher *N*, the computational complexity of the last two techniques could be a significant problem. Importantly, the Steiglitz-McBride method, iteratively solving the constraint total least squares problem, is already implemented in Matlab, while the Matrix Pencil code is very short and simple.

For signals containing thousands of samples, the IpDFT Yoshida method (**`fYoshida()`**) which generally performs very well in damping estimation is advised, while the modified

Agrež algorithm (**`fIpDFTd()`**) is suggested for frequency estimation. The IpDFT methods are also a good choice for providing a starting point to iterative minimum search algorithms.

Acknowledgments

The authors would like to thank prof. Leszek B. Magalas form the Faculty of Metal Engineering and Industrial Computer Science, AGH University of Science and Technology, for cooperation on the estimation of damped oscillation parameters in low-frequency mechanical spectroscopy [7], and, in particular for introducing them to the Yoshida algorithm.

References

[1] http://en. wikipedia.org/wiki/Oscillation#Electrical

[2] Sedlacek, M., Stoudek, Z. (2011). Active power measurements - an overview and comparison of DSP algorithms by noncoherent sampling. *Metrol. Meas. Syst.*, 18(2), 173-184.

[3] Ramos, P.M., Janeiro, F.M., Radil, T. (2010). Comparative analysis of three algorithms for two-channel common frequency sinewave parameter estimation: ellipse fit, seven parameter sine fit and spectral sinc fit. *Metrol. Meas. Syst.*, 17(2), 255-270.

[4] Source codes of all Matlab programs tested in this paper: http://kt.agh.edu.pl/~tzielin/papers/M&MS-2011/

[5] Pintelon, R., Schoukens, J. (2001). System Identification: A Frequency Domain Approach. *IEEE Press*, Piscataway (USA).

[6] Magalas, L.B. (2006). Determination of the logarithmic decrement in mechanical spectroscopy. *Solid State Phenomena*, 115, 7-14.

[7] Duda, K., Zieliński, T.P., Magalas, L.B., Majewski, M. (2011). DFT-based Estimation of Damped Oscillation Parameters in Low-frequency Mechanical Spectroscopy. *IEEE Trans. Instrum. Meas.*, 60(11), 3608-3618.

[8] Radil, T., Ramos, P.M., Serra, A.C. (2009). New Spectrum Leakage Correction Algorithm for Frequency Estimation of Power System Signals. *IEEE Trans. Instrum. Meas.*, 58(5), 1670-1679.

[9] Duda, K. (2011). Fourier-Based Estimation of Line Spectra. *AGH Publishing*, Kraków. (in Polish)

[10] Andria, G., Savino, M., Trotta, A. (1989). Windows and interpolation algorithms to improve electrical measurement accuracy. *IEEE Trans. Instrum. Meas.*, 38, 856-863.

[11] Duda, K. (2011). DFT Interpolation Algorithm for Kaiser-Bessel and Dolph-Chebyshev Windows. *IEEE Trans. Instrum. Meas.*, 60(3), 784-790.

[12] Oppenheim, A.V., Willsky, A.S., Nawab, S.H. (1997). *Signals & Systems.* Prentice Hall.

[13] Oppenheim, A.V., Schafer, R.W., Buck, J.R. (1999). *Discrete-Time Signal Processing*. Prentice-Hall.

[14] Poularikas, A.D., Seely, S. (1985). *Signals and Systems.* Boston, PWS Engineering.

[15] Agneni, A., Balis-Crema, L. (Jan., 1989). Damping measurements from truncated signals via Hilbert transform. *Mechanical Systems and Signal Processing*, 3(1), 1-13.

[16] Laila, D.S., Larsson, M., Pal, B.C., Korba, P. (2009). Nonlinear damping computation and envelope detection using Hilbert transform and its application to power systems wide area monitoring. *IEEE Power & Energy Society General Meeting*, 1-7.

[17] Magalas, L.B., Malinowski, T. (2003). Measurement Techniques for Logarithmic Decrement. *Solid State Phenomena*, 89, 247-258.

[18] Messina, A.R. et al. (2006). Interpretation and Visualization of Wide-Area PMU Measurements Using Hilbert Analysis. *IEEE Trans. Power Systems*, 21(4), 1763-1771.

[19] Shin, K., Hammond, J.K. (2007). *Fundamentals of Signal Processing for Sound and Vibration.* Wiley.

[20] Zieliński, T.P. (2005, 2007, 2009). *Digital Signal Processing: From Theory To Applications,* WKL.

[21] Proakis, J.G., Manolakis, D.G. (1992). *Digital Signal Processing: Principles, Algorithms, Applications.* Macmillan.

[22] Golub, G.G., Van Loan, Ch.F. (1996). *Matrix Computation.*3rd ed. Johns Hopkins Univ. Press.

[23] Steiglitz, K., McBride, L.E. (1965). A technique for identification of linear systems. *IEEE Trans. Automatic Control*, 10, 461-464.

[24] McClellan, J.H., Lee, D. (1991). Exact Equivalence of the Steiglitz-McBride Iteration and IQLM. *IEEE Trans. Signal Processing*, 39(2), 509-512.

[25] Moon, T.K., Stirling, W.C. (1999). *Mathematical Methods and Algorithms for Signal Processing.* Prentice Hall.

[26] Kumaresan, R., Tufts, D.W. (1982). Estimating the parameters of exponentially damped sinusoids and pole-zero modeling in noise. *IEEE Trans. Acoust. Speech Signal Processing*, ASSP-30, 837-840.

[27] Van Beek, J.D. (2007). Software from http://matnmr.sourceforge.net/. matNMR: a flexible toolbox for processing, analyzing and visualizing magnetic resonance data in Matlab. *J. Magn. Res,* 187, 19-26.

[28] Rahman, M.A., Yu, K.B. (1987). Total least squares approach for frequency estimation using linear prediction. *IEEE Trans. Acoustics. Speech Signal Processing*, 35(10), 1440-1454.

[29] Hua, Y., Sarkar, T.K. (1990). Matrix pencil method for estimating parameters of exponentially damped/undamped sinusoid in noise. *IEEE Trans. Acoustics. Speech Signal Processing*, 38(5), 814- 824.

[30] Sarkar, T.K., Pereira, O. (1995). Using the Matrix Pencil Method to Estimate the Parameters of a Sum of Complex Exponentials. *IEEE Antennas and Propagation Magazine*, 37(1), 48-55.

[31] Li, Y., Ray Liu, K.J., Razavilar, J. (1997). A Parameter estimation Scheme for Damped Sinusoidal Signals Based on Low-Rank Hankel Approximation. *IEEE Trans. Signal Process*, 45(2), 481-486.

[32] Razavilar, J., Li, Y., Ray Liu, K.J. (1998). A structured low-rank matrix pencil for spectral estimation and system identification. *Signal Processing (Elsevier)*, 65, 363-372.

[33] Ruiz, D.P., Carrion, M.C., Gallego, A., Medouri, A. (1995). Parameter Estimation of Exponentially Damped Sinusoids Using a Higher Order Correlation-Based Approach. *IEEE Trans. on Signal Processing*, 43(11), 2665-2677.

[34] Allu, G.K. (2003). Estimating the parameters of exponentially damped sinusoids in noise. University of Rhode Island, Technical Report, http://www.ele.uri.edu/~gopi/report.pdf.

[35] Kay, S.M. (1988). *Modern Spectral Estimation: Theory and Applications*, Prentice Hall.

[36] Kay, S.M., Marple, S.L. (1981). Spectrum Analysis – A Modern Perspective. *Proc. of IEEE*, 69(11), 1380-1419.

[37] Marple, S.L. (1987). *Digital Spectral Analysis with Applications.* Englewood Cliffs, Prentice Hall.

[38] Hayes, M.H. (1996). *Statistical Digital Signal Processing and Modeling.* New York, Wiley.

[39] Lobos, T., Leonowicz, Z., Rezmer, J., Schegner, P. (2006). High-Resolution Spectrum-Estimation Methods for Signal Analysis in Power Systems. *IEEE Trans. Instrum. Meas.*, 55(1), 219-225.

[40] Cooley, J.W., Tukey, J.W. (1965). An Algorithm for the Machine Computation of Complex Fourier Series. *Mathematics of Computation*, 19, 297-301.

[41] Jacobsen, E., Lyons, R. (2003). The sliding DFT. *IEEE Signal Processing Mag.*, 20(2), 74-80.

[42] Duda, K. (2010). Accurate, Guaranteed-Stable, Sliding DFT. *IEEE Signal Processing Mag.*, 124-127.

[43] Borkowski, D., Bien, A. (2009). Improvement of accuracy of power system frequency analysis by coherent resampling. *IEEE Trans. Power Delivery*, 24(2), 1004-1013.

[44] Harris, F.J. (1978). On the use of windows for harmonic analysis with the discrete Fourier transform. *Proc. IEEE*, 66, 51-83.

[45] Bertocco, M., Offeli, C., Petri, D. (1994). Analysis of damped sinusoidal signals via a frequency-domain interpolation algorithm. *IEEE Trans. Instrum. Meas.*, 43(2), 245-250.

[46] Yoshida, Y.I., Sugai, T., Tani, S., Motegi, M., Minamida, K., Hayakawa, H. (1981). Automation of internal friction measurement apparatus of inverted torsion pendulum type. *J. Phys. E: Sci. Instrum.*, 14, 1201-1206.

[47] Jain, V.K., Collins, W.L., Davis, D.C. (1979). High-Accuracy Analog Measurements via Interpolated FFT. *IEEE Trans. Instrum. Meas.*, Im-28(2), 113-122.

[48] Grandke, T. (1983). Interpolation Algorithms for Discrete Fourier Transforms of Weighted Signals. *IEEE Trans. Instrum. Meas.*, Im-32(2), 350-355.

[49] Agrež, D. (2002). Weighted Multipoint Interpolated DFT to Improve Amplitude Estimation of Multifrequency Signal. *IEEE Trans. Instrum. Meas.*, 51, 287-292.

[50] Offelli, C., Petri, D. (1990). Interpolation Techniques for Real-Time Multifrequency Waveform Analysis. *IEEE Trans. Instrum. Meas.*, 39(1), 106-111.

[51] Borkowski, J. (2000). LIDFT—The DFT Linear Interpolation Method. *IEEE Trans. Instrum. Meas.*, 49(4), 741-745.

[52] Borkowski, J., Mroczka, J. (2002). Metrological Analysis of the LIDFT Method. *IEEE Trans. Instrum. Meas.*, 51(1), 67-71.

[53] Agrež, D. (2009). A frequency domain procedure for estimation of the exponentially damped sinusoids. *International Instrumentation and Measurement Technology Conference.*

[54] Kay, S.M. (1993). *Fundamentals of Statistical Signal Processing: Estimation Theory.* Englewood Cliffs, NJ: Prentice-Hall.

[55] Yao, Y., Pandit, S.M. (1995). Cramér-Rao lower bounds for a damped sinusoidal process. *IEEE Trans. Signal Process.*, 43(4), 878-885.

[56] Duda, K. (2011). Tracking performance of digital sinusoidal signals using adaptive filters, *Electrical review*, (1), 140-143. (in Polish)

Appendix A. Estimation of amplitude and phase for known frequency and damping

In practice the most important parameters of the damped sinusoidal signal (1) are frequency Ω_x and damping D, which can equal zero. When they are already known, the remaining parameters, i.e. amplitude and phase, can be estimated using the least squares (LS) approach or the Fourier transform (FT), as shown below.

A1. LS solution for amplitude and phase

Assume that Ω_x and D are known and rewrite the signal (1) into complex form:

$$x_n = Ae^{-Dn}\cos(\Omega_x n + \varphi) = cE_{n,1} + c^* E_{n,2}\,, \tag{A.1}$$

where $c = \frac{A}{2}e^{j\varphi}$, $E_{n,1} = e^{j(\Omega_x + jD)n}$, $E_{n,2} = e^{-j(\Omega_x - jD)n}$ and "*" stands for complex conjugation. Equation (A.1) can be rewritten using matrix notation:

$$\mathbf{x} = \mathbf{Ec}\,, \quad \mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} E_{0,1} & E_{0,2} \\ E_{1,1} & E_{1,2} \\ \vdots & \vdots \\ E_{N-1,1} & E_{N-1,2} \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c \\ c^* \end{bmatrix}. \tag{A.2}$$

LS solution of (A.2) is given by [25]:

$$\hat{\mathbf{c}} = \left(\mathbf{E}^H\mathbf{E}\right)^{-1}\mathbf{E}^H\mathbf{x}\,. \tag{A.3}$$

and signal amplitude and phase are given by:

$$A = 2|c|,\ \varphi = \angle c. \tag{A.4}$$

For pure sinusoidal signal, matrix **E** is built with D=0 and the solutions (A.4) for amplitude and phase estimation still hold. The presented LS solution can be calculated iteratively as an adaptive RLS (Recursive Least Squares) filter. In [56], adaptive estimation of instantaneous frequency and amplitude of sinusoidal signal is presented with the use of RLS and LMS adaptive filters.

A2. FT solution for amplitude and phase

For known Ω_x and D, amplitude and phase of the signal (1) can be estimated by using the discrete-time Fourier transform for the frequency bin Ω_x:

$$A = |V(e^{j\Omega_x})| / \bar{m} \text{ and } \varphi = \angle V(e^{j\Omega_x}), \tag{A.5}$$

where

$$V(e^{j\Omega_x}) = \sum_{n=0}^{N-1} v_n e^{-j\Omega_x n},\ v_n = \bar{w}_n x_n,\ \ \bar{m} = \frac{1}{N}\sum_{n=0}^{N-1} \bar{w}_n \tag{A.6}$$

and $\bar{w}_n$ is an arbitrary damped time window [7]:

$$\bar{w}_n = w_n e^{-Dn}. \tag{A.7}$$

By putting D=0 in (A.7), amplitude and phase of the pure sinusoidal signal can also be estimated from (A.5).

Appendix B. Matlab source codes of four efficient algorithms for estimating frequency and damping

%==

Program 1. Estimation of Ω_x and D using the iterative Steiglitz-McBride method of linear system identification (with comparison to the noniterative Prony method).

%==

```
function [Om, D] = fSTMCB(x, NB, NA)
% x - analyzed signal – sum of damped sinusoids
% NB – transfer function numerator order   (calling choose NB=1)
% NA - transfer function denominator order (calling choose NA=2)
% Required Matlab functions stmcb() and prony()from Signal Process Toolbox
[B, A] = stmcb(x, NB, NA);    % Steiglitz-McBride method
%[B, A] = prony(x, NB, NA);   % Prony method
[z,p,k]= tf2zpk(B, A);
[val ind] = max(abs(p));
Om = abs( angle(p(ind(1))) );
D = abs( log(val) );
```

%==

Program 2. Estimation of Ω_x and D using the linear-prediction SVD-based Kumaresan-Tufts method [26, 27].

%==

```
function [Om, D] = fKT(x,K)
% x - analyzed signal – sum of real damped sinusoids
% K - assumed number of sine components
% author: Yung-Ya Lin, 12/11/97, function lpsvd.m from matNMR [27]
M = 2*K;                           % number of complex components
N = length(x);                     % number of signal samples
L = floor(N*3/4);                  % linear prediction order L = 3/4*N
Y = hankel( x(2:N-L+1), x(N-L+1:N) ); % backward prediction matrix
```

```
y = x(1:N-L)';                       % backward prediction data vector
[U,S,V] = svd(Y,0);                  % singular value decomposition
S = diag(S);                         % diagonal elements only
bias = mean(S(M+1:min([N-L,L])));    % bias compensation
b = -V(:,1:M)*(diag(1./(S(1:M)-bias))*(U(:,1:M)'*y)); % polynomial coeffs
z = roots([ 1; b]); p=log(z);        % roots, logarithm
p = p( find(real(p)>0) );            % roots outside the unit circle
Om = imag(p); [Om indx] = sort( Om, 'descend' ); Om = Om(1:K); % frequency
D = real(p(indx(1:K)));                                       % damping
%=============================================================================================================
```

Program 3. Estimation of Ω_x and D using the linear-prediction SVD-based Matrix Pencil method [29, 30].

```
%=============================================================================================================
function [Om, D] = fMatPen(x,K)
% x - analyzed signal – sum of real damped sinusoids
% K - assumed number of real damped sine components
M = 2*K;                                 % number of complex components
N = length(x);                           % number of signal samples
L = floor(N/3);                          % linear prediction order L = N/3
X = hankel(x(1:N-L),x(N-L:N)); ));       % X1=X(:,2:L+1),X0=X(:,1:L)
[U,S,V] = svd(X(:,2:L+1), 0); S = diag(S);% SVD of X1
p = log( eig( diag(1./S(1:M)) * ((U(:,1:M)'*X(:,1:L))*V(:,1:M)) ) );
Om = imag(p); [Om indx] = sort( Om, 'descend' ); Om = Om(1:K); % frequency
D = real(p(indx(1:K)));                                       % damping
%=============================================================================================================
```

Program 4. Estimation of Ω_x and D using the Yoshida IpDFT method [46].

```
%=============================================================================================================
function [Om, D]=fYoshida(x)
%x = A*cos(Om*n+p).*exp(-n*D)
N = length(x);
K = [1:round(N/2)];
Xw = fft(x);
[Xabs, ind] = max(abs(Xw(K)));
k = [K(ind)-1 K(ind) K(ind)+1];
if (K(ind)-2)>0
   if (abs( Xw(K(ind)+2) ) > abs( Xw(K(ind)-2) ))  k = [k K(ind)+2];
   else                                             k = [K(ind)-2 k];
   end
else
   k = [k K(ind)+2];
end
R = (Xw(k(1))-2*Xw(k(2))+Xw(k(3)))/(Xw(k(2))-2*Xw(k(3))+Xw(k(4)));  %(55)
D = (2*pi)/N*imag(-3/(R-1)-1); %(56)
if (k(4)-K(ind) == 1) Om = (2*pi/N)*real(K(ind)-1-3/(R-1)-2);  %(56)
else                  Om = (2*pi/N)*real(K(ind)-1-3/(R-1)-1);  %(56)
end
%=============================================================================================================
```

Program 5. Estimation of Ω_x and D using the generalized Agrež algorithm [7]

```
%=============================================================================================================
function [Om,D1,D2] = fIpDFTd(x,M)
% x = A*cos(Om*n+p).*exp(-n*D)
% M - order of RVCI window from 0 to 6
% 0-rectangular window, 1-Hanning (Hann) window
N = length(x);
K = [1:round(N/2)];
[wind, Am] = window_RVCI(N,M);
Xdft = fft(x(:).*wind(:));
[R1,R2,ind,Vk] = ratio(Xdft,K);
delt = -(2*M+1)/2*(R1-R2)/(2*(M+1)*R1*R2-R1-R2-2*M); %(64)
% delt=0.5
while abs(0.5-delt)<1e-3;
    x = [x 0];
    N = length(x),
    [wind, Am] = window_RVCI(N,M);
    Xdft = fft(x(:).*wind(:));
    [R1,R2,ind,Vk] = ratio(Xdft,K);
    delt = -(2*M+1)/2*(R1-R2)/(2*(M+1)*R1*R2-R1-R2-2*M);
end
% Damping from (60) and (61)
D1 = abs( (2*pi/N)*sqrt(abs( ((delt+M)^2-R1*(delt-M-1)^2)/(R1-1))));
D2 = abs( (2*pi/N)*sqrt(abs( ((delt-M)^2-R2*(delt+M+1)^2)/(R2-1))));
% Frequency from (57)
if ind(2)>ind(3) Om=(K(ind(1))-1+delt)*2*pi/N;
else             Om=(K(ind(1))-1-delt)*2*pi/N;
end
%------------------------------
function [wind, Am] = window_RVCI(N,ord);
% RVCI cosine windows
% N - window length, ord – window type
A(1,:)=[1 0 0 0 0 0 0 ];
A(2,:)=[1 1 0 0 0 0 0 ];
A(3,:)=[1 4/3 1/3 0 0 0 0 ];
A(4,:)=[1 3/2 3/5 1/10 0 0 0 ];
A(5,:)=[1 8/5 4/5 8/35 1/35 0 0 ];
A(6,:)=[1 105/63 60/63 45/126 5/63 1/126 0 ];
A(7,:)=[1 396/231 495/462 110/231 33/231 6/231 1/462];
A(8,:)=[0.54 0.46 0 0 0 0 0 ];
A(9,:)=[0.42 0.5 0.08 0 0 0 0 ];
dw = 2*pi/N; Om = (0:N-1)*dw;
NW=A(ord+1,:); ind=find(NW); Am=NW(ind);
wind=zeros(1,N);
for k=1:length(ind)
    wind=wind+(-1)^(k-1)*NW(k)*cos((k-1)*Om);
end
%------------------------------
function [R1,R2,ind,Vk] = ratio(Xdft,K);
[Xabs, ind] = max(abs(Xdft(K)));
Vk = Xdft(K(ind));
if K(ind)>1
   if abs(Xdft(K(ind)+1))>abs(Xdft(K(ind)-1))
      Xabs(2)=abs(Xdft(K(ind)+1));
      Xabs(3)=abs(Xdft(K(ind)-1));
      ind(2)=ind(1)+1; ind(3)=ind(1)-1;
   else
      Xabs(2)=abs(Xdft(K(ind)-1));
      Xabs(3)=abs(Xdft(K(ind)+1));
      ind(2)=ind(1)-1; ind(3)=ind(1)+1;
   end
else
   Xabs(2)=abs(Xdft(K(ind)+1));
   Xabs(3)=Xabs(2);
   ind(2)=ind(1)+1; ind(3)=ind(1)-1;
end
Xabs = Xabs.^2;
R1 = Xabs(2)/Xabs(1); %(58)
R2 = Xabs(3)/Xabs(1); %(58)
%==============================================================================================================================
```