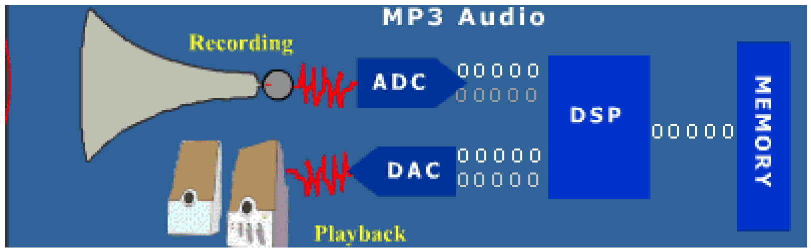
# An Introduction to Digital Signal Processors

Select a topic

**What is a DSP?**
Digital Signal Processors (DSPs) take real-world signals like voice, audio, video, temperature, pressure, or position that have been digitized and then mathematically manipulate them. A DSP is designed for performing mathematical functions like "add", "subtract", "multiply" and "divide" very quickly.

Signals need to be processed so that the information that they contain can be displayed, analyzed, or converted to another type of signal that may be of use. In the real-world, analog products detect signals such as sound, light, temperature or pressure and manipulate them. Converters such as an Analog-to-Digital converter then take the real-world signal and turn it into the digital format of 1's and 0's. From here, the DSP takes over by capturing the digitized information and processing it. It then feeds the digitized information back for use in the real world. It does this in one of two ways, either digitally or in an analog format by going through a Digital-to-Analog converter. All of this occurs at very high speeds.

To illustrate this concept, the diagram below shows how a DSP is used in an MP3 audio player. During the recording phase, analog audio is input through a receiver or other source. This analog signal is then converted to a digital signal by an analog-to-digital converter and passed to the DSP. The DSP performs the MP3 encoding and saves the file to memory. During the playback phase, the file is taken from memory, decoded by the DSP and then converted back to an analog signal through the digital-to-analog converter so it can be output through the speaker system. In a more complex example, the DSP would perform other functions such as volume control, equalization and user interface.



A DSP's information can be used by a computer to control such things as security, telephone, home theater systems, and video compression. Signals may be compressed so that they can be transmitted quickly and more efficiently from one place to another (e.g. teleconferencing can transmit speech and video via telephone lines). Signals may also be enhanced or manipulated to improve their quality or provide information that is not sensed by humans (e.g. echo cancellation for cell phones or computer-enhanced medical images). Although real-world signals can be processed in their analog form, processing signals digitally provides the advantages of high speed and accuracy.

Because it's programmable, a DSP can be used in a wide variety of applications. You can create your own software or use software provided by ADI and its third parties to design a DSP solution for an application.

For more detailed information about the advantages of using DSPs to process real-world signals, please read Part 1 of the article from Analog Dialogue titled: *Why Use DSP? Digital Signal Processing 101- An Introductory Course in DSP System Design*.

back to top

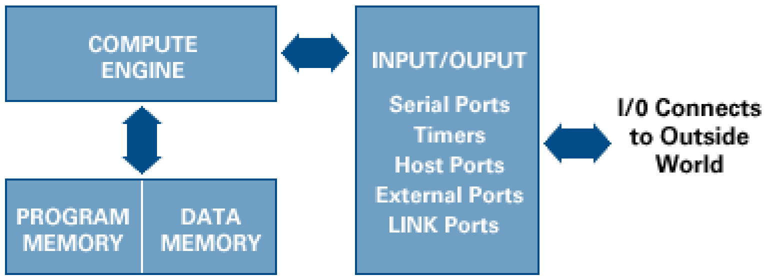
---

## What's Inside a DSP?

A DSP contains these key components:

- **Program Memory:** Stores the programs the DSP will use to process data
- **Data Memory:** Stores the information to be processed
- **Compute Engine:** Performs the math processing, accessing the program from the Program Memory and the data from the Data

Memory
- **Input/Output:** Serves a range of functions to connect to the outside world



back to top

---

**Recommended Reading**
Digital Signal Processing is a complex subject that can overwhelm even the most experienced DSP professionals. Although we have provided a general overview, Analog Devices offers the following resources that contain more extensive information about Digital Signal Processing:

- The Scientist and Engineer's Guide to Digital Signal Processing
- Analog Dialogue Series: Digital Signal Processing 101- An Introductory Course in DSP System Design
  - Part 1: Why use DSP? DSP Architecture and DSP Advantages Over Traditional Analog Circuitry
  - Part 2: Learn More About Digital Filters
  - Part 3: Implement Algorithms on a Hardware Platform
  - Part 4: Programming Considerations for Real-Time I/O
- Let's Talk DSP: Commonly Use Worlds and What They Mean

DSP workshops are a very fast and efficient way to learn how to use Analog Devices DSP chips. The workshops are designed to develop a strong working knowledge of Analog Devices' DSPs through lecture and hands-on exercises. For schedule and registration information, visit the DSP Workshops page.

# How to find signals in noise using estimation

By Trace Baker

**Embedded Systems Programming**
(09/04/03, 17:00:00 PM EDT)

PRINT THIS STORY SEND AS EMAIL

**One of the most challenging aspects of digital signal processing is finding signals in noise when familiar tools such as averaging and low-pass filtering don't work. Maximum-likelihood estimation is another technique to extract information from a sea of noise.**

Not only is unwanted noise is the bane of modern living, it can also be a stumbling block for the embedded systems engineer working with digital signal processing. Figure 1 shows the graph of a signal that's almost lost in a sea of noise. This article will show you how to write code to estimate the peak amplitude of a signal despite the noise. This technique is especially useful in analyzing rare events in which averaging isn't an option but you still have to determine if a signal's amplitude is above or below an alarm threshold. Although you could use a low-pass filter, you'd then have two computational steps (filtering followed by a search for the peak) to prepare the signal before you can measure it.

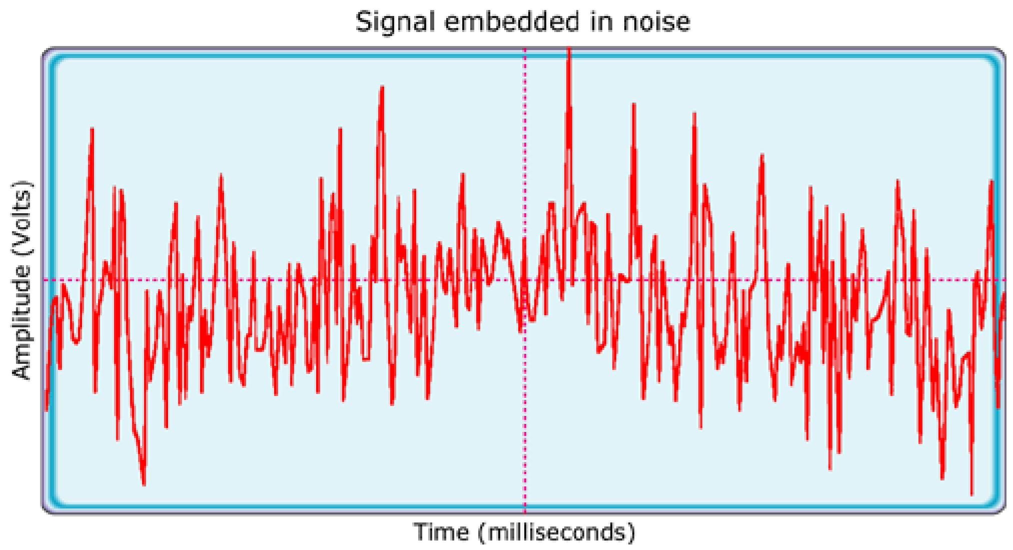


**Figure 1: Example signal in Gaussian noise**

The technique we'll use to estimate the parameters of a signal embedded in noise belongs to a class of statistical methods called *maximum-likelihood estimators* (MLEs for short, and also called *optimal estimators*). MLEs are computationally cheap, intuitive to understand, and straightforward to program.

By the way, in case you're wondering, the noise in Figure 1 hides a unit-amplitude, half-cycle sine wave; the signal-to-noise ratio (SNR) is "2.45dB. Figure 1's peak amplitude, estimated using the techniques presented in this article, is 0.986 with an error of 1.4%.

## Mathematics

The mathematical formulation of the MLE for the received signal in Figure 1 is:

$$\hat{a} = \frac{1}{u}\int_0^T \sin(t) r(t) dt \quad (1)$$

where *â* is the estimate of the amplitude of the transmitted noise-free signal, constant *u* is the time integral of the unit-amplitude signal in the absence of noise, and *r* describes the received noisy signal as a function of time. In this example, *u* is determined by computing:

$$u = \int_0^T \sin^2(t) dt \quad (2)$$

The integrals in Equations 1 and 2 are evaluated over the time interval [0, *T*], which is the duration of the signal. In this case, *T* is the duration of the half-period of the sine wave. The sine function takes a single argument that is an angle in radians, so the time interval maps to an angle by the simple relationship π/*T*, expressed in units of radians per second.

Of course, not every noisy signal you encounter will be a half-cycle sine wave. For these other situations, the general forms for Equations 1 and 2 are:

$$\hat{a} = \frac{1}{u}\int_0^T s(t) r(t) dt \quad (3)$$

and:

$$u = \int_0^T s(t)^2 dt \quad (4)$$

where function *s* is the unit-amplitude transmitted signal, and *r* is the received signal as defined for Equation 1. In practice, *u* can be determined during a calibration run with a signal of known amplitude. Applying *u* to signals of unknown received amplitude in subsequent runs gives the estimated amplitude of the received signal as a fraction of the calibration standard. The derivation of Equations 3 and 4 is well beyond the scope of this article; you can explore it further by consulting the references at the end of the article.

If you already have a collection of algorithms for approximating definite integrals, you should now have enough information to start coding an MLE. But if you read on, you'll learn some tricks for implementing MLEs efficiently and you'll better understand the trade-offs to consider when deciding if this technique is appropriate for a particular project.

## Implementation

Because this article doesn't focus on numerical-integration methods, the examples that follow use the extended form of Simpson's rule to approximate integrals, a technique that's widely used, accurate enough for many applications, easy to understand, and straightforward to program. The numerical approximation of the integration in Equation 3 by extended Simpson's rule looks like this:

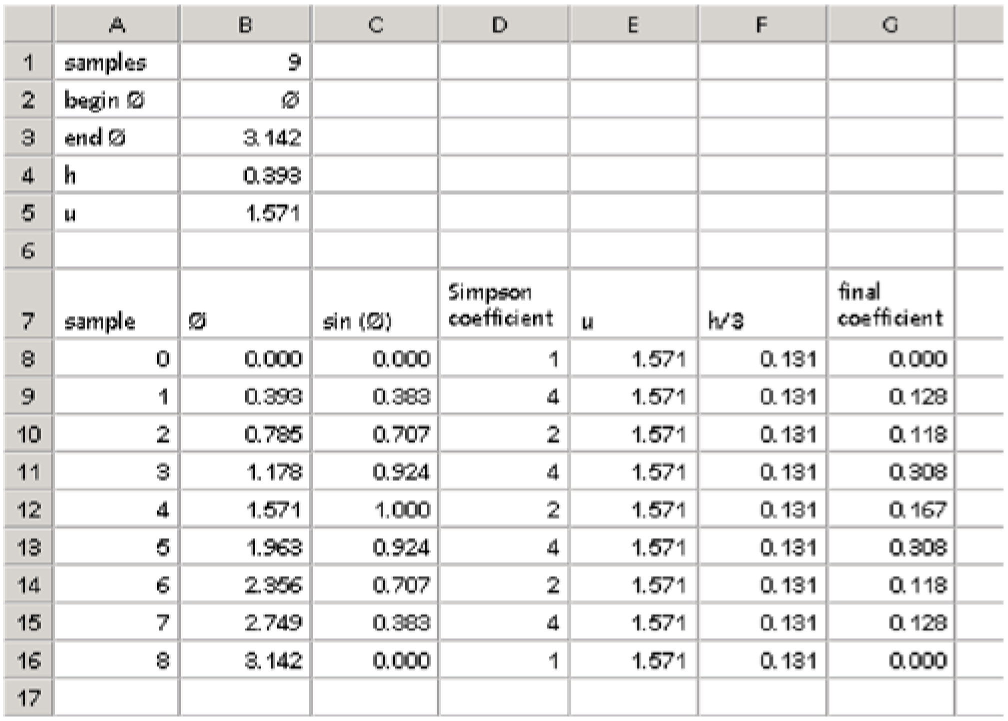
$$\frac{1}{u}\int_0^T s(t)r(t)dt \approx c_0r_0+4c_1r_1+2c_2r_2+4c_3r_3+\ldots+c_nr_n$$

where *h* is the equally-spaced sampling interval computed for *n* samples as:

$$h=\frac{t_n-t_0}{n}$$

and:

$$c_i=\frac{h}{3u}s(ih),\ i=0,1,\ldots,n$$



|  | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | samples | 9 |  |  |  |  |  |
| 2 | begin Ø | Ø |  |  |  |  |  |
| 3 | end Ø | 3.142 |  |  |  |  |  |
| 4 | h | 0.393 |  |  |  |  |  |
| 5 | u | 1.571 |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |
| 7 | sample | Ø | sin (Ø) | Simpson coefficient | u | h/3 | final coefficient |
| 8 | 0 | 0.000 | 0.000 | 1 | 1.571 | 0.131 | 0.000 |
| 9 | 1 | 0.393 | 0.383 | 4 | 1.571 | 0.131 | 0.128 |
| 10 | 2 | 0.785 | 0.707 | 2 | 1.571 | 0.131 | 0.118 |
| 11 | 3 | 1.178 | 0.924 | 4 | 1.571 | 0.131 | 0.308 |
| 12 | 4 | 1.571 | 1.000 | 2 | 1.571 | 0.131 | 0.167 |
| 13 | 5 | 1.963 | 0.924 | 4 | 1.571 | 0.131 | 0.308 |
| 14 | 6 | 2.356 | 0.707 | 2 | 1.571 | 0.131 | 0.118 |
| 15 | 7 | 2.749 | 0.383 | 4 | 1.571 | 0.131 | 0.128 |
| 16 | 8 | 3.142 | 0.000 | 1 | 1.571 | 0.131 | 0.000 |
| 17 |  |  |  |  |  |  |  |

**Figure 2: Using a spreadsheet to calculate MLE coefficients**

In many implementations, it's possible to use the distributed property of multiplication and your knowledge of *s*(*t*) to precompute the coefficients and store them in a table rather than compute each term at each signal acquisition. You can do this part in a spreadsheet, as shown in Figure 2 for a half-period sinusoidal signal taken as nine samples equally spaced in time. Using the precomputed coefficients from Figure 2, an entire MLE can be expressed as a fragment of C code as shown in Listing 1.

**Listing 1: A maximum-likelihood estimator in C**

```
double coeff[] = {0.00, 0.128, 0.118, 0.308, 0.167, 0.308, 0.118, 0.128, 0.000;};
double estimate = 0.0;
int samples = (sizeof(coeff) / (sizeof(double));

for (int i = 0; i < samples; i++)
{
 estimate += GetSample() * coeff[i];
}
```

The code in Listing 1 shows that the MLE method is efficient. But you can do much more than just tabularize the coefficients needed to approximate the integral. Any multiplicative constant can be folded in. For example, if a calibration is associated with each sample, as would be the case if the received signal were extracted from a linear charge-coupled device (CCD) array rather than from a single detector, the gain of each CCD element can be applied to make the final line of Listing 1 become:

```
estimate += GetSample() *
   coeff[i] * gain[i];
```

Note that if extreme accuracy is important or you're doing scaled-integer math with a limited number of bits, you need to take time to evaluate how error propagates through the entire chain of multiplications and the final summation.

To determine whether an MLE is the appropriate solution for your problem, you have to remember that you're dealing with a signal in random noise. Therefore, you can't determine the absolute worst-case performance analytically. Instead, you have to be content to work with the mean and variance of the estimate. With enough math, you can show that the MLE is a good estimator—the variance of the estimate approaches zero as the number of samples approaches infinity.

If you need to understand the behavior of an MLE to this level, analytical methods for determining average performance as a function of the amount of noise contaminating the signal and the number of samples are available in the references at the end of this article.

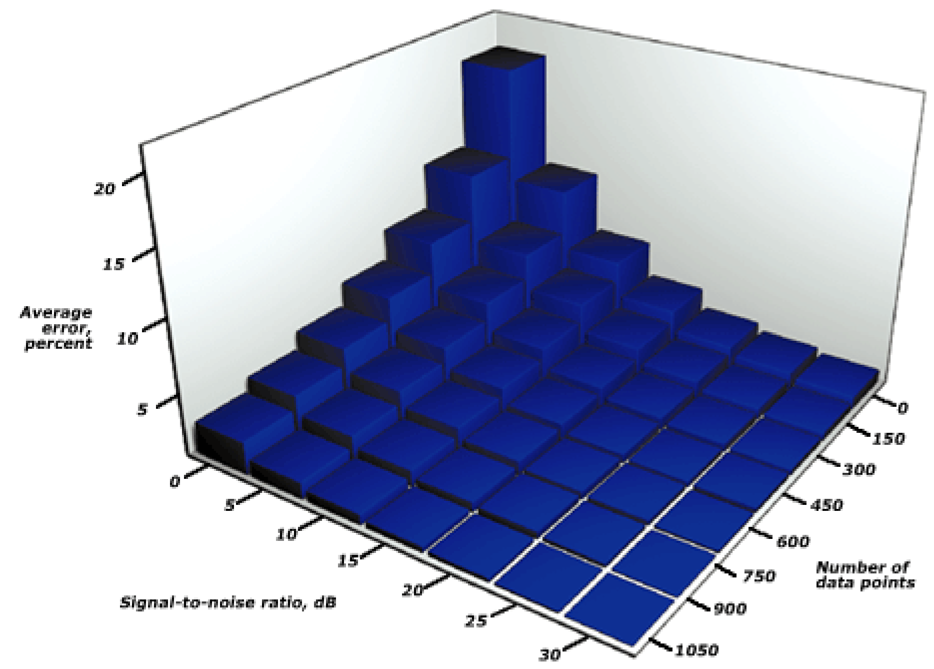


**Figure 3: Average error estimating the peak of the received signal as a function of SNR and number of data points**

For most applications, though, you can approach questions of accuracy with rules of thumb:

- The more noise you have, the more samples you need to take to get rid of it—no surprise there. The average error of the estimate decreases as a power of the number of samples and also decreases with increasing signal-to-noise ratio. Figure 3 illustrates these relationships with data from a simulation experiment for our half-cycle sine wave example. The variance of the estimate exhibits similar properties.
- The algorithm you choose to approximate the integral also influences accuracy. For extended Simpson's rule, error decreases approximately with the fourth power of the number of points.

## Design

Regardless of how much accuracy you need for your application, the accuracy you get from any practical estimation system will be limited by the number of samples you process. Since system resources limit the number of samples the system can process, tradeoffs are necessary. The trade space shown graphically in Figure 3 is summarized as the relationships among design goals for SNR, sampling rate, and accuracy in Table 1.

**Table 1: Tradeoffs for maximum-likelihood estimators**

| **Design goal** | **Effects on design** | **Figures of merit** |
|---|---|---|
| Increase estimation accuracy | Increase sampling rate | Average error, variance of error |

| | | |
|---|---|---|
| Handle additional noise | Increase sampling rate | Signal-to-noise ratio |
| Decrease processor utilization | Decrease sampling rate | Processor speed, memory |

By now, you've seen that maximum-likelihood estimators have a number of characteristics that make them desirable for use in embedded systems:

- MLEs use all the information available in the received signal.
- Each data point acquired only needs one multiplication and one addition
- You have your result as soon as you process the last sample
- MLEs have advantages over time averaging and finite-impulse-response filters in not requiring storage for bins of samples or individual samples.

Even with these benefits, though, MLEs are not a one-size-fits-all solution; any estimation technique has its limitations. For example, the MLE described in this article requires advance knowledge of the shape and the phase of the transmitted signal. The signal also has to be a single-valued function that is differentiable over the domain 0 t *T*. You also have to be able to assume the noise is additive, zero-mean, and Gaussian with frequency content higher than that of the signal you're trying to estimate. Finally, you'll need memory to store the coefficients if you can't or don't want to compute them on the fly. Fortunately, many real-world applications fit these limitations.

Here's a summary of the design process for an MLE:

1. Determine the accuracy needed for the quantity you're estimating. This isn't the required accuracy of the entire system, only that part allocated to the estimation method to be implemented in software. Other subsystems such as optics and electronics take their share; as usual, software has to work with what is left over.
2. Determine the amount and type of noise in the input signal.
3. Determine the amount of processing time and memory available for use by the MLE. Unless you're using a processor dedicated to performing the estimation, you'll have to work with some fraction of the total processor bandwidth.
4. Pick the number of samples needed to give the required accuracy, considering the amount of noise expected.
5. Choose an algorithm to approximate the integral. This decision also has an influence on accuracy, so you may need to do Step 4 again.
6. Decide if you'll use precomputed tables of coefficients or will compute coefficients in real time with each sample. If you choose the table form, determine the number of significant digits you need to store for each coefficient to meet your accuracy requirement.
7. Estimate processor time and memory consumed from your decision in Step 6.
8. If you exceed the resource budget determined in Step 3, you'll have to go back to Step 1.

If you get to Step 8 without an implementable or practical design, you'll have to renegotiate with other system stakeholders, asking for a faster processor, more memory, or a relaxation of the accuracy requirement.

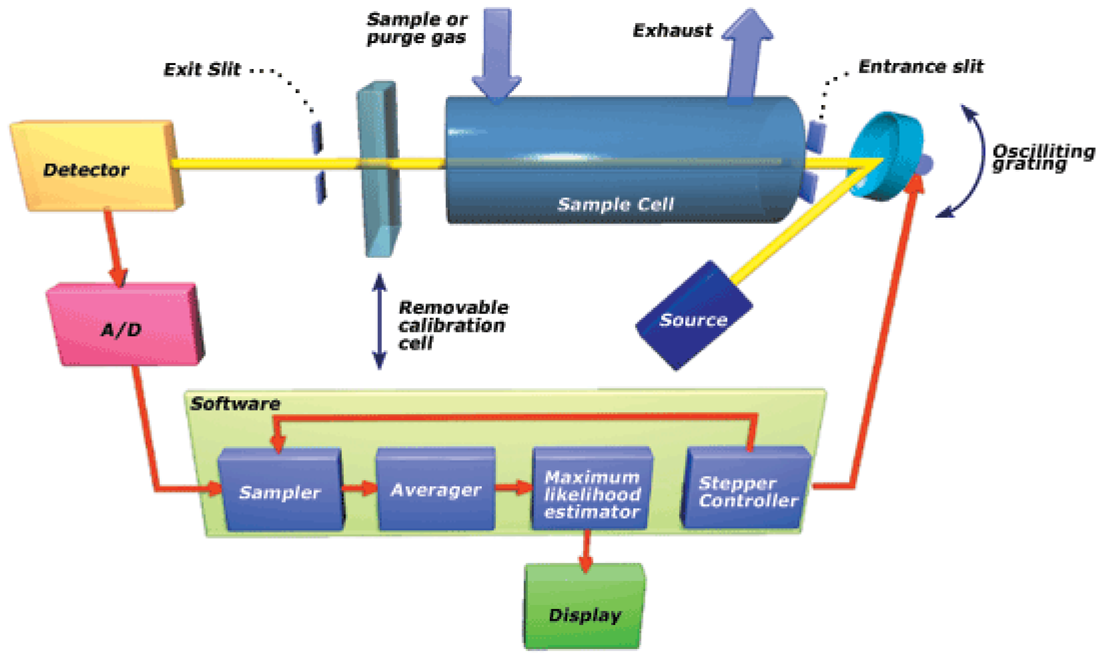


**Figure 4: Example of air-pollution monitor**

## Applications

Figure 4 shows a real-world example of how a maximum-likelihood estimator was used to estimate the concentration of an atmospheric pollutant to the level of a few parts per billion.

The optical elements of the system included a light source producing a collimated beam at the wavelength of an absorption peak of the pollutant, a diffraction grating driven by a software-controlled stepper that produced a transmitted signal modulating wavelength in time, a cell with transparent end windows to hold a sample of gas, a calibration cell containing a known concentration of pollutant that could be positioned in the optical path, and a photodetector. The light beam passing through the sample cell was attenuated in proportion to the concentration of the pollutant in the sample cell.

The signal-processing elements of the system included software that controlled the stepper motor that moved the diffraction grating, an equal-interval sampler, an averager, and an MLE. Constant $u$ (Equation 2) was determined by purging the sample cell with gas free of the pollutant, inserting the calibration cell, and performing a measurement. Calibration was checked by inserting the calibration cell and repeating the measurement.

Software controlled both the position of the grating and the timing of sampling, enabling the wavelength of each sample to be known. Averaging over long intervals (tens of seconds) removed much of the noise. The MLE estimated the amplitude of the signal without having to explicitly search for the peak in residual noise. This combination of averaging and an MLE was dictated by limited processing resources, which in turn limited the number of samples that could be taken in each half-cycle of the signal. The integration was performed by an eight-term Newton-Cotes approximation using coefficients derived from Lagrange interpolation polynomials developed specifically for the input waveform.

## Multiple choice

You can develop maximum-likelihood estimators to estimate other signal parameters, including phase, arrival time, and frequency. Simultaneous estimation of multiple unknown parameters is possible. Other techniques can also help you estimate parameters other than zero-sum, white Gaussian noise, and for samples that aren't equally spaced in time. You'll find these techniques in McDonough and Whalen's book *Detection of Signals in Noise*.

**Trace Baker** has developed embedded software for measurement and control since the days when a 4KB memory card required three power supplies and cost about $2,000. He specializes in mission-critical systems for regulated markets, and currently works in the aerospace industry. Contact him at trace@treeline.com.

## Further reading

Abramowitz, M. and I. A. Stegun, Handbook of Mathematical Functions, Washington, DC: U.S. Government Printing Office, 1964.

McDonough, R. N. and A. D. Whalen, *Detection of Signals in Noise* (2nd edition), San Diego: Academic Press, 1995.

Schaeffer, R. L. and J. T. McClave, *Probability and Statistics for Engineers*, Belmont, CA: Duxbury, 1995.

NDT.net July 2003, Vol. 8 No.07

# Noisy signal processing in real time DSP systems

E. Kazanavicius, A. Mikuckas, I. Mikuckiene, V.Kazanavicius
Digital Signal Processing Laboratory, Computer Department, Kaunas University of Technology
Studentu 50-214c, 3031 Kaunas, LITHUANIA, E-mail: ekaza @dsplab.ktu.lt

Published in Ultragarsas Journal 2003 Vol 46 No1

## Introduction

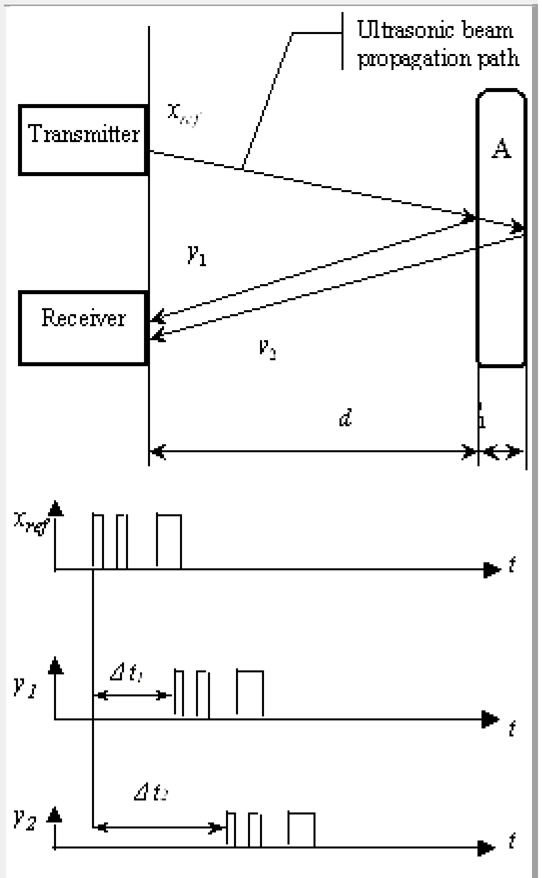


Fig 1: Ultrasonic NDT system.

Modern ultrasonic and radar measurement systems are widely used in the field of non-destructive testing for a long time. The limitation of a currently available ultrasonic instruments hardly lies on the property of hardware but it may lie on the lack of sufficient signal processing techniques [1]. At present, the ultrasonic A-scan type instruments are most commonly used. It is believed that the received A-scan signal may carry a lot of information on material properties and defects, but information appears in various guises of noise, which is to be deciphered completely. A lot of research has been done on ultrasonic signal processing and still now is going on in search for more reliable and versatile signal processing techniques [2-6]. Generally, the flaw signals measured in ultrasonic NDT include the effects of the measurement system and are corrupted by different kind of noise. The highly complex interaction between the defect geometry and the back-scattered ultrasonic wave inside the test piece may not be assumed as a linear process. So, the signal processing techniques which require apriory knowledge of noise statistics, are subject to fail in many situations. Therefore the approach of signal processing should be involving the noisy signal itself in constructing the signal processing method.

## Signal processing in ultrasonic NDT systems

Let us analyze a real time system in which a transmitter and a receiver are located at predetermined points. Several M-sequences (usually the Barker code), modulated by ultrasonic wave are used as the transmitted signal and receiver receives signals reflected from the target. Such a system is depicted in Fig. 1.

This system measures the thickness of moving object **A**. The reference signal $x_{ref}$ consisting of a certain coded sequence is emitted by the ultrasonic transmitter at the time moment $t_0$. The propagating signal partially reflects from the front side of object **A**:

$$y_1(t) = k_1\, x_{ref}(t - \Delta t_1) \tag{1}$$

The other part of the emitted reference signal $x_{ref}$ reflects from the rear side of object **A**:

$$y_2(t) = k_1\, x_{ref}(t - \Delta t_2) \tag{2}$$

where: $k_1$ and $k_2$ are the coefficients depending on a distance to the object, environment and object properties, $\Delta t_1$ and $\Delta t_2$ are the delay times directly proportional to the distance $d$ and the thickness of object **A**:

$$\Delta t_1 = d \cdot c_{env} \tag{3}$$

$$\Delta t_2 = d \cdot c_{env} + d_1 \cdot c_{obj} \tag{4}$$

Finally in the receiver we get the signal *y*:

$$y(t) = k_1 x_{ref}(t - \Delta t_1) + k_2 x_{ref}(t - \Delta t_2) \tag{5}$$

The task of signal processing is to determine the time instances $\Delta t_1$ and $\Delta t_2$. Then values of control signals are calculated and transmitted to actuators. The signal processing time is restricted by properties of a technological process and the velocity of the object **A**. For determination of the time instances $\Delta t_1$ and $\Delta t_2$ usually it is used principle of obtaining the impulse response by a correlation process. Let's consider $x_{ref}$ to be the transmitted sequence, *y*(t) to be the received sequence and *h*(t) to be the impulse response of the composite system, which includes the test piece, the transceiver system and their associated electronics. The scattered ultrasound that is picked up by the receiver and the additive system noise *n*(t) constitute the received signal *y*(t). The signal of the output of the correlation filter can be represented by

$$\mu(\tau) = \int x(t+\tau)\left[x(t) * h(t) + n(t)\right]dt \tag{6}$$

If peaks corresponding to reflections from the targets were clearly identified in the cross correlation function (CCF), it would be easy to determine the time instances $\Delta t_1$ and $\Delta t_2$. In practice, however, it quite difficult to identify them because of suspicious peaks in the CCF due to a noise from the surrounding medium and it is essential to cancel out effects of noise. In order to reduce the effects of a noise during transmission and reception some measures have to be taken.

## Band modification by moving average

Data processing is performed on both of the sampled received signals and the original M-sequence and the sampling frequency is such that there are *j* samples per unit pulse of the M-sequence. So, the expected peak on the CCF is supposed to consist of 2*j* samples. In order to minimize spurious peaks, widths of which are less than 2*j* samples, the sampled data are smoothed by performing the moving average of the data sequence:

$$y[n] = c_1 y[n-1] + c_2 y[n] + c_3 y[n+1] \tag{7}$$

$$c_1 + c_2 + c_3 = 1 \tag{8}$$

This is equivalent to application of the Hanning spectral window. This method considerably minimizes the noise peaks of comparatively small width while keeping the expected peaks intact.

## Averaging

The system emits the reference signal $x_{ref}$ periodically

$$x_{ref}[n] = x_{ref}[n + lN] \tag{9}$$

where *N* is the period of the reference signal. If the position of the object **A** during *l* periods changes a little, it is possible to average input signal *l* times:

$$y[n] = \frac{1}{l} \sum_{l=0}^{l-1} y[n - lN] \tag{10}$$

The noise level is reduced $\sqrt{l}$ times. This is effective, but time wasting method and is not used in the case of signal processing time restrictions in real time systems. This method does not allow eliminating peaks caused by surrounding environment.

## Noise cancellation by subtraction

Some coherent peaks, additive to the expected peaks, appear on the CCF, which are confusing in regard to the clear distinction of a target. This is due to the surrounding structure or due to the effect of limitations of the measuring system. These clutter peaks appear irrespective of presence of any target. To perform the subtraction, first of all the data are collected from the test object without presence of any the target. Another data are taken with the presence of target. Coherent peaks are cancelled by taking the difference between the CCF of second data and that of the first data. This helps distinguishing the peaks corresponding to the reflections from the newly developed targets by removing the coherent noise of the system.

## Inverse filtering

Passing the signal through an inverse filter can significantly reduce a random and clutter type noise. A major part of the long CCF is to be assumed as a noise except the portion corresponding to the direct signal and the signal reflected from the target. The inverse filtering operation [7-9] of a signal is described as:

$$\hat{y}[n] = y[n] - \sum_{k=1}^{P} a_k y[n-k] \tag{11}$$

where $\hat{y}[n]$ is the output of the filter and *P* is prediction or the model order. The inverse filter is designed calculating the coefficients $\{a_k\}$ based on noisy data. Coefficients $\{a_k\}$ are obtained by solving the equation [10]:

$$\sum_{k=1}^{P} a_k R[i-k] = -R[i] \tag{12}$$

where *R*[i] is the autocorrelation function defined by:

$$R[i] = \sum_{k=1}^{N-1} y[k]y[k+i] \tag{13}$$

This autocorrelation function is constructed with *N* samples of data from a suitable portion of the received signal, which presumably contains no expected peak. Such a filter is depicted in Fig. 2

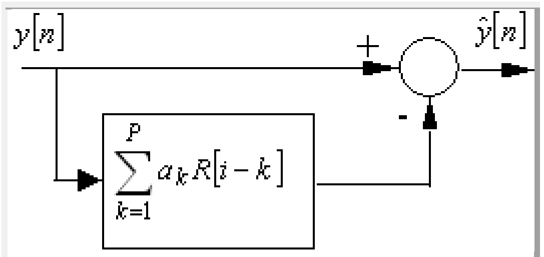


Fig 2: Inverse filter.

This filter attempts to remove the predictable part of the signal and produce an output $\hat{y}[n]$, which is completely unpredictable to the filter.

## Wavelet transform based noise reduction

During the last time the wavelets have become a popular de-noising (or noise reduction) tool [11]. Donoho and Johnston [12] showed that this method has statistical optimality properties. Many algorithms define a criterion to divide wavelet transform coefficients into two groups. The first group contains the coefficients dominated by a noise, while other coefficients are rather clean. These algorithms eliminate all wavelet coefficients below a certain threshold, because these coefficients are dominated by a noise.

Let's consider the following model of the received discrete noisy signal

$$z[n] = y[n] + \varepsilon[n]\,,\ i = 1,\ldots,N \tag{14}$$

or in a vector notation:

$$\mathbf{z} = \mathbf{y} + \boldsymbol{\varepsilon} \tag{15}$$

To reconstruct the original data, a wavelet representation is used. We use simple non-redundant orthogonal, discrete wavelet transforms. An orthogonal matrix *W* can be used to represent this operation. We consider the following transform:

$$\begin{aligned} \mathbf{v} &= W \cdot \mathbf{y}\,, \\ \boldsymbol{\omega} &= W \cdot \boldsymbol{\varepsilon}\,, \\ \mathbf{w} &= W \cdot \mathbf{z} = \mathbf{v} + \boldsymbol{\omega} \end{aligned} \tag{16}$$

These transforms localize the most important spatial and frequencies characteristics of a regular signal in a limited number of wavelet coefficients. On the other hand, it is easy to prove that an orthogonal transform of a stationary, white noise results in a stationary white noise. This means that the expected noise energy is the same in all coefficients. If this energy is not to large, the noise has a relatively small influence on the important large regular signal coefficients. These observations suggest replacing the small coefficients by zero, because they are dominated by noise and carry only a small amount of information.

The thresholding operations can be represented as

$$w_{\delta i} = D_\delta \cdot w_i \tag{17}$$

where

$$D_\delta = diag\left[d_{ii}\right] \tag{18}$$

There are known two threshold methods – hard threshold and soft threshold (or shrinking function) [13-15].

In the case of the hard threshold the entries of the matrix $D_\delta$ are

$$d_{ii} = \begin{cases} 0, \text{ if } |w_i < \delta|, \\ 1, \text{ otherwise.} \end{cases} \tag{19}$$

In the case of soft threshold the entries of the matrix $D_\delta$ are

$$d_{ii} = \begin{cases} 0, & \text{if } |w_i < \delta|, \\ 1 - \dfrac{\delta}{|w_i|}, & \text{otherwise} \end{cases} \tag{20}$$

These threshold functions are shown in Fig. 3. A wavelet coefficient $\omega$ between $-\delta$ and $\delta$ is set to zero, while others have the same value in the case of the hard threshold, or are shrunk in an absolute value in the case of the soft threshold.

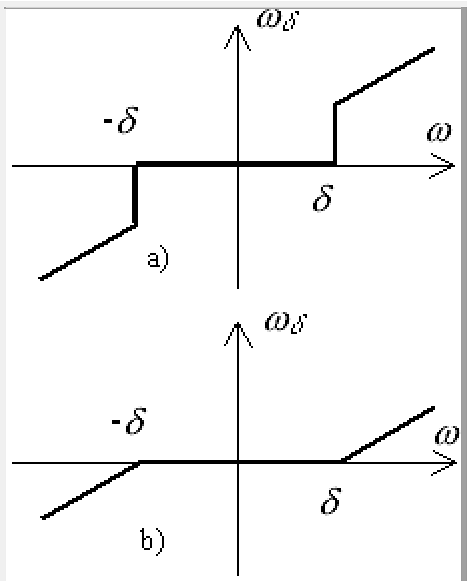


Fig 3: Hard thresholding (a) and soft thresholding (b) functions.

A natural question arising from this procedure is how to chose the threshold. If $\mathbf{y}_\delta$ is the result of applying threshold procedure to the wavelet coefficients of signal y, and $\varepsilon_\delta$ = $\mathbf{y}_\delta$-$f$ is the noise of this result, then an often used criterion to measure the quality of this result is its signal to noise ratio ($SNR(\delta)$):

$$SNR(\delta) = 10 \log_{10} \frac{\sum_{i=0}^{n-1} f_i^2}{\sum_{i=0}^{n-1} \varepsilon_{\delta i}^2} \tag{21}$$

An optimal choice of $\delta$ should maximize $SNR(\delta)$. This is equivalent to minimizing the mean squared error $R$:

$$R(\delta) = \frac{\sum (y_{\delta i} - f_i)^2}{n} = \frac{\|\varepsilon_\delta\|^2}{n} \tag{22}$$

Because the wavelet transform is orthogonal, we can also compute $R$ from the wavelet coefficients as:

$$R(\delta) = \frac{\|\omega_\delta\|^2}{n} \tag{23}$$

$\omega_\delta = W\varepsilon_\delta$ is the noise after operation in the wavelet domain.

However, because $\boldsymbol{f}$ is unknown, the function $R_\delta$ is not computable and hence it cannot be used to find optimal $\delta$. For automatic spline smoothing it was suggested to adapt general cross validation (GCV). Our aim is to minimize error function based on an unknown exact signal. We therefore try to find a good compromise between a goodness of fit and smoothness. We assume that the original signal is regular to some extend, which means that the value $f_i$ can be approximated by an linear GCV combination of its neighbors. So, by considering $y_{\delta i}$ a combination of $f_i$, not depending on $f_i$ itself, we can eliminate the noise in this particular component. Since we replace it by a weighted average of its neighbors, the noise in these components is smoothed, and so we end up with a relatively clean, noise-independent value. Applied to the wavelet procedure this GCV should be a function of a threshold value using only known data and having approximately the same minimum as the residual function $R(\delta)$.

We have a definition of general cross validation:

$$GCV(\delta) = \frac{\frac{1}{N}\left\|\omega_\delta - \omega\right\|^2}{\left(\frac{N - Tr\left(D'\right)}{N}\right)^2} \tag{24}$$

where $d'_{ij} = \frac{\partial \omega_{\delta i}}{\partial \omega_j}$ .

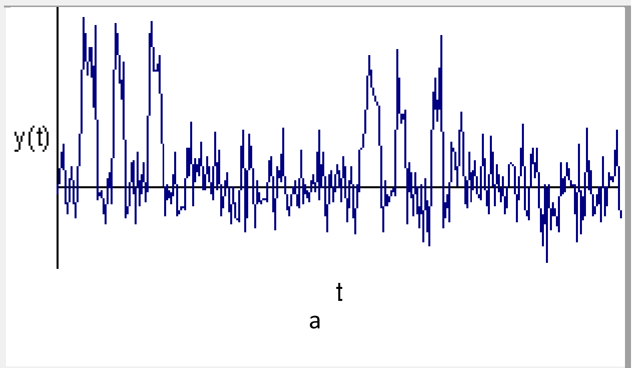
Note that if $i \neq j$, then $d_{ij}=0$. For $i=j$ we have

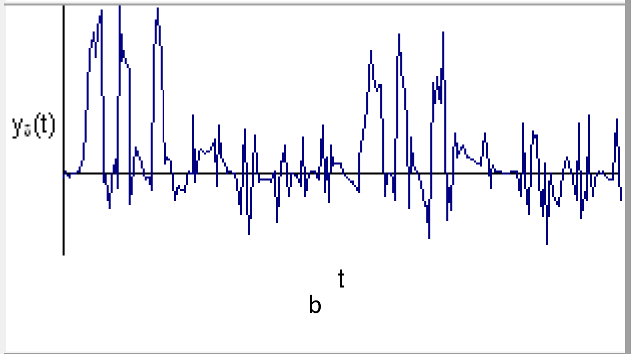
$$d'_{ii} = \begin{cases} 0, & \left|\omega_i\right| < \delta, \\ 1, & \left|\omega_i\right| \geq \delta. \end{cases} \tag{25}$$

Thus, if $Tr(D')$ is the trace of $D'$,

$$Tr\left(D'\right) = \#\left\{i \middle| \omega_{\delta i} \neq 0\right\}. \tag{26}$$

The results of applying the threshold procedure on the reflected signal are depicted in Fig. 4. In this case only a fragment of the Barker code is used for formatting M-sequence.





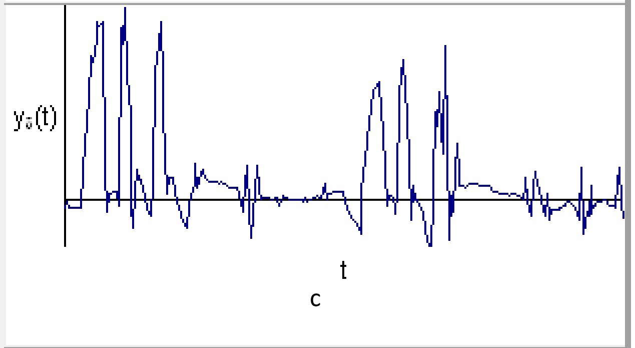


Fig 4: The reflected signal: a - without applying de-noising procedure, b - after applying hard thresold procedure, c - after applying soft threshold procedure.

## Wavelet transform based signal processing method for ultrasonic NDT system

In order to reduce computations, the reference signal $x_{ref}$ and the reflected signal $y_{\delta}$ (signal $y$ after applying thresholding procedure) are transformed:

$$\tilde{x}_{ref} = \frac{1+\operatorname{sgn}(x_{ref} - \delta_1)}{2}$$
$$\tilde{y} = \frac{1+\operatorname{sgn}(y_\delta - \delta_1)}{2}. \quad (27)$$

After applying this transform we get digital signals $\tilde{x}_{ref}$ and $\tilde{y}$ with logical values "0" and "1". This transform is possible because the most important information is the pulse widths of M-sequence. Results of applying procedure (27) are shown in Fig. 5

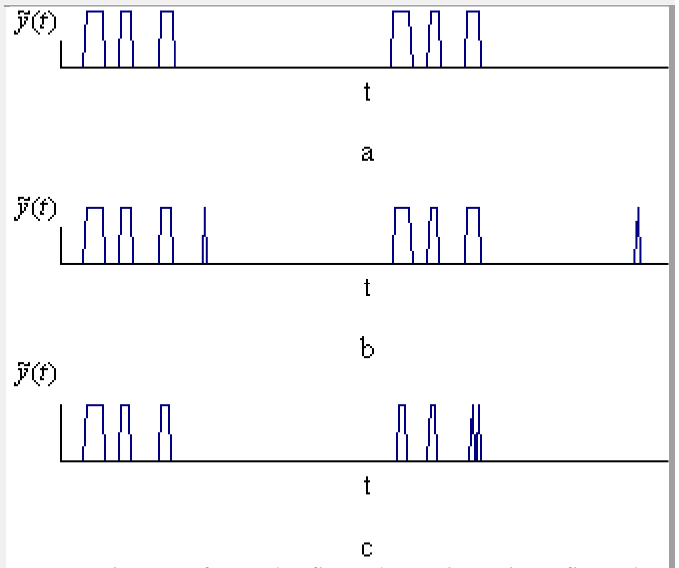


Fig 5: The transformed reflected signal: a - the reflected signal without noise, b - the reflected noisy signal after applying hard threshold procedure, c - the reflected noisy signal after applying sof threshold procedure.

The soft threshold procedure allows achieving a better visual quality, than the hard threshold procedure as shown in Fig.4. But, when the noise level is high, applying the soft thresholding procedure more distorts pulse widths of the transformed signal $\tilde{y}$ as shown in Fig.5.

The results of these transforms depend on the threshold $\delta_1$ value, which optimal value varies in accordance to the noise level $\varepsilon$. When the threshold is low, an additional pulses emerge in the transformed signal $\tilde{y}$, as shown in Fig. 5b. When the threshold value is high, the pulse widths of the transformed signal $\tilde{y}$ are shrinking and pulses may be distorted as shown in Fig. 5c. The optimal threshold $\delta_1$ value was defined by computing maximum value of the correlation function:

$$\rho[n] = \frac{1}{M}\sum_{i=0}^{M-1}(\tilde{x}_{ref}[n] \cdot y[n+i] - \tilde{x}_{ref}[n] \oplus y[n+i]) \qquad (28)$$

at different threshold $\delta_1$ values.

The best results were achieved when $\delta_1$=1.4$\delta$, where $\delta$ is the threshold value defined by computing GCV function. Note that GCV function computation in this case does not require any floating-point operation and may be computed by a hardware. These computations may be simplified in the case when a fixed M-sequence is used.

Such a procedure may be used for recovering a distance to the object from noisy data. It possesses three steps:

1. apply the interval adapted pyramidal algorithm of Cohen, Daubechies, Jawerth and Vial [11] to the measured data, obtaining empirical wavelet coefficients $\omega_i$
2. apply the soft threshold nonlinearity $w_{\delta i} = \operatorname{sgn} w_i (w_i - \delta)_+$coordinatwise to the empirical wavelet coefficients with the specially chosen threshold $\delta$;
3. invert the pyramid filtering recovering $y_\delta(t_i),\ i = 0,1,\ldots,n-1$;
4. apply the transform (xx) to the reference signal $x_{ref}$ and de-noised data $y_\delta$;
5. detect the argument of correlation function maximum value.

For a fast wavelet transform we need 2*N*2*F* flops, where *F* is the number of filter coefficients. For *F*=4 , we have 16*N* flops. To reconstruct the signal after operation with the optimal threshold $\delta$ we need 16*N* flops.

Computation of *GCV*($\delta$) can be performed completely in the wavelet domain. Because *GCV*($\delta$) is an approximation itself it is not useful to compute its minimum very precisely. Moreover, in most cases this is not necessary to the curve of *R*($\delta$) in the neighborhood of its minimum. A relative accuracy of $10^{-3}$ is enough. Using a classic minimization procedure (such as Fibonacci) this requires approximately 15 function evaluations. The denominator $N$-$Tr(D')$counts the number of coefficients that are set to zero. This does not require any floating-point operation. Computation of the nominator can be done with 2$N$ floating point operations. So 15 function evaluations lead to some 30$N$ floating-point operations.

Computation of the signal $\tilde{y}$ can be done with $N$ floating point operations.

Computation of the correlation function does not require any floating-point operation.

So execution of the suggested signal processing algorithm leads to 63$N$ operations. Execution of a classical signal processing algorithm leads to ($L$+2$P$)$N$ operations, where $N$ is the number of samples, $L$ is the length of M-sequence, $P$ is the model order of the inverse filter. The suggested algorithm requires less floating-point operations, when
($L$+2$P$)>63.

## Conclusions

Generally the flaw signals measured in ultrasonic NDT systems are spoiled by different kind of a noise. Therefore, the approach of signal processing should be involving the noisy signal itself in constructing the signal processing method. The noise in such systems is cancelled by band modification using moving average, signal averaging, inverse filtering and noise cancellation by subtraction. These methods are time consuming and due to signal processing time restrictions not always may be used in real time systems. During the last time the wavelets have become a popular de-noising (or noise reduction) tool and this method has statistical optimality properties. New data processing method based on the wavelet transform for real time systems is suggested. It is shown that the hard threshold algorithm is preferred to the soft threshold in such systems. Execution of this method leads to less amount of floating point operations than classical signal processing methods.

## References

1. **Sinclair A**. An analysis of ultrasonic frequency response for flaw detection: a technique review. Materials evaluation. 1989. Vol. 43. P. 870-883.
2. **Fomitchev M. I. et. al.** Ultrasonic pulse shaping with optimal lag filters. Int. J. Imaging syst. technol. (USA). 1999. Vol. 10. P. 397-403.
3. **Sallard J. et. al**. Use of a priori information for the deconvolution of ultrasonic signals. Review of progress in quantitative nondestructive evaluation. 1998. P. 735-742.
4. **Crawford D. C. et. al.** Compensation for the signal processing characteristics of ultrasound B-mode scanners in adaptive speckle reduction. Ultrasound Med. Biol. (UK). 1993. Vol. 19. P. 469-485.
5. **Andrade Lima E. et. al.** Image processing techniques to remove depth bias effects in magnetic source images of deep cracks. Review of progress in quantitative nondestructive evaluation. Plenum press. New York. 1997. Vol.1. P. 797-803.
6. **Jianzhong C. et. al.** Noise analysis of digital ultrasonic system and elimination of pulse noise. Int. J. Pressure vessels piping. 1998. Vol.75. P. 887-890.
7. **Bengt M. et. al.** Weighted least squares pulse shaping filters with application to ultrasonic signals. IEEE Trans. UFFC. 1989. Vol. 36. P. 109-113.
8. **Venkantraman S. et. al.** Combining pulse compression and adaptive drive signal design to inverse filter the transducer system response and improve resolution in medical ultrasound. Med. Biomed. eng & comp. 1996. P/318-320.
9. **Izguierdo M. A. G. et. al.** Multipattern adaptive inverse filter for real time deconvolution of ultrasonic signals in scattering media. Sensors and actuators. 1999. Vol. 76. P. 26-31.
10. **Lim J. S. et al.** Advanced topics in signal processing. Prentice Hall. Tokyo. 1988. P. 1-55.
11. **Daubechies.**Ten Lectures on Wavelets. CBMS-NSF Regional Conf. Series in Appl. Math., Vol 61. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.
12. **Donoho D. L. and I. M. Johnstone**. Adapting to unknown smoothness via wavelet shrinkage. Journal of American Statistical Association, 1995, 1200-1224.
13. **Mallat S.** A theory for multiresolution signal decomposition: The wavelet representation.IEEE Transactions on Pattern Analysis and Machine Intelligence, 11, 1998, 674-693.
14. **Donoho D. L.and I. M. Johnstone.**Minimax estimation via Wavelet Shrinkage. The Annals of Statistics, 26, 1998, 879-921.
15. **Neumann M. H. and Von Sachs R.**Wavelet thresholding: beyond the Gaussian I.I.D. situation. In A. Antoniadis and G. Openheim Wavelets and Statistics, New York: Springer-Verlag 1995, 302-329.