BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATICS
DEPARTMENT OF MEASUREMENT AND INFORMATION SYSTEMS

TAMÁS ZOLTÁN BILAU, TAMÁS MEGYERI, ATTILA SÁRHEGYI

ALGORITHM AND CONVERGENCE OF THE FOUR-PARAMETER SINE FITTING

Supervisors:

István Kollár

János Márkus

Table of Contents

Abstract

Nowadays the manufacturers of analog-digital converters (ADCs) use self-defined methods for testing and datasheets for highlighting the good attributes of these devices. Therefore the engineers waste a lot of time to work at the parameters of ADCs. The solution of this problem would be a new standard method.

The Institute of Electrical and Electronics Engineers (IEEE) created the IEEE-1241 standard [1] for ADC's testing in 2001. This standard describes the full testing method. Unfortunately this description is not totally complete.

The testing procedure is simple. First we need a high precision sine wave generator. Its signal is connected to the ADC, what digitizes it. From these samples we try to estimate the original sine wave's parameters. The standard suggests a 3 and a 4 parameters sine wave fitting method. Both of these methods are Least Squares fitting procedures. In 3 parameters fitting we have to know the frequency of the sine wave. In this case the fitting is linear. When we use 4 parameters fitting we do not know the frequency, therefore this is nonlinear procedure. So it is solved by an iteration algorithm. (It derived by us, matrix-based proof of these algorithms in appendix). After the fitting we compare the measured samples with the estimated sine wave's samples. The differences of these samples (residuals) are the ADC's error. With these residuals we can calculate every test parameter like effective number of bits (ENOB), signal-to-noise and distortion ratio (SINAD) and root mean square of the residuals (RMS). These calculations are explained in the standard.

In this work some improvements to the iteration procedure of Standard 1241-2001 are suggested, and extension of the standard MATLAB program implementing the sine wave test is discussed. The program is compatible with the LABVIEW program already announced, and in other working modes offers extensions.

We presented our results and improvements in a conference in Prague [13].

Keywords: IEEE-STD-1241, ADC testing, sine wave method, three and four parameter method, Interpolated FFT, Newton-Gauss Method, Cramér-Rao Bound, LABVIEW, MATLAB.

1 INTRODUCTION

The IEEE standard on ADC testing [1] defines the way how to make a reasonable sine fit to the measured data in a sine wave test. Unfortunately, no standard can deal with all details of the associated calculation procedures. This is especially true for iterative procedures. Starting values, stop criteria, numerical details can differ from implementation to implementation, and can cause different results in different implementations even when processing the same data. This causes special problems. The so-called four-parameter fitting, described in detail in this standard (Section 4.1.4.3), is not fully defined. This procedure is of iterative nature, therefore circumstances like starting values, and way of calculation, stop criteria, etc. need to be exactly defined. While the recursive steps are precisely described in the standard the stop criterion of the iteration is not specified at all, and the setting of the starting value is not uniquely specified. However, we think that internationally reproducible results can only be achieved if these details are uniquely defined. Therefore, we need to move in the direction of more precisely fixing these algorithms – either in the standard, or is some associated document.

We have started to study this standard one year ago. We made some improvements in 4 parameters fitting algorithm and implemented it to the MATLAB program. In the followings we describe what have we done:

- We got acquainted with the fitting algorithm that is in the standard [1].
- We implemented the original algorithm in MATLAB and tested it and found some causes when it did not converge.
- We looked into why did not converge those and we modified the searching of staring value with Interpolated FFT method and we modified our algorithm in the MATLAB program. This can be seen it in Section 3.
- We made many tests of the new algorithm. We have not found any case yet when did not converge it, if we keep rules of the standard. This can be seen it in Section 3.1.
- We observed how much precision do we need for displaying in the result window of the MATLAB program. How much error estimating causes if we know how many number of bits does ADC have? This can be seen it in Section 4.1.

- We observed how many iteration steps the program needs for appropriate result. When must we stop the iteration? How large is the error of estimating? This can be seen it in Section 4.2.

- We looked into LABVIEW implementation and improved user interface of MATLAB program and made help generated for it.

- We finished the compatible mode fully and we are working on the other modes. Therefore we could compare it with LABVIEW program. We got good results. This can be seen it in Section 5.

- We could find any proof about 4 parameters algorithm therefore we made it. This can be seen it in appendix Section 7.2.

- We found a little mistake in the standard and suggested modification. The IEEE accepted it. This can be seen it in appendix Section 7.3.

- We announced our improvements in a conference in Prague [13].

The method and the program are not yet fully complete. We are working on the following problems:

- We are looking into that what can we do when iteration tries to move toward wrong direction (positive gradient). We are trying to implement the Levenberg-Marquardt method.

- We are studying the speed of convergence. It is possible that we do not need to examine the stop criteria, because the algorithm is converged in 10 steps at all cases.

- We are working on building in the user's events recording function to the program, and try to make an independent recorder for any MATLAB programs, which has standard interface.

- We would like to finish the modes of the program as soon as possible.

2 DISCUSSION

In the mathematical literature, details of iterative numerical methods are extensively discussed. Therefore, it is possible to use these to exactly define the details of our algorithm.

2.1 Starting values

Setting of the starting value is described in the standard as “Make an initial estimate of the angular frequency ω_0 of the recorded data. The frequency may be estimated by using a DFT (either on the full record or a portion of it), or by counting zero crossings, or simply by using the applied input frequency.” While this is correct from scientific viewpoint, leaving a choice to the user can hinder international reproducibility even on the same data. As [3] points out, for short records even convergence can change with the setting of the starting values, especially when the phase of the sine wave takes certain values. Also, convergence speed may depend on proper setting of the starting values. By default, the procedure needs to have at least one default way of calculation.

2.2 Calculation method

The standard number representation for scientific calculations is IEEE double precision, like in MATLAB. However, even using this, the expression (4.1.4.3.6) is numerically inefficient, and imprecise. Instead of the calculation of $x_i = \left(D_i^T D_i\right)^{-1}\left(D_i^T y\right)$, one needs to use rather matrix factorization algorithms to solve $y = D_i x_i$ [10]. The result is theoretically the same, however, in extreme cases the explicit solution may give erroneous results while the numerical solution still works.

2.3 Stop criterion

An iterative algorithm needs to perform a finite number of iterations. The problem is in general that the number of necessary iterations depends on the nature of data, so it cannot be given in advance. If we observe the change in the cost function, and the limit of change is set too high, the error will still be too large at the end, while if the limit of change is too low, we waste our time on useless iterations.

3 IMPROVEMENT OF THE STARTING VALUES

If the sampling frequency is accurately known we can use the 3-parameter fitting [1]. We have to find the parameters A, B and C, where A is the amplitude of the cosine, B is the amplitude of the sine and C is the DC value. In this case we do not have to use iteration, the algorithm converges in one step, because the error function is quadratic.

Although 3-parameter fitting with known frequency is extremely easy, usually we use 4-parameter fitting. In general we do not know the frequency precisely, because of the error of the sampling device and of the frequency generator. Naturally in this way we can eliminate the human factor (forgetting the sampling frequency), too. As we can see in Figure 1., if we do not know the sampling frequency exactly, we can make a high *rms error*, by using the 3-parameter fitting.

As it is known, the result of iteration algorithm depends on the initial guess, but this is not completely defined in the standard. There were several attempts to determine the starting frequency accurately (DFT either of the full record or a portion of it, counting zero crossings, using the applied input frequency), but our algorithm (IpFFT) gives a better result. In this article we only deal with DFT, IpFFT [11], and one other method [3].

Although IEEE-STD-1241 [1] requires that at least 4 periods of the sine wave should be sampled, and this is usually enough to avoid getting into local minima, it can be made possible that the algorithm works well even when the record length is less than 4 periods. The key is to find a good starting frequency value.

One of these attempts was M. Fonseca da Silva attempt [3]. This method based on the fact, that if we use DFT to determine the starting value, we do not make higher error than $1/(2M\Delta t)$, with Δt being the sampling time, and M the number of samples. When using this treatment we need to use 4 times the three parameter sine-fitting method, which is not so fast, especially when the sample is long. This method supposed that the shape of the error curve is known. Interpolated FFT is more systematic, and it does not need any extra resources, the execution time does not depend on the length of the sample.

When the frequency of the sinus wave can be written as $n*\Delta f$, where n is a natural number, and Δf is the frequency bin, the result of the DFT is absolutely punctual. Otherwise, you will see more component of the signal. That is called leakage. That is what

IpFFT utilizes. By applying a f relative frequency sine-wave, by using DFT we will get the following result:

$$H(m) = \frac{e^{2\pi j(m-f)} - 1}{e^{2\pi j(m-f)/N} - 1} \tag{4.0}$$

If we know the two biggest component of the FFT, the frequency of the original signal can be found by using IpFFT. If no windowing is used, an exact equation is known to interpolate the frequency:

$$f = (L + \delta) \cdot \Delta f = \lambda \cdot \Delta f \quad 0 \le \delta < 1 \tag{4.1}$$

The algorithm is as follows. The DFT of the time series is taken, and the maximum and its larger neighbor are selected, where $0 \le L < M$.

$$X(L) = U_L + jV_L \,,\, X(L+1) = U_{L+1} + jV_{L+1} \tag{4.2}$$

Then,

$$\lambda = \arccos\left(\frac{Z_2 \cos(n(L+1)) - Z_1 \cos(nL)}{Z_2 - Z_1} \right) \Big/ n \tag{4.3}$$

with

$$Z_1 = V_L \left(\frac{K_{opt} - \cos(nL)}{\sin(nL)} \right) + U_L \tag{4.4}$$

$$Z_2 = V_{L+1} \left(\frac{K_{opt} - \cos(n(L+1))}{\sin(n(L+1))} \right) + U_{L+1} \tag{4.5}$$

$$K_{opt} = \frac{(\sin(nL))(V_{L+1} - V_L) + (\cos(nL))(U_{L+1} - U_L)}{U_{L+1} - U_L} \tag{4.6}$$

where $n = 2\pi / M$.

By using IpFFT to determine the start frequency, a significantly better result can be found than by DTF, as shown in Figure. 1.

By using f_1=0.020773f_s as sine frequency and n=70 points, the four parameter fit to the sine wave,

$$y = \sin(2\pi * 0.020773 * [1:70]); \tag{4.7}$$

is bad when using DFT to estimate initial frequency, but when using IpFFT a good result is determined. With DFT we get to a local minimum at f_0=0.042293f_s, while by IpFFT we get to the global minimum with the true f_1. This is illustrated by the program, non-convergence in compatible mode, and convergence in standard mode (with IpFFT). These data are available for comparison on the Internet [12].

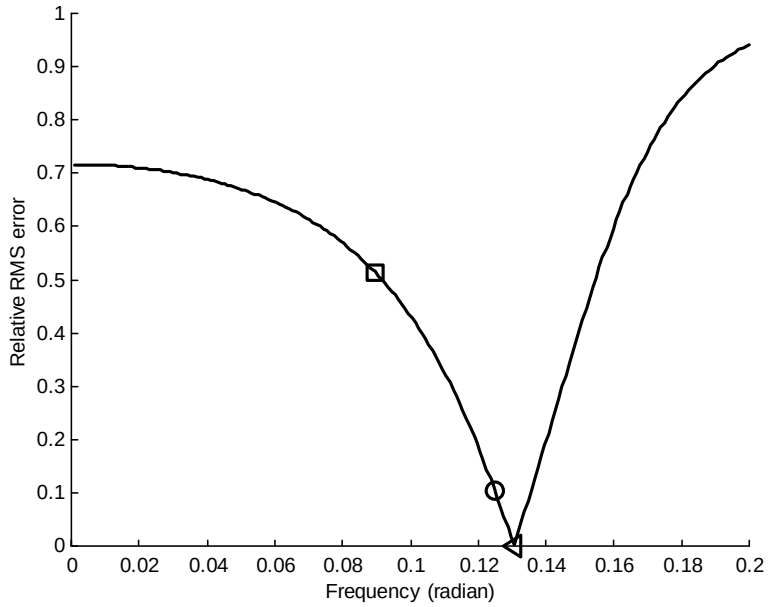

Figure 1. – The rms error of the 3-parameter algorithm (in LSB) as a function of the applied frequency value (shown as related to the nominal value). f_1=0.020773f_s,, M=70 points, no noise.

□ the frequency estimated by DFT

o the frequency estimated by Silva's method [3]

Δ the frequency estimated by IpFFT

In general, it is true that when having more than 4 periods of a sine wave, it can be recommended but it is not necessary to use IpFFT. If using IpFFT, it's somewhat more probable that we get to the global minimum and the convergence can be faster.

3.1 Testing and results

Testing has been made in order to compare our, and other methods in convergence. We tested usually less than 4 periods; because as we studied if more than 4 periods are sampled all the algorithms converge well by using simple DFT.

We randomly modified all of these parameters at same time:

- Number of bits (6…24)
- Amplitude (60%-100% of half of the full-scale)
- Signal to noise ratio (Gaussian noise, 20dB-80dB)
- Starting frequency (to be sampled 10...40 samples per period)
- Phase (0-2π)
- Number of period (1-4)
- DC (depends on the amplitude)

The number of samples is calculated from the frequency and the number of periods. As we stated, the original algorithm converges when we sampled more than 4 periods, but when sampling less then 4 periods especially about one and a half period (half period because we wanted to make the highest mistake) sometimes the algorithm does not converge.

After running 100 000 random testing cycles, we could not find any case which does not converge, or converges to a bad place. Problems with a DFT happened only when sampling about one and a half periods. When sampling more than 2 periods both algorithm converge.

3.1.1 Example

As shown in Figure 1., we get better initial frequency value, when using IpFFT. In Figure 2, we can see, the iteration steps for the same samples shown in Figure 3. The rms on the figure has been normalized. So rms error 1 is the effective value of the signal. After the 50^{th} iteration step the frequency change does not stop, although the rms does not change too much. It is because the computed amplitude is very small, as rms error we get the effective value of the signal.

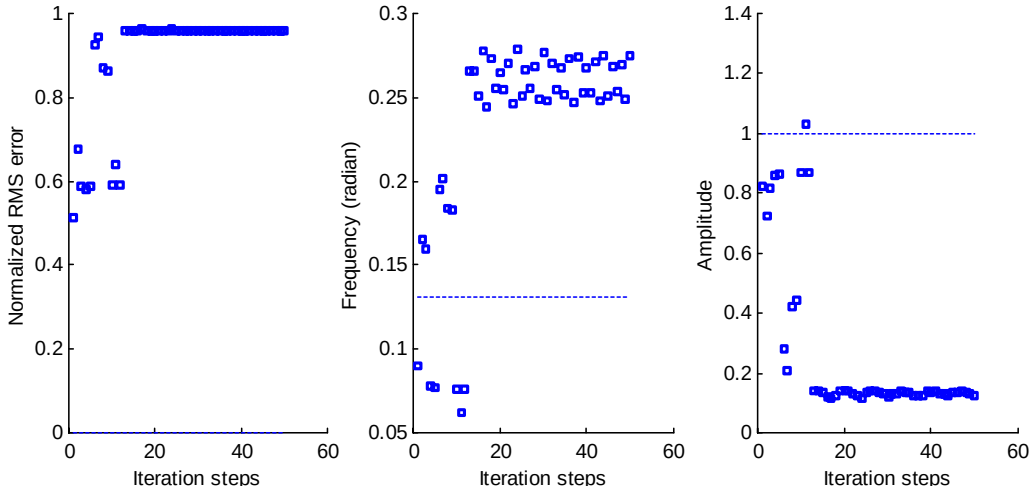

Figure 2. The rms error, the frequency and the amplitude during iteration f_1=0.020773f_s, M=70 points, no noise, high precision

□ the rms error/frequency estimated by DFT

- - the true frequency/rms error

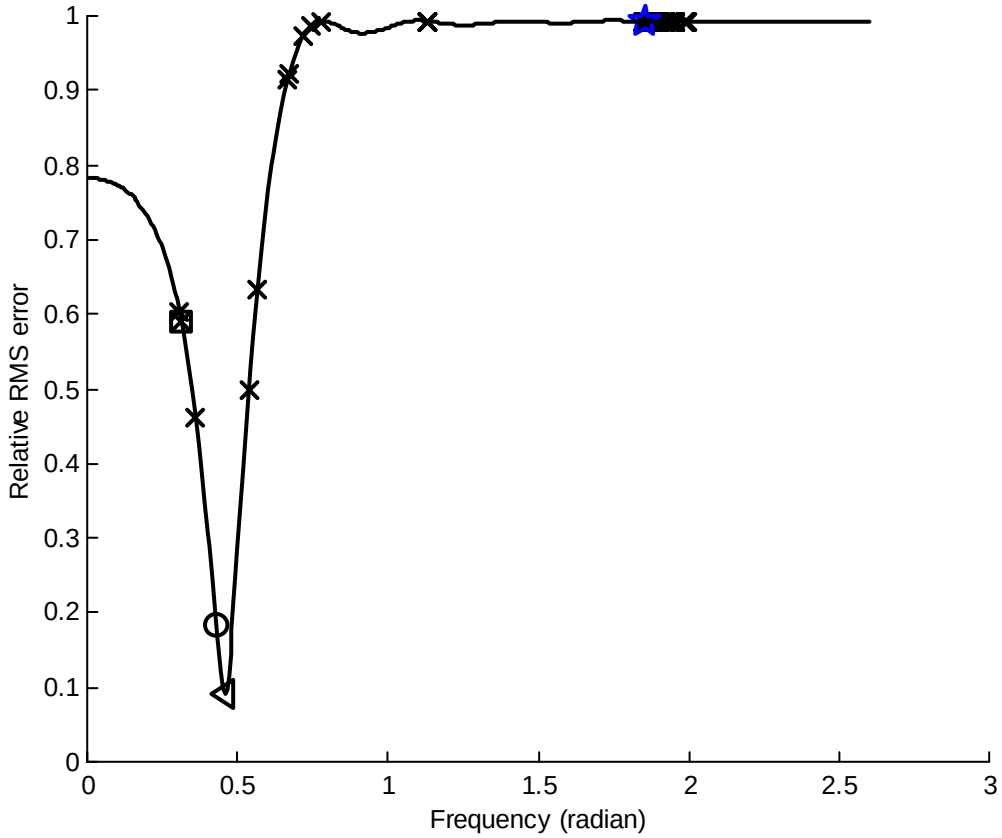

Figure 3. The rms error of the three parameter algorithm as a function of the applied frequency value in sample 01

□ the frequency estimated by DFT

o the frequency estimated by Silva's method [3]

△ the frequency estimated by IpFFT

+ the frequency during the iterations

✯ the frequency after the iteration when starting frequency is estimated by DFT

Iteration	1	2	3	4	5	6	7	8	9
Frequency	0.3142	0.5679	0.5399	0.3074	0.3599	0.6719	0.7201	0.6665	0.6665

Here we can see some typical, and non-typical non-convergence for DFT.

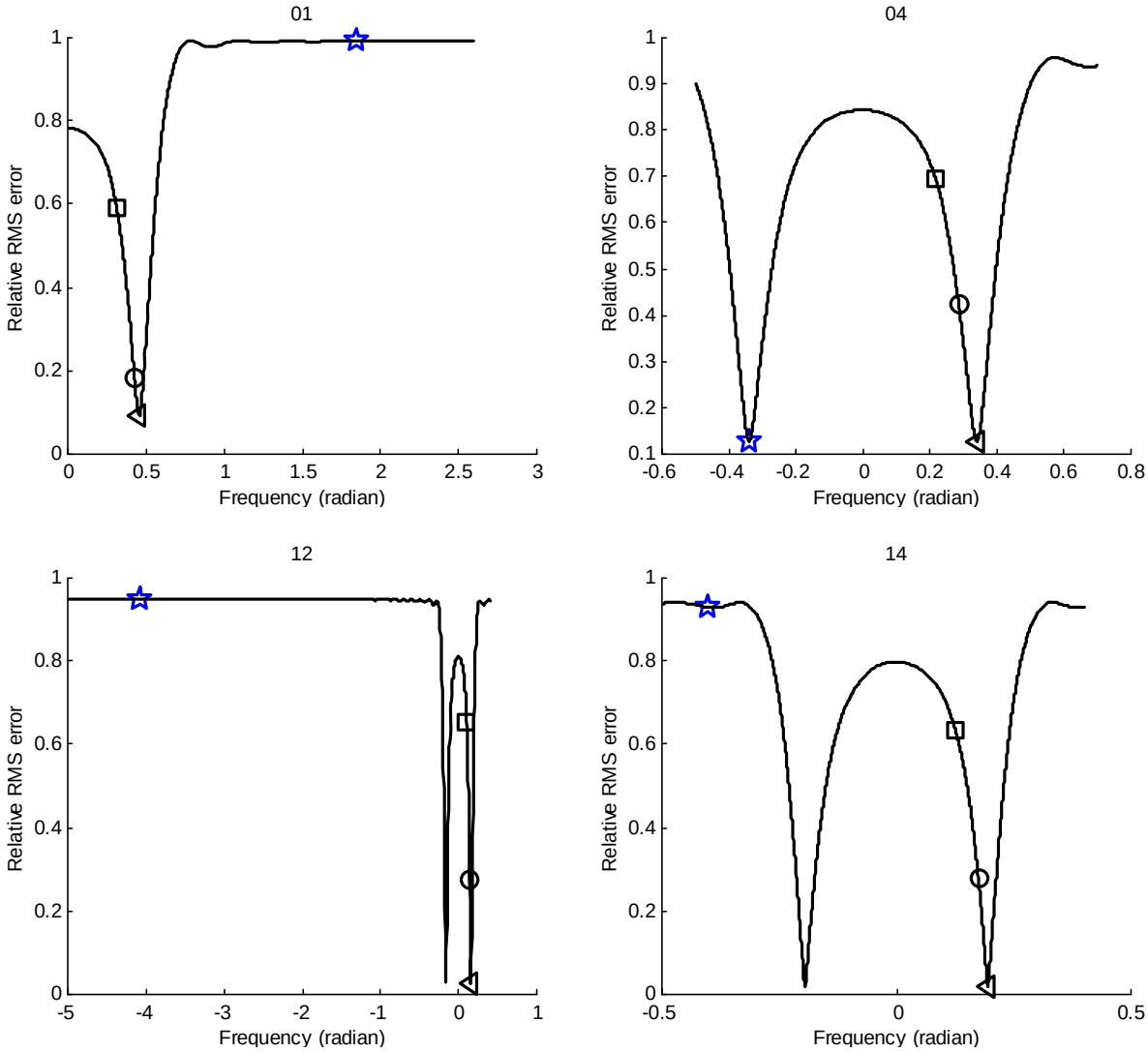

Figure 4. The rms error of the three parameter algorithm as a function of the applied frequency value

□ the frequency estimated by DFT

o the frequency estimated by Silva's method [3]

△ the frequency estimated by IpFFT

✯ the frequency after the iteration when starting frequency is estimated by DFT

	Bits	A	SNR (dB)	ω	Phase	Period	DC	N
01	23	0.6257	22.32	0.4597	4.6223	1.4617	0.1140	20
04	15	0.6607	21.37	0.3398	4.9636	1.5927	-0.0935	29
12	6	0.7586	34.99	0.1596	1.2674	1.5411	0.0232	61
14	11	0.7766	37.33	0.1951	5.2615	1.5627	-0.0335	50

For example in 01 it iterates completely bad place. It is because in the first two gradients were too big to determine the frequency. These gradients directions were opposite, but after a while the distance from the original frequency gets so high, that is not able to find the original frequency.

In 04 finally we get to (-1) multiply the original frequency. However it does not give a bad value for the rms error (we find the global minimum), which is the most important, but we cannot give this result for the user. I found some other examples, when the algorithm converged to one of the aliases (very high distance from the original), but we cannot be sure that the algorithm will find a global minimum.

In 12, 14 the initial gradients were so high, that the program was not able to find the global minimum. In Figure 3, we can see all the iteration cycle of the sample 01.

All the presented data can be found in [12]. This data is the result of simulations.

4 ANALYSIS OF FREQUENCY PRECISION

An interesting question in the 4-parameter fitting is the following: how many iterations are needed, and how accurately the result need to be displayed (non-significant digits should not be shown). The measured sine wave is imprecise, because of observation noise, quantizaton noise, parameter inaccuracies, etc. In this case the cost function analysis would be too difficult, so it needs simplification. We analyze here only the effect of frequency inaccuracy.

The 4-parameter least squares fit to a sine wave minimizes the following sum (cost function) of the squared differences:

$$e = \sum_{n=1}^{M} \left(y_n - A \cdot \cos(\omega t_n) - B \cdot \sin(\omega t_n) - C \right)^2 . \tag{5.1}$$

where: M = number of sequential samples in the record

y_n = the nth output data sample within the record

A_i, B_i, C_i, and ω_i parameters of iteration

Assume that the errors stem only from the inaccurately given frequency:

$$e = \sum_{n=1}^{M} r_n^2 = \sum_{n=1}^{M} \left(A \cdot \sin(\omega t_n) - A \cdot \sin(\omega + \Delta\omega) t_n \right)^2 \tag{5.2}$$

Utilize the next relationship:

$$\int_0^{k \cdot T} r^2 dt = \sum_{n=1}^{M} r_n^2 \cdot \Delta t \text{ . Where } \Delta t = \frac{k \cdot T}{M} . \tag{5.3}$$

Sufficiently close to the minimum, we expect that the cost function can be well approximated by a quadratic form, which can be written as (see appendix 8.1.):

$$e = \frac{1}{\Delta t} \int_0^{k \cdot T} \left(A \cdot \sin(\omega \cdot t) - A \cdot \sin(\omega + \Delta\omega) \cdot t \right)^2 dt \approx \frac{2 \cdot (A \cdot k \cdot \pi)^2 \cdot M}{3} \cdot \left(\frac{\Delta\omega}{\omega} \right)^2 \tag{5.4}$$

The rms error is

$$e_{rms} = \sqrt{\frac{1}{M} \cdot \sum_{n=1}^{M} r_n^2} = \sqrt{\frac{2}{3}} \cdot A \cdot k \cdot \pi \cdot \left(\frac{\Delta\omega}{\omega} \right) . \tag{5.5}$$

If we prescribe that the rms error is smaller than half of the standard deviation of the quantizer error:

$$e_{rms} < 0.5\sigma_q = 0.5\frac{\left(\frac{2 \cdot A}{2^N}\right)}{\sqrt{12}}, \tag{5.6}$$

where N is the number of digitized bits, we obtain

$$\frac{\Delta\omega}{\omega} < \frac{\sqrt{2}}{k \cdot \pi \cdot 2^{N+2}}. \tag{5.7}$$

Figure 5 illustrates this relationship.

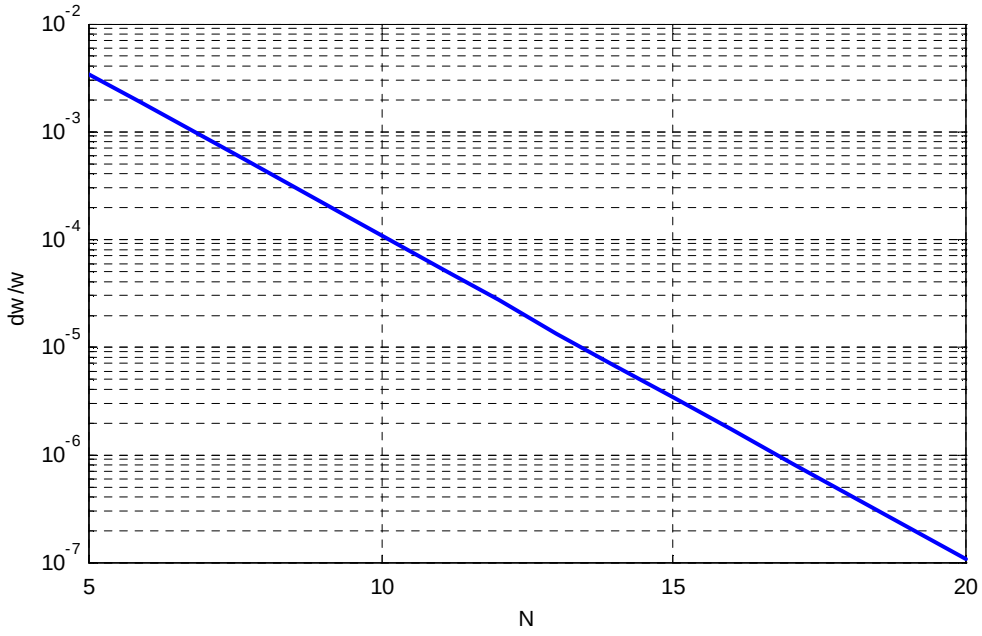

Figure 5. Allowable relative frequency error as a function of the number of bits

In a word this mean that the bigger the resolution of ADC the smaller the relative frequency error, and the result can displayed more precisely.

Some numerical examples:

If the number of bits is N=8, $\frac{\Delta\omega}{\omega} < 4.4 \cdot 10^{-4}$, so we need 5 digits.

If the number of bits is N=20, $\frac{\Delta\omega}{\omega} < 1.07 \cdot 10^{-7}$, so we need 8 digits.

4.1 The Cramér-Rao Bound

In the previous section, we analyzed the effect of frequency inaccuracy on the cost function. We arrived at the conclusion that the relative frequency error does not need to decrease to zero, because on the one hand, we would waste time on useless iterations, and on the other hand, this minimum can never be reached. There is a lower bound on the variance of the estimated parameters, so we cannot get any better estimate. This lower bound on the variance is the so-called Cramér-Rao bound (CRB).

In the distortionless case this lower bound of conditional covariance matrix can be calculated from [14]

$$\operatorname{cov}[\tilde{a},\tilde{a}\mid a]=E\left\{(\hat{\alpha}-a)\cdot(\hat{\alpha}-a)^{T}\mid a\right\}\geq J^{-1} \tag{5.1.1}$$

where J is the Fisher information matrix.

$$J_{ij}\underline{\Delta}E\left\{\frac{\partial \ln f_{z|a}(z\mid a)}{\partial a_i}\cdot\frac{\partial \ln f_{z|a}(z\mid a)}{\partial a_j}\right\}=-E\left(\frac{\partial^2 \ln f_{z|a}(z\mid a)}{\partial a_i\partial a_j}\right) \tag{5.1.2}$$

$f_{z|a}(z\mid a)$ is an N dimension conditional probability function, where the samples are independent of each other. So

$$f_{z|a}(z\mid a)=f_{y_1|a}(y_1\mid a)\cdot f_{y_2|a}(y_2\mid a)\cdot\ldots\cdot f_{y_N|a}(y_N\mid a) \tag{5.1.3}$$

The noise of observation:

$$e_k=y_k-A\cdot\cos(\omega\cdot t_k)-B\cdot\sin(\omega\cdot t_k)-C \tag{5.1.4}$$

Where $f_{y_k|a}(y_k\mid a)$ is a zero-mean Gaussian error function with variance σ^2:

$$f_{y_k|a}(y_k\mid a)=\frac{1}{\sqrt{2\cdot\pi}\cdot\sigma}\cdot\exp\left(-\frac{{e_k}^2}{2\cdot\sigma^2}\right) \tag{5.1.5}$$

$$\ln f_{z|a}(z\mid a)=\ln\left(\frac{1}{\sqrt{2\cdot\pi}\cdot\sigma}\right)^{N}-\frac{1}{2\cdot\sigma^2}\cdot\sum_{k=1}^{N}{e_k}^2 \tag{5.1.6}$$

$$\frac{\partial \ln f_{z|a}(z \mid a)}{\partial a_i} = -\frac{1}{2 \cdot \sigma^2} \cdot \sum_{k=1}^{N} \frac{\partial {e_k}^2}{\partial a_i} = -\frac{1}{\sigma^2} \cdot \sum_{k=1}^{N} e_k \cdot \frac{\partial e_k}{\partial a_i} \tag{5.1.7}$$

$$\frac{\partial \ln f_{z|a}(z \mid a)}{\partial A} = \frac{1}{\sigma^2} \cdot \sum_{k=1}^{N} e_k \cdot \cos(\omega \cdot t_k) \tag{5.1.8}$$

$$\frac{\partial \ln f_{z|a}(z \mid a)}{\partial B} = \frac{1}{\sigma^2} \cdot \sum_{k=1}^{N} e_k \cdot \sin(\omega \cdot t_k) \tag{5.1.9}$$

$$\frac{\partial \ln f_{z|a}(z \mid a)}{\partial C} = \frac{1}{\sigma^2} \cdot \sum_{k=1}^{N} e_k \tag{5.1.10}$$

$$\frac{\partial \ln f_{z|a}(z \mid a)}{\partial \omega} = \frac{1}{\sigma^2} \cdot \sum_{k=1}^{N} e_k \cdot (-A \cdot t_k \cdot \sin(\omega \cdot t_k) + B \cdot t_k \cdot \cos(\omega \cdot t_k)) \tag{5.1.11}$$

Utilize the next relations:

$E\{e_p \cdot e_q\} = 0$ if $p \neq q$, because the samples are independent of each other, and

$$E\{e_p + e_q\} = E\{e_p\} + E\{e_q\}.$$

$$\begin{aligned} J_{ij} &= E\left\{ \frac{1}{\sigma^4} \cdot \left(\sum_{k=1}^{N} e_k \cdot \frac{\partial e_k}{\partial a_i} \right) \cdot \left(\sum_{k=1}^{N} e_k \cdot \frac{\partial e_k}{\partial a_j} \right) \right\} = \\ &= \frac{1}{\sigma^4} \cdot E\left\{ \sum_{k=1}^{N} {e_k}^2 \cdot \frac{\partial^2 e_k}{\partial a_i \partial a_j} + \sum_{p=1}^{N} \sum_{\substack{q=1 \\ p \neq q}}^{N} e_p \cdot e_q \cdot \frac{\partial^2 e_k}{\partial a_i \partial a_j} \right\} = \\ &= \frac{1}{\sigma^4} \cdot \sum_{k=1}^{N} E\left\{ {e_k}^2 \cdot \frac{\partial^2 e_k}{\partial a_i \partial a_j} \right\} = \frac{1}{\sigma^4} \cdot \sum_{k=1}^{N} E\{{e_k}^2\} \cdot \frac{\partial^2 e_k}{\partial a_i \partial a_j} = \frac{1}{\sigma^2} \cdot \sum_{k=1}^{N} \frac{\partial^2 e_k}{\partial a_i \partial a_j} \end{aligned} \tag{5.1.12}$$

Thus **J** can be written as

$$J = \frac{1}{\sigma^2} \cdot \sum_{k=1}^{N} \begin{pmatrix} I_{AA} & I_{AB} & I_{AC} & I_{A\omega} \\ I_{AB} & I_{BB} & I_{BC} & I_{B\omega} \\ I_{AC} & I_{BC} & I_{CC} & I_{C\omega} \\ I_{A\omega} & I_{B\omega} & I_{C\omega} & I_{\omega\omega} \end{pmatrix} \tag{5.1.13}$$

Where

$$I_{AA} = \cos^2(\omega \cdot t_k) \tag{5.1.14}$$

$$I_{BB} = \sin^2(\omega \cdot t_k) \tag{5.1.15}$$

$$I_{CC} = 1 \tag{5.1.16}$$

$$I_{\omega\omega} = (A \cdot t_k \cdot \sin(\omega \cdot t_k) - B \cdot t_k \cdot \cos(\omega \cdot t_k))^2 \tag{5.1.17}$$

$$I_{AB} = \cos(\omega \cdot t_k) \cdot \sin(\omega \cdot t_k) \tag{5.1.18}$$

$$I_{AC} = \cos(\omega \cdot t_k) \tag{5.1.19}$$

$$I_{BC} = \sin(\omega \cdot t_k) \tag{5.1.20}$$

$$I_{A\omega} = -\cos(\omega \cdot t_k) \cdot (A \cdot t_k \cdot \sin(\omega \cdot t_k) - B \cdot t_k \cdot \cos(\omega \cdot t_k)) \tag{5.1.21}$$

$$I_{B\omega} = -\sin(\omega \cdot t_k) \cdot (A \cdot t_k \cdot \sin(\omega \cdot t_k) - B \cdot t_k \cdot \cos(\omega \cdot t_k)) \tag{5.1.22}$$

$$I_{C\omega} = -(A \cdot t_k \cdot \sin(\omega \cdot t_k) + B \cdot t_k \cdot \cos(\omega \cdot t_k)) \tag{5.1.23}$$

One can note that the information matrix is independent of the offset C. We are mainly interested in the frequency estimation error. $CRB(\omega) = \left[I^{-1}\right]_{4,4}$

Decompose **J** as

$$J = \frac{1}{\sigma^2} \cdot \begin{pmatrix} I_{11} & I_{12} \\ I_{21} & I_{22} \end{pmatrix}, \tag{5.1.24}$$

where $\mathbf{I}_{11}$ is the upper left 3×3 matrix and $I_{22} = \sum_{k=1}^{N} I_{\omega\omega}$. Then [15]

$$CRB(\omega) = \left[J^{-1}\right]_{4,4} = \frac{1}{I_{22} - I_{12}^T \cdot I_{11}^{-1} \cdot I_{22}}. \tag{5.1.25}$$

For uniform sampling $t_k = k/f_s$ with f_s being the sampling frequency, an approximation of the CRB of the absolute frequency $\hat{f}$ and large N is

$$CRB\left(\hat{f}\right) \approx \left(\frac{f_s}{2 \cdot \pi}\right)^2 \cdot \frac{2 \cdot \sigma^2}{\left(A^2 + B^2\right)} \cdot \frac{12}{N \cdot \left(N^2 - 1\right)} = \left(\frac{f_s}{2 \cdot \pi}\right)^2 \cdot \frac{12}{SNR \cdot N \cdot (N-1)} \tag{5.1.26}$$

where SNR denotes the signal-to-noise ratio, that is, $SNR = \left(A^2 + B^2\right)/2 \cdot \sigma^2$. The asymptotic result (5.1.26) only depends on the SNR. In particular, it is independent of absolute frequency and initial phase of the sine wave.

5 IMPLEMENTATION OF THE STANDARD

In ADC testing procedure we need a device or a program, which makes the sine wave estimating process. If we use a program, we will have to save the ADC's samples and then we could make the estimating with the program. In many case the recording of the ADC's samples and the estimating do not happen in same place. We are studying these estimating programs and it's algorithms, and we have joined in developing of a MATLAB implementation [5].

5.1 The first implementation of the standard (SWT VI for LABVIEW)

Jerry Blair and his co-workers made the first implementation of the standard for the United States Department of Energy. Makers named this SWT VI for Analyzing Sine Wave Test. The program is originally made for IEEE-STD-1057 [2] standard, but it is very useful for IEEE-STD-1241 [3], because the standards describe very similar testing procedures. The makers have chosen LABVIEW for implementation of the program, but LABVIEW is not often too user friendly for this application type.

5.1.1 A simple description of the standard testing method (IEEE-STD-1241)

In the first step we digitize very accurately a generated sine wave with an *Analog to Digital Converter (ADC)*, which we want to test. Then this data has to be recorded in a text-based file. If we get samples in same time periods, we haven't to record time values. But if we record only amplitude value, we need the sampling frequency of the test. The last step is estimating the input sine wave parameters from the recorded data. With this estimated values we can calculate very important parameters of the tested ADC. The program is used to make the estimating procedure, to calculate important parameters and to draw many useful figures.

5.1.2 Testing procedure in SWT VI program

The LABVIEW program assigns one description to every sine wave data file. This description contains the path of the data file, the parameters of the test (sampling fre-

quency, etc.) the parameters of the ADC (Full Scale, Number of the distinguishable levels, etc.) and the type of data file (we recorded time and amplitude values or only amplitude values). We could choose what type does have the data file, it only contains amplitude data or it contains sample time and amplitude values. If we want to start a test, we have to fill a description for our data file. The program can manages lot of descriptor in same time, but it can only process one of them. Unfortunately the program cannot save the descriptor's information to a file, that's why we have to fill again the descriptor if the program is restarted.

Here we explain a simple testing process:

- First, you have to get samples from an ADC test and put it (only the amplitude values) into a text file.
- Create a descriptor (Add new descriptor button) in the descriptor window (*Figure 6.*), which saves cluster of information about sine wave samples, and locate the sample file.
- Fill in model, serial number of the ADC and sampling frequency. Leave in time min, time max, amp min and amp max fields default values. If you put only amplitude values into the file you, leave amplitude value in data fmt field.
- If we have additional information of the test signal, we could write it in the comments field.
- Click on Process Data button and wait for the results.

After this procedure, if we have a little luck, the program makes estimates and displays a new window with the result. On the next figure (*Figure 6*) we can see the main window of the program, where we can give the starting parameters. This window is named *sw_com.vi*.

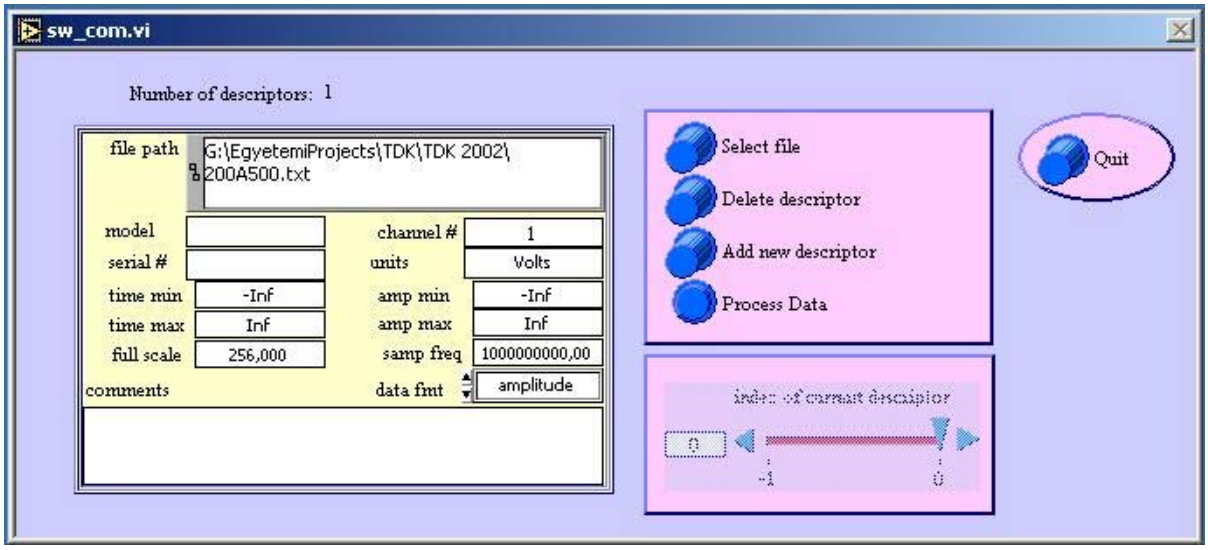

Figure 6. This is main window of the SWT VI. We can give the starting values at here.

For the simple understanding I will explain the data field's meaning in the next table (Table 1.). If we fill the necessary fields, we could start the estimating process with pushing of *Process Data* button. If the given data is correct, the program will start calculate. Unfortunately the processing time is often too long on a fast machine, and the program doesn't show, how many percent of the process has been done.

Name of field	Description
File Path	The path to a sine wave response file.
Model	The model number of waveform recorder.
Channel #	Provided for those who want to improve this program so that it can read more than one sine wave response from a single sine wave response file.
Serial #	The serial number of the waveform recorder.
Units	The units of the amplitudes of the sine response.
Time min	Discard all sine wave data with times less than time min.
Time max	Discard all sine wave data with times greater than time max.
Amp min	Discard all sine wave data with amplitudes less than amp min.
Amp max	Discard all sine wave data with amplitudes greater than amp max.
Full scale	The full scale of the waveform recorder in number of levels.
Sample freq	If the sine wave response contains amplitudes only, you must provide the sampling frequency.
Data fmt	Describes the format of the sine wave response file.
Comments	Type of any information you wish here.

Table 1. This table explains meanings of the Data Fields in SW_MAIN window.

Now for 20000 samples the calculation time is approximately 2 – 3.5 minute on a *Pentium 2 Celeron 333 MHz PC*. This time is not too long in itself, but if we need 100 calculations in one after the other, the process time will be up to 5 hours.

5.1.3 The result window

The Sine Wave Test Result window is divided to three parts, descriptive information, computed information and graphs. Descriptive information shows the starting values what we gave at descriptor creation in sw_main window. Computed information includes ADC's measured parameters (effective bits, SINAD) and the estimated sine wave parameters (amplitude, frequency, phase and DC). Unfortunately the program cannot change precision of displayed numbers when testing parameters range has changed. The result includes three graphs. The first is residual's figure. It shows variance of estimated signal. The second is PSDF diagram. This shows the integral of the power spectral density. The units on the vertical axis are the square of the units of the input data. Third is modulo time plot (Mod-T-Plot) of the signal. This shows all of the residuals plotted as a function of their phase rather than as a function time. The amplitude of the sine wave cycle has no meaning. This cycle is displayed to show the phase of each residual with respect to the input signal. On the next figure we can see a result window (Figure 7.).

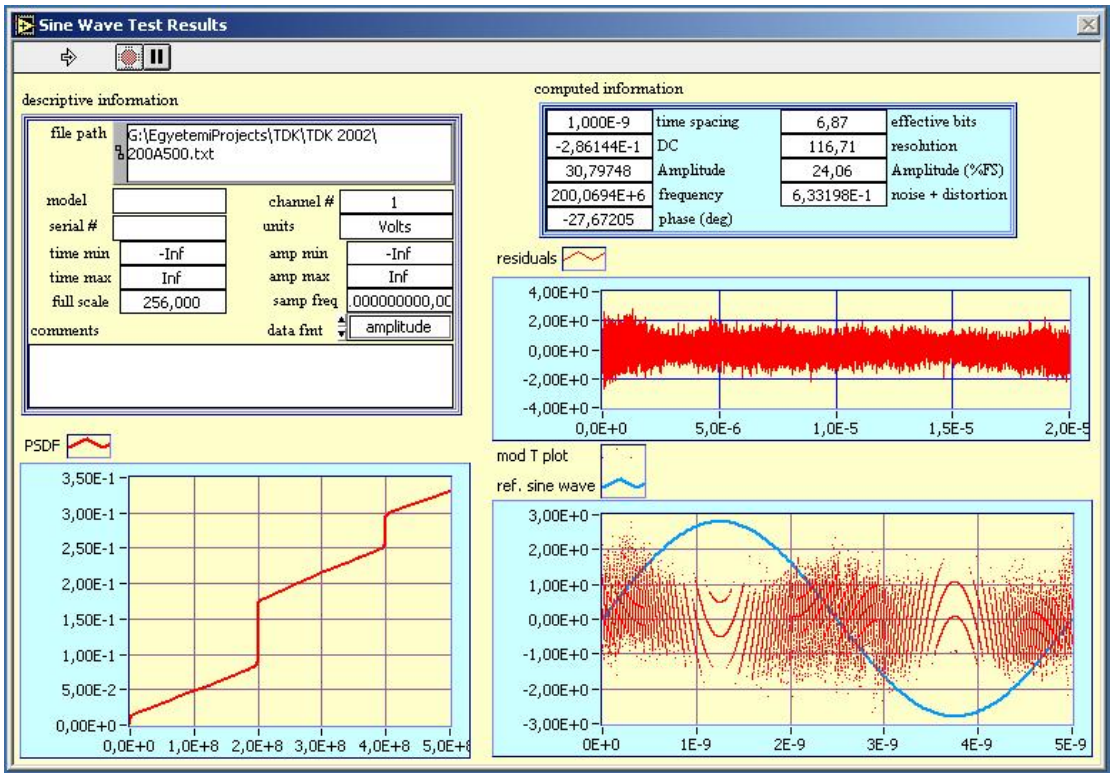

Figure 7. This is a sine wave result window.

The next table (Table 2.) includes summarized information of the computed information.

Name	Description
Time spacing	The average time between samples of the input signal.
DC	The constant term in the fitted signal.
Amplitude	The coefficient of the sinusoidal term in the fitted signal.
Frequency	The frequency of the fitted signal in units inverse to the time units of the input data.
Phase	The value of ϕ for a fitted signal of the form $A*\cos(\omega t+\phi)$
Effective bits	Calculated as in the IEEE-STD-1241
Resolution	2^{ENOB}, Where ENOB is the number of effective bits.
Amplitude (%FS)	Amplitude in percent of full scale.
Noise + Distor-tion	The rms value of the residuals.

Table 2. This is summarized information of the computed values on the result window.

5.1.4 How can I save my adjustments and results to hard disk?

Unfortunately the SWT cannot save descriptors and adjustments of that to the hard disk. Therefore we must set descriptors and starting values after we restarted the LAB-VIEW program and repeat of testing procedures are very complicated work. The actual result is saved after every processing and appended to SWT log file in the active directory of the system. This recording is made by LABVIEW system and we can see it, when the program (VI) doesn't run. The logger file is binary file that is why we can't read with a simple text editor.

5.1.5 Summary of the SWT VI

The SWT VI is the first implementation of the standard that is why there are many problems with the program. The user interface is simple, but not oblivious. We got mixed up many times and the program froze lot of times. In the next table (Table 3.) we try to summarize good and bad attributes of the program.

Good attributes	Bad attributes
Simple and good-looking user interface.	User interface is difficult and hardly using.
User can save the testing result.	The program is not stable enough. When we tested it frost many times.
Implemented the standard in a famous program. Therefore it could be got to know very simple.	Very slow algorithm
	Program does not contain any help for the user. Only a user manual is available.

Table 3. Comparing of good and bad attributes

5.2 MATLAB implementation of the standard

János Márkus and István Kollár made this MATLAB implementation of the standard [5]. The first goal of the program is keep good attributes of the LABVIEW program and improve bad attributes. We relied on that the MATLAB implementation is much faster than LABVIEW implementation, because MATLAB was developed for matrix computation and equation solving.

Main features of the MATLAB program:

- The program is based on same structure then SWT VI. It has two main parts (control and result window).
- The program uses descriptors to distinguish sine wave dataset's parameters, but program knows to save it to hard disk.
- The program has a full compatibility mode with LABVIEW SWT VI. This mode is compatible mode.
- The program contains our improvements of the standard and estimating algorithms (starting values, stop criteria).
- Makes a new improved user interface with good on-line user help.
- Improves result window and make dynamic precision of the calculated information.
- Users can use their fitting algorithms and can compare its result with our algorithm's result. This mode is advanced mode.
- The program can record events what user made, and could play it if we want. This function is good for pre-made testing procedures.
- The program can save the result to the hard disk in a useful file format.

5.2.1 Working modes of the program

This program has five working modes. This modes is one after the other:

- Compatible Mode.
- Standard Mode.
- Graphical Mode.

- Advanced Mode.
- Development Mode.

User can use four modes of all, because only developers can work development mode. These modes have different user interface and result window. These differences are explained in a latter section.

5.2.2 Properties of the user interface

Program's user interface is very similar like a normal Microsoft Windows program. It has same components (text box, check box, combo box, etc.), pop-up help, Windows help system, user menus and window control function control (close, maximize, etc.). There is one difference with two kinds of programs. MATLAB must be running if we want to use this program, because it cannot run stand-alone. These similar properties make this program very useful for first time user, because user doesn't have to get used to the new user interface.

5.2.3 Control window

Our first goal is the similar user interface to the LABVIEW program. Therefore the program has two main windows, the Control window, which makes the setting the starting values, and the Result window, which shows same structure of results like LABVIEW. This section explains main features of the control window, but it doesn't care for differences between working modes. We can look this window on next figure (Figure 8.).

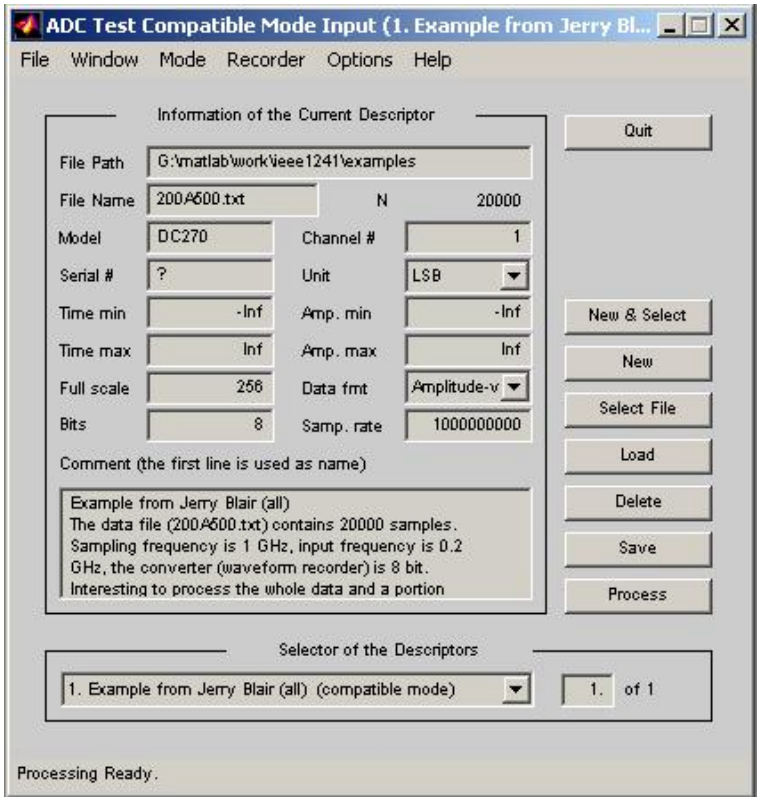

Figure 8. This is Control window (in Compatible Mode) of the MATLAB SWT program.

The structure of this is window similar like LABVIEW SWT, but we can give more information about testing. For example, we can give number of bits of the ADC and the Full Scale parameter in Compatible Mode, but the program uses only Full Scale parameter, because LABVIEW program uses it too. We can work descriptors with buttons in right side of the window. We can create, set and save descriptors. Therefore the testing parameters is set back later. Program can use several descriptors in same time, and when we process data, program calculates results of all descriptors. Accordingly user can make many tests in same time, and we can save its parameters. If we start processing, we must press process button.

5.2.4 Processing

When program uses MATLAB equation-solving features (‘\’ operator), the estimating procedure is faster than LABVIEW. This time for 20000 samples processing is 2-3 second on a *Pentium 2 Celeron 333MHz PC*. Probably this speed-up is due to MATLAB features. Unfortunately we don’t use processing bar too, because its time is very small. We improve the algorithm of processing and built its in this program. Program uses interpolated-FFT [11] for evaluating starting values and monitors cost function for stopping crite-

ria. These methods were explained in previous sections. If we make more descriptors than one, the program will process all of them. Therefore we could work with more than one sine wave at same time.

5.2.5 Result window

The result window is very various in different modes of program. Therefore we explain main features of this window. We can see result window on the next figure (Figure 9.) We can find on it same information's and figures like LABVIEW (descriptive information, computed information, PSDF, Mod-T-Plot and Residuals diagram). If we want to process more datasets than one, user could change the active result with buttons.

5.2.6 Results logging

The program doesn't make any file for latter use of results. We don't feel it necessary to the program save the result, because the testing process is very fast and we could make it in very smart time. In latter version of the program we will implement a logger function probably in order to user could make good presentation about test.

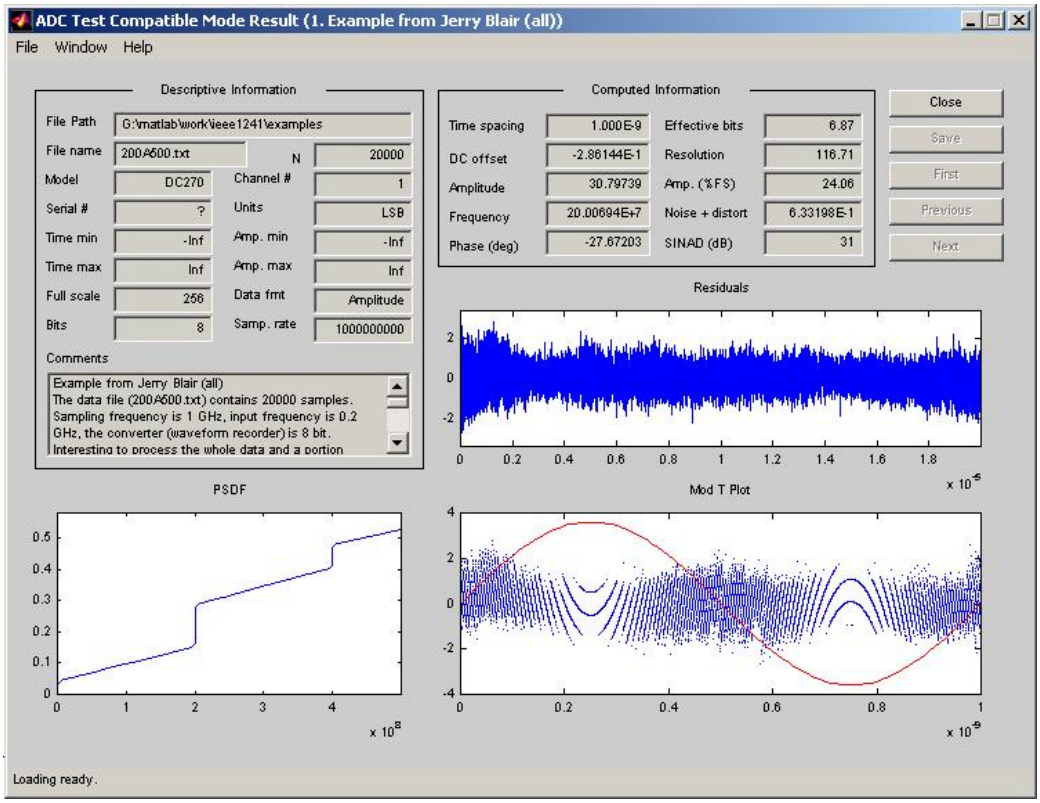

Figure 9. This is result window (Compatible mode).

5.2.7 Saving user specific method of the test

The program could save any setting and changing with user interface and program data. The name of this function is recording. This module is discrete part of the system and we could use it in any MATLAB user's interface, which contains appropriate function and structure. This recorder was developed for Frequency Identification Toolbox GUI for MATLAB [13], but we made some changes for user could use it in any MATLAB program. We can see it in Figure 10. This function would be used for making same test periodically and searching functional disorders of the program. The recorder saves any happenings with GUI and orders to this an index. After the testing, we could play back every GUI event in order and we could insert comment for every step.

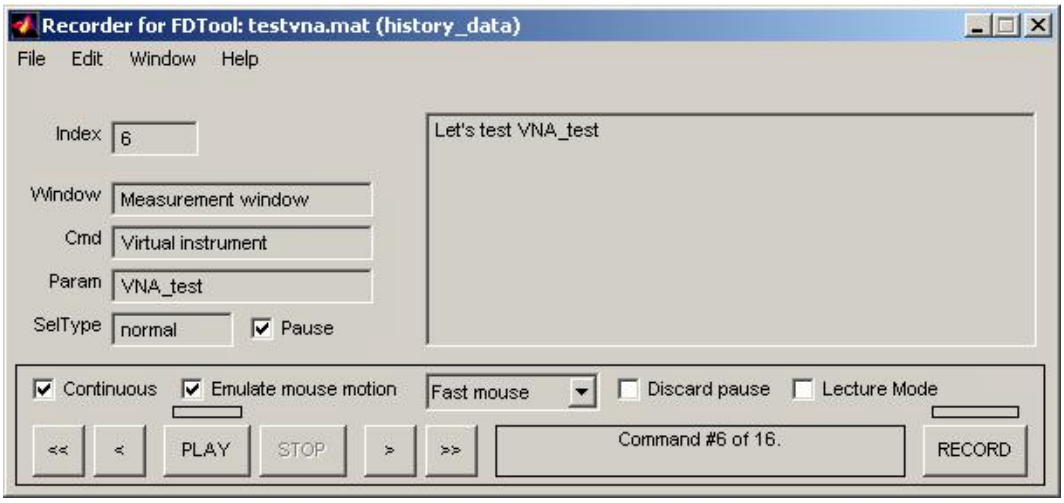

Figure 10. This is the recorder's window.

5.2.8 Help of the program

Program has two kind of helping function. The first is the on-line popup help, which appears when we move mouse icon above any window control (textbox, combo box, button, etc.). This helps explain simply and briefly the chosen control functionality. The other help is the built in MATLAB help. Text of this help we can write in Microsoft Word and we could make a MATLAB file from it with a Word Script. The program uses this mat file for creating the help.

5.2.9 The compatible mode

Intention of this mode is giving a perfectly compatible user interface with [6]. The user's interface contains two windows. The first is the *Control Window*. This window

manages the input dataset and the processing configurations of dataset. The F*igure 7* shows these windows in the LABVIEW implementations. We expand this window with two data fields (the number of the samples [N] and the number of the ADC's bits) and new functions (Load and Save descriptors). These fields don't contain any new information, there are only for the better perspicuity and the new functions facilitate for later using of the measured dates. The second window is the *Test Result Window*. In the SWT VI [6], the window includes three figures and two information divisions. *Figure 9* shows the results of the programs. Windows have very similar framework, but about the else development programs the visual appearance of the windows are different. We minded that the [5] program gives very similar number precision of the computed information. There is one difference with two programs. The [6] saves all figures to the disk, so user can use it later, but the MATLAB program can save any starting parameter (into descriptor) and user's commands (with Recorder), and we can reproduce the user's events.

5.2.10 Standard mode

In this mode we improved availability of the program. For example, when will be the number representation of the calculated values good enough? And what kind of power spectrum diagrams the user like? Distribution or consistence. We made big change in the *Control Window*. In this window we can change the type of power spectrum diagram or we can set which figure (residuals, Mod-T plot) will be visible in the result window. Buildup of the result window is similar the Compatible mode. We change precision and representation of the calculated fitting values. For example we don't set store by precision of the phase and DC, but the frequency needs to be very accurate. For our alteration of this mode we used our idea that we explain in previous sections of this document (staring value, stop criteria, Cramér-Rao bound).

5.2.11 Graphical mode

This mode has very similar functionality with the standard mode. In this mode the program has fully graphical control window and the user cannot change starting parameters. This mode is used to make very fast previously set estimating.

5.2.12 Advanced mode

The first aim is configuration of testing procedure in this mode. We can change almost everything (which fitting procedure will be used by the program, what will be the stop criteria, which method will be used for generate the staring frequency, etc.). We can set up the logging of procedures and figures. We can use this mode to try different or own algorithms and it's very usual for any experiment. Therefore this mode will be used to measure processing time of algorithm and to make very user specific output data. We can see this mode in Figure 11.

5.2.13 Developing mode

This mode has same functionality with advanced mode and it is used to test the program's functionality. Developers can test the new functionality of the program and they can make comparison of two fitting algorithms. Therefore user cannot view this part of the program.

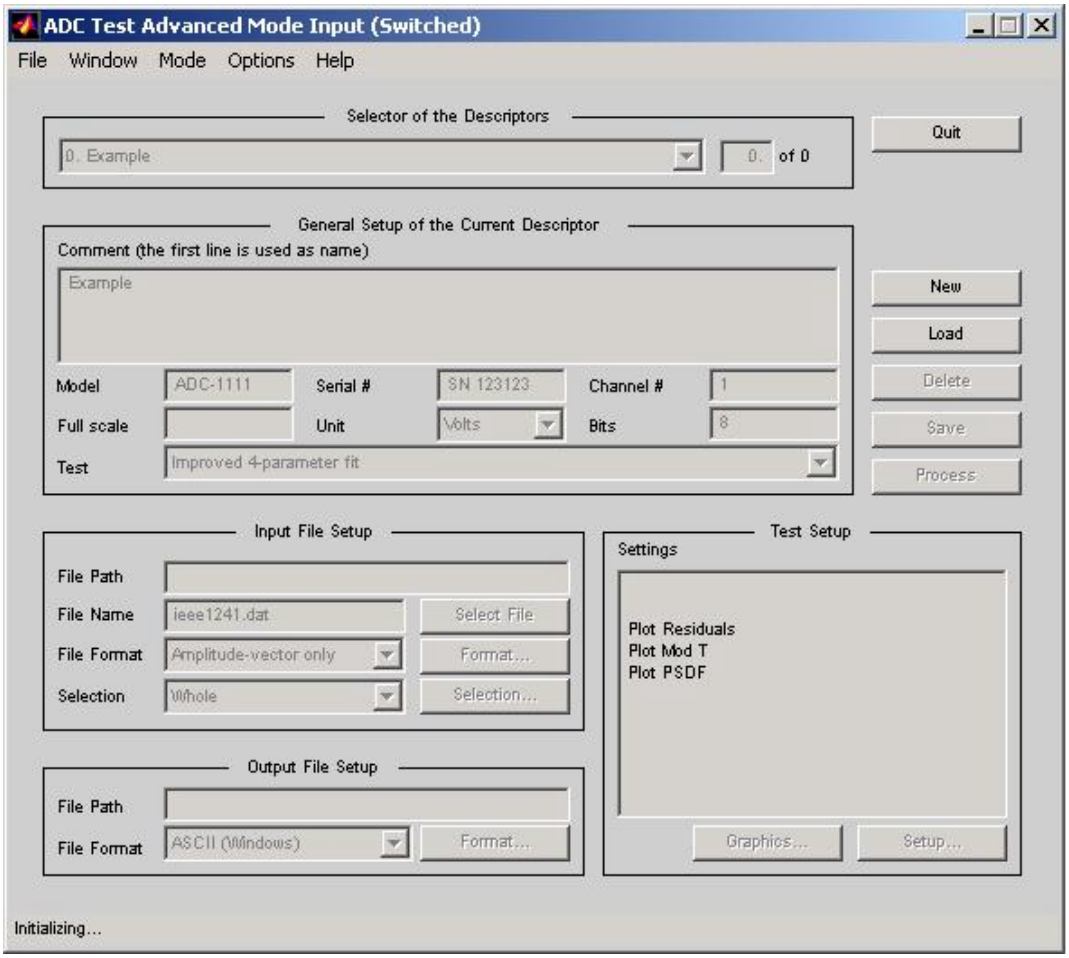

Figure 11. This is the control window in advanced mode of the MATLAB program.

5.2.14 Summary of our results

We made some improvement in the standard and in the program. These improvements make the program better speed and usability. The program cannot execute all of our goals what we describe in this section, but we can make the standardized estimating method in the compatible mode. In the following I will list that what we will have to make and we have made yet:

- We made fully the compatible mode with LABVIEW program. In this mode the program has better performance (speed and usability). We reduced the time of process about 30 times.
- We implemented out results (starting values, stop criteria and number of iteration) and built these into the program. These results can be used in the standard mode of the program which mode is half-made at this time.
- We are working on recording function of the program, we want to make available for other MATLAB programs.
- We will want to finish the other modes of the program.
- We will want to make a module, which can be used for real time data acquisition and real time estimating.

6 REFERENCES

[1] IEEE TC-10, IEEE Std 1241-2001, Standard for Terminology and Test Methods for Analog-to-Digital Converters, URL: http://grouper.ieee.org/groups/1241/.

[2] IEEE TC-10, IEEE Std 1057-94 – IEEE Standard for Digitizing Wave-form Recorders, Dec. 1994 URL: http://grouper.ieee.org/groups/1057/.

[3] M. Fonseca da Silva, A. Cruz Serra, 'Improving Convergence Of Sine Fitting Algorithms', Proc. 6 Euro Workshop on ADC Modelling and Testing, September 13-14, 2001, Lisbon, Portugal. pp. 121-124.

[4] J. Márkus and I. Kollár, 'A MATLAB Tool to Execute IEEE-STD 1241' Proc. IEEE Instrumentation and Measurement Technology Conference, IMTC/2001, Budapest, Hungary, May 21-23, 2001. pp. 1847-52.

[5] J. Márkus, 'ADC Test Data Evaluation Program for Matlab' Home Page URL: http://www.mit.bme.hu/services/ieee/ADC-test/.

[6] J. J. Blair, "Sine-fitting software for IEEE standards 1057 and 1241," in Proc. of the 16th IEEE Instr. and Meas. Technology Conference, IMTC/99, Venice, Italy, May 1999, vol. 3, pp. 1504–1506.

[7] J. Márkus and I. Kollár, "Standard environment for the sine wave test of ADC's," Submitted to the Measurement Journal of IMEKO, February, 2001.

[8] N. Giaquinto and A. Trotta, "Fast and accurate ADC testing via an enhanced sine wave fitting algorithm," IEEE Trans. on Instrumentation and Measurement, vol. 46, no. 4, pp. 1020–1024, August 1997.

[9] P. Arpaia et al., "ADC testing based on IEEE 1057-94 standard – some critical notes," in Proc. of the 17th IEEE Instr. and Meas. Technology Conference, IMTC/2000, Baltimore, Maryland USA, May 2000, vol. 1, pp. 119–124.

[10] Press, W. H., B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, Numerical Recipes: The Art of Scientific Computing. Cambridge University Press, Cambridge, 1986.

[11] Schoukens, J., R. Pintelon and H. Van hamme, 'The Interpolated Fast Fourier Transform: A Comparative Study,' *IEEE Trans. on Instrumentation and Measurement*, IM-41, No. 2, April 1992, pp. 226-32.

[12] Test data. URL: http://www.mit.bme.hu/services/ieee/ADC-test/misleading-data/

[13] T. Bilau, T. Megyeri, A. Sárhegyi, J. Márkus, I. Kollár: 'Four-Parameter Fitting of Sine Wave Testing Results – Iteration and Convergence' Proc. 4th International Conference on Advanced A/D and D/A Conversion Techniques and their Applications 7th European Workshop on ADC Modelling and Testing.

[14] Schnell László: Jelek és Rendszerek Méréstechnikája: Az átlagos négyzetes hiba alsó korlátja,Cramér-Rao egyenlőtlenség. Műegyetem Kiadó, Budapest, 1998, 177-180 old.

[15] Peter Händel: Properties of the IEEE-STD-1057 Four-Parameter Sine Wave Fit Algorithm. IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, VOL. 49, NO. 6, DECEMBER 2000

7 APPENDIX

7.1 Analysis of the effect of frequency misfit

$$e = \frac{1}{\Delta t} \cdot \int_0^{k \cdot T} \left(A \cdot \sin(\omega \cdot t) - A \cdot \sin((\omega + \Delta\omega) \cdot t)\right)^2 dt =$$

$$= \frac{A^2}{\Delta t} \cdot \int_0^{k \cdot T} \left(\sin^2(\omega \cdot t) - 2 \cdot \sin(\omega \cdot t) \cdot \sin((\omega + \Delta\omega) \cdot t) + \sin^2((\omega + \Delta\omega) \cdot t)\right) dt =$$

$$= \frac{A^2}{\Delta t} \cdot \int_0^{k \cdot T} \left(\frac{1 - \cos(2 \cdot \omega \cdot t)}{2} + \cos((2 \cdot \omega + \Delta\omega) \cdot t) - \cos(\Delta\omega \cdot t) + \frac{1 - \cos((2 \cdot \omega \cdot + 2 \cdot \Delta\omega) \cdot t)}{2}\right) dt =$$

$$= \frac{A^2}{\Delta t} \cdot \left(k \cdot T - \frac{1}{2} \cdot \frac{\sin(2 \cdot (\omega + \Delta\omega) \cdot k \cdot T)}{2 \cdot (\omega + \Delta\omega)} + \frac{\sin((2 \cdot \omega + \Delta\omega) \cdot k \cdot T)}{2 \cdot \omega + \Delta\omega} - \frac{\sin(\Delta\omega \cdot k \cdot T)}{\Delta\omega}\right) =$$

$$= \frac{A^2}{\Delta t} \cdot \left(k \cdot T - \frac{1}{2} \cdot \frac{\sin(2 \cdot \Delta\omega \cdot k \cdot T)}{2 \cdot (\omega + \Delta\omega)} + \frac{\sin(\Delta\omega \cdot k \cdot T)}{2 \cdot \omega + \Delta\omega} - \frac{\sin(\Delta\omega \cdot k \cdot T)}{\Delta\omega}\right) \approx$$

$$\approx \frac{A^2}{\Delta t} \cdot \left(k \cdot T - \frac{1}{2} \cdot \frac{2 \cdot \Delta\omega \cdot k \cdot T}{2 \cdot (\omega + \Delta\omega)} + \frac{1}{2} \cdot \frac{(2 \cdot \Delta\omega \cdot k \cdot T)^3}{2 \cdot (\omega + \Delta\omega)} \cdot \frac{1}{3!} + \frac{\Delta\omega \cdot k \cdot T}{2 \cdot \omega + \Delta\omega} - \frac{(\Delta\omega \cdot k \cdot T)^3}{2 \cdot \omega + \Delta\omega} \cdot \frac{1}{3!} - \frac{\Delta\omega \cdot k \cdot T}{\Delta\omega} + \frac{(\Delta\omega \cdot k \cdot T)^3}{\Delta\omega} \cdot \frac{1}{3!}\right) =$$

$$= \frac{A^2}{\Delta t} \cdot \left(\Delta\omega \cdot k \cdot T \cdot \left(\frac{1}{2 \cdot \omega + \Delta\omega} - \frac{1}{2 \cdot (\omega + \Delta\omega)}\right) + (\Delta\omega \cdot k \cdot T)^3 \cdot \left(\frac{2}{3} \cdot \frac{1}{2 \cdot (\omega + \Delta\omega)} - \frac{1}{6} \cdot \frac{1}{2 \cdot \omega + \Delta\omega}\right) + \frac{1}{6} \cdot \Delta\omega^2 \cdot (k \cdot T)^3\right) \approx$$

$$\approx \frac{A^2}{\Delta t} \cdot \left(\frac{1}{4} \cdot \frac{(\Delta\omega \cdot k \cdot T)^3}{\omega} + \frac{1}{6} \cdot \Delta\omega^2 \cdot (k \cdot T)^3\right) = \frac{2 \cdot A^2 \cdot k^2 \cdot \pi^2 \cdot M}{3} \cdot \left(\frac{\Delta\omega}{\omega}\right)^2 \cdot \left(\frac{3}{2} \cdot \frac{\Delta\omega}{\omega} + 1\right) \approx \frac{2}{3} \cdot (A \cdot k \cdot \pi)^2 \cdot M \cdot \left(\frac{\Delta\omega}{\omega}\right)^2$$

7.2 Matrix based proof of the algorithm for 4-parameter (General Use) Least Squares fit to sine wave data

7.2.1 General statements 1: Newton-Raphson method

Let us consider the Taylor expansion of the cost function:

$$K(\mathbf{p}+\delta\mathbf{p}) = K(\mathbf{p}) + \frac{\partial K(\mathbf{p})}{\partial \mathbf{p}} \cdot \delta\mathbf{p} + \frac{1}{2} \cdot \delta\mathbf{p}^{\mathbf{T}} \cdot \frac{\partial^2 K(\mathbf{p})}{\partial \mathbf{p}^2} \cdot \delta\mathbf{p} + \ldots \tag{8.2.1}$$

where **p** is a column vector, and $\operatorname{grad} K(\mathbf{p}) = \dfrac{\partial K(\mathbf{p})}{\partial \mathbf{p}}$ is a row vector.

For having an extreme for K in $\mathbf{p}+\delta\mathbf{p}$, its derivative with respect to $\delta\mathbf{p}$ at $\mathbf{p}+\delta\mathbf{p}$ should be equal to zero.

$$\left(\frac{\partial K(\mathbf{p}+\delta\mathbf{p})}{\partial \delta\mathbf{p}}\right)^{\mathbf{T}}_{\delta\mathbf{p}} = 0 + \left(\frac{\partial K(\mathbf{p})}{\partial \mathbf{p}}\right)^{\mathbf{T}}_{\mathbf{p}} + \left(\frac{\partial^2 K(\mathbf{p})}{\partial \mathbf{p}^2}\right)_{\mathbf{p}} \delta\mathbf{p} \tag{8.2.2}$$

The solution of this linear set of equation gives the value of $\delta\mathbf{p}$**:**

$$\delta\mathbf{p} = -\left(\frac{\partial^2 K(\mathbf{p})}{\partial \mathbf{p}^2}\right)^{-1} \left(\frac{\partial K(\mathbf{p})}{\partial \mathbf{p}}\right)^{\mathbf{T}} \tag{8.2.3}$$

7.2.2 General statements 2: Gauss-Newton method

Make use of the quadratic nature of the cost function. Let us consider the following cost function:

$$K(\mathbf{p}) = (\mathbf{y} - \mathbf{g}(\mathbf{p}))^{\mathbf{T}} \cdot (\mathbf{y} - \mathbf{g}(\mathbf{p})). \tag{8.2.4}$$

The second order derivations are given by

$$\frac{\partial^2 K(\mathbf{p})}{\partial \mathbf{p}^2} = \sum_{j=1}^{N} 2\left(\left(\frac{\partial g_j}{\partial \mathbf{p}}\right)^{\mathbf{T}} \left(\frac{\partial g_j}{\partial \mathbf{p}}\right) - (y_j - g_j)\frac{\partial^2 g_j}{\partial \mathbf{p}^2}\right) \tag{8.2.5}$$

If the second term in this sum becomes small (i.e. **y**-**g** is small) it can be discarded, and the second order derivative (the Hessian) can be approximated by

$$\frac{\partial^2 K(\mathbf{p})}{\partial \mathbf{p}^2} \approx 2\sum_{j=1}^{N}\left(\frac{\partial g_j}{\partial \mathbf{p}}\right)^{\mathbf{T}}\left(\frac{\partial g_j}{\partial \mathbf{p}}\right) = 2\left(\frac{\partial \mathbf{g}}{\partial \mathbf{p}}\right)^{\mathbf{T}}\left(\frac{\partial \mathbf{g}}{\partial \mathbf{p}}\right) = 2\mathbf{D}^{\mathbf{T}}\mathbf{D} \tag{8.2.6}$$

with the Jacobian

$$\mathbf{D} = \frac{\partial \mathbf{g}}{\partial \mathbf{p}}. \tag{8.2.7}$$

The Hessian (8.2.6) is positive semidefinite, thus grad$K(\mathbf{p})\delta\mathbf{p}$ is non-positive in (8.2.3), and the direction of the change is in the direction of decreasing K.

Subtitution of this approximation in (8.2.3), and replacement of the first derivatives of K by

$$\left(\frac{\partial K(\mathbf{p})}{\partial \mathbf{p}}\right)^{\mathbf{T}} = -2\mathbf{D}^{\mathbf{T}}(\mathbf{y} - \mathbf{g}(\mathbf{p})) \tag{8.2.8}$$

results in the Gauss-Newton method:

$$\delta\mathbf{p}_i = \left(\mathbf{D}_{i-1}^{\mathbf{T}}\mathbf{D}_{i-1}\right)^{-1}\mathbf{D}_{i-1}^{\mathbf{T}}(\mathbf{y} - \mathbf{g}(\mathbf{p}_{i-1})) \tag{8.2.9}$$

or the Least Squares solution of

$$\mathbf{D}_{i-1}\delta\mathbf{p}_i = \mathbf{y} - \mathbf{g}(\mathbf{p}_{i-1}). \tag{8.2.10}$$

7.2.3 Proof of the Least Squares Fit to Sine Wave Data Using Matrix Operations

Minimize the following quadratic cost function:

$$K = \sum_{n=1}^{M}\left[y_n - A_{i-1}\cos(\omega_{i-1}t_n) - B_{i-1}\sin(\omega_{i-1}t_n) - C_{i-1}\right]^2 = (\mathbf{y} - \mathbf{g}(\mathbf{p}))^{\mathrm{T}}(\mathbf{y} - \mathbf{g}(\mathbf{p})) \tag{8.2.11}$$

Create the following matrixes:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_M \end{bmatrix} \tag{8.2.12}$$

$$\mathbf{p}_{i-1} = \begin{bmatrix} A_{i-1} \\ B_{i-1} \\ C_{i-1} \\ \omega_{i-1} \end{bmatrix} \tag{8.2.13}$$

$$\mathbf{p}'_{i-1} = \begin{bmatrix} A_{i-1} \\ B_{i-1} \\ C_{i-1} \\ 0 \end{bmatrix} \tag{8.2.14}$$

$$\delta\mathbf{p}_i = \mathbf{p}_i - \mathbf{p}_{i-1} = \begin{bmatrix} A_i - A_{i-1} \\ B_i - B_{i-1} \\ C_i - C_{i-1} \\ \omega_i - \omega_{i-1} \end{bmatrix} = \begin{bmatrix} A_i - A_{i-1} \\ B_i - B_{i-1} \\ C_i - C_{i-1} \\ \Delta\omega_i \end{bmatrix} \tag{8.2.15}$$

$$\mathbf{D}_{i-1} = \frac{\partial \mathbf{g}(\mathbf{p}_{i-1})}{\partial \mathbf{p}_{i-1}} = \begin{bmatrix} \cos(\omega_{i-1}t_1) & \sin(\omega_{i-1}t_1) & 1 & -A_{i-1}t_1\sin(\omega_{i-1}t_1) + B_{i-1}t_1\cos(\omega_{i-1}t_1) \\ \cos(\omega_{i-1}t_2) & \sin(\omega_{i-1}t_2) & 1 & -A_{i-1}t_2\sin(\omega_{i-1}t_2) + B_{i-1}t_2\cos(\omega_{i-1}t_2) \\ . & . & & . \\ . & . & & . \\ \cos(\omega_{i-1}t_M) & \sin(\omega_{i-1}t_M) & 1 & -A_{i-1}t_M\sin(\omega_{i-1}t_M) + B_{i-1}t_M\cos(\omega_{i-1}t_M) \end{bmatrix} \tag{8.2.16}$$

The model can be expressed as

$$\mathbf{g}(\mathbf{p}_{i-1}) = \mathbf{D}_{i-1}\mathbf{p}'_{i-1}. \tag{8.2.17}$$

Substituting (8.2.17) into (8.2.9):

, (8.2.18)

and from (8.2.15):

(8.2.19)

And finally, this is clearly the LS solution of

(8.2.20)

This is important because (8.2.19) is numerically more difficult to evaluate than to solve (8.2.20) in LS sense.

The new values are A_i, B_i, C_i, and .

7.3 Suggestion to correct IEEE-STD-1241

7.3.1 Present text

4.1.4.3 An Algorithm for Four Parameter (General Use) Least Squares Fit to Sine Wave Data Using Matrix Operations.

Assuming the data record contains the sequence of M samples, y_1, y_2, ..., y_M taken at times t_1, t_2, ..., t_M, this algorithm uses an iterative process to estimate the parameters A_i, B_i, C_i and ω_i, that minimize the following sum of squared differences:

$$\sum_{n=1}^{M}\left[y_n - A_i\cos(\omega_i t_n) - B_i\sin(\omega_i t_n) - C_i\right]^2 \qquad (4.1.4.3.1)$$

where ω_i is the frequency applied to the ADC input.

- a. Set index i=0. Make an initial estimate of the angular frequency ω_0 of the recorded data. The frequency may be estimated by using a DFT (either on the full record or a portion of it), or by counting zero crossings, or simply by using the applied input frequency. Perform a prefit using the 3-parameter matrix algorithm given in clause 4.1.4.1 or clause 4.1.4.2 to estimate A_0, B_0, and C_0.
- b. Set i = i + 1 for the next iteration.
- c. Update the angular frequency estimate using:
- (4.1.4.3.2)
- d. Create the following matrices:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_M \end{bmatrix} \qquad (4.1.4.3.3)$$

$D_i =$ (4.1.4.3.4)

$$x_i = \begin{bmatrix} A_i \\ B_i \\ C_i \\ \Delta\omega_i \end{bmatrix} \quad (4.1.4.3.5)$$

- e. Compute the least-squares solution, x_i:

$$x_i = \left(D_i^T D_i\right)^{-1}\left(D_i^T y\right) \quad (4.1.4.3.6)$$

- f. Compute the amplitude, A, and phase, θ, for the form

$$y_n' = A\cos(\omega t_n + \theta) + C \quad (4.1.4.3.7)$$

using

$$A = \sqrt{A_i^2 + B_i^2} \quad (4.1.4.3.8)$$

and

$$\theta = \tan^{-1}\left[-\frac{B_i}{A_i}\right], \text{ if } A_i \geq 0 \quad (4.1.4.3.9)$$

$$\theta = \tan^{-1}\left[-\frac{B_i}{A_i}\right] + \pi, \text{ if } A_i < 0 \quad (4.1.4.3.10)$$

- g. Repeat steps b-f, recomputing the model based on the new values of A_i, B_i, and ω_i, calculated from the previous iteration. Continue to iterate until the changes in A, B, C, and ω are suitably small.

The residuals, r_n, of the fit are given by

$$r_n = y_n - A_i\cos(\omega_i t_n) - B_i\sin(\omega_i t_n) - C_i \quad (4.1.4.3.11)$$

and the rms error is given by

$$e_{rms} = \sqrt{\frac{1}{M}\sum_{n=1}^{M} r_n^2} \qquad (4.1.4.3.12)$$

7.3.2 Modification

In step g. the values A_i, B_i, C_i and ω_i do not belong to each other. The proper set is A_i, B_i, C_i and $\omega_{i+1} = \omega_i + \Delta\omega_i$. Practically, after convergence there is no problem, since $\omega_{i+1} = \omega_i$. But it would make sense to modifiy the description to have a consistent set of parameters at each step.

Suggested modifications: Replace ω_i with ω_{i-1}, and step (c) should follow step (e), so that the updated frequency is used in steps (f) and (g).

4.1.4.3 An Algorithm for Four Parameter (General Use) Least Squares Fit to Sine Wave Data Using Matrix Operations.

Assuming the data record contains the sequence of M samples, y_1, y_2, ..., y_M taken at times t_1, t_2, ..., t_M, this algorithm uses an iterative process to estimate the parameters A_i, B_i, C_i and ω_i, that minimize the following sum of squared differences:

$$\sum_{n=1}^{M}\left[y_n - A_i\cos(\omega_i t_n) - B_i\sin(\omega_i t_n) - C_i\right]^2 \qquad (4.1.4.3.1)$$

where ω_i is the frequency applied to the ADC input.

- a. Set index i=0. Make an initial estimate of the angular frequency ω_0 of the recorded data. The frequency may be estimated by using a DFT (either on the full record or a portion of it), or by counting zero crossings, or simply by using the applied input frequency. Perform a prefit using the 3-parameter matrix algorithm given in clause 4.1.4.1 or clause 4.1.4.2 to estimate A_0, B_0, and C_0.
- b. Set i = i + 1 for the next iteration.
- c. Create the following matrices:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_M \end{bmatrix} \quad (4.1.4.3.2)$$

$D_i =$ (4.1.4.3.3)

$$x_i = \begin{bmatrix} A_i \\ B_i \\ C_i \\ \Delta\omega_i \end{bmatrix} \quad (4.1.4.3.4)$$

- d. Compute the least-squares solution, x_i:

$$x_i = \left(D_i^T D_i\right)^{-1}\left(D_i^T y\right) \quad (4.1.4.3.5)$$

- Update the angular frequency estimate using:

(4.1.4.3.6)

- f. Compute the amplitude, A, and phase, θ, for the form

$$y_n' = A\cos(\omega t_n + \theta) + C \quad (4.1.4.3.7)$$

using

$$A = \sqrt{A_i^2 + B_i^2} \quad (4.1.4.3.8)$$

and

$$\theta = \tan^{-1}\left[-\frac{B_i}{A_i}\right], \text{if } A_i \geq 0 \quad (4.1.4.3.9)$$

$$\theta = \tan^{-1}\left[-\frac{B_i}{A_i}\right] + \pi, \text{ if } A_i < 0 \qquad (4.1.4.3.10)$$

- g. Repeat steps b-f, recomputing the model based on the new values of A_i, B_i, and ω_i, calculated from the previous iteration. Continue to iterate until the changes in A, B, C, and ω are suitably small.

The residuals, r_n, of the fit are given by

$$r_n = y_n - A_i\cos(\omega_i t_n) - B_i\sin(\omega_i t_n) - C_i \qquad (4.1.4.3.11)$$

and the rms error is given by

$$e_{rms} = \sqrt{\frac{1}{M}\sum_{n=1}^{M} r_n^2} \qquad (4.1.4.3.12)$$