SCIENTIFIC PAPERS, UNIVERSITY OF LATVIA, 2010. Vol. 756
COMPUTER SCIENCE AND INFORMATION TECHNOLOGIES 207–220 P.

A Modified Spline Interpolation Method for Function Reconstruction from Its Zero-Crossings

Viktorija Solovjova
University of Latvia, 19 Raina bulv., LV-1459, Riga, Latvia
viktorija.solovjova@gmail.com
http://www.lu.lv/df

There are different algorithms for reconstruction of a one-dimensional function from its zero-crossings. However, none of them is stable and computable in real time. Methods for one-dimensional function reconstruction from its generalized zero-crossings based on cubic spline interpolation are introduced in this paper. The main goal is to build an algorithm which is able to reconstruct a smooth function from the generalized zero-crossings as close to the original function as possible. The three main advantages of the approaches described in this paper are reliability, stability, and linear time processing.

Keywords: cubic spline interpolation, one-dimensional function reconstruction, zero-crossings.

1 Introduction

In this paper we are going to explore the task of one argument function reconstruction from its zero-crossings. In literature, a one-dimensional function is often called *one-dimensional signal*. The only argument of the function most often represents time or space. Compression and further reconstruction of a one dimensional function is required in optics, acoustics, crystallography, vision, and many other fields [1]. The literature on this problem describes different approaches to function reconstruction from its zero-crossings - see [1,2,3,4,5].
Venkatesh [1] has attempted to use generalised Hermite polynomials for reconstruction of an unknown function from zero-crossing information. He has also introduced a computational implementation of the algorithm. However, as the author states, it is very unlikely that a real-time solution to a practical reconstruction problem can be achieved with the algorithm. Thus, these results are interesting, but unfortunately unusable in practice.
Boufounos and Baraniuk [2] have assumed that the input data are a sparse signal, and formulated the reconstruction problem as minimization of sparsity. The authors state that the presented algorithm converges in typical cases and produces the correct solution with a very high probability. Yet the method is unstable, and return of a correct answer is not guaranteed.

Mallat [4] proposed to use wavelet zero-crossing representation as a complete signal description. He introduced a reconstruction algorithm based on alternate projections and very accurate reconstruction results were obtained. Later Mallat and Zhong [6] introduced the wavelet maxima representation as an alternative to the wavelet zero-crossing representation. Quite accurate reconstruction results were also demonstrated. Unfortunately, both approaches are unstable, i.e., is the algorithms are not guaranteed to process any kind of input data and return an acceptable answer in real time.

We introduce an algorithm based on the cubic spline interpolation method for function reconstruction. We also explore its several modifications, ranging from the basic, when a small amount of data are extracted from the original function, to more precise and efficient approaches.

In existing reconstruction algorithms, only x values are stored for zero-crossings of the function. In contrast, we extract function value $f(x)$ and sometimes its derivative $f'(x)$ at these points. Zero-crossings usually stand for the points, where the value of the function is 0. We extend this notion. In our interpretation, zero-crossings stand for the points where the value of the function $f(x)$ or the value of its derivative of any order $f^{(n)}(x)$ is 0. Obviously, these extensions give us high reconstruction precision potential.

2 Notations and Definitions

We are going to denote derivatives of the function $f(x)$ in the following way:

$$\begin{array}{ll} f^{(1)}(x) = f'(x) & \text{– first-order derivative of the function } f(x); \\ f^{(2)}(x) = f''(x) & \text{– second-order derivative of the function } f(x); \\ \ldots & \\ f^{(n)}(x) & \text{– n-order derivative of the function } f(x). \end{array} \tag{1}$$

We will call points of one-argument function f at which the function value or its derivative of any order is equal to 0 **zero-crossings**.

$$\text{Zero-crossings of } f(x) = \{x \mid f(x) = 0 \text{ or } f^{(n)}(x) = 0,\ n = 1, 2, \ldots\} \tag{2}$$

Moreover, we will call points at which the first-order derivative of the function is equal to 0 **first-order zero-crossings**, and points at which the second-order derivative of the function is 0 - **second-order zero-crossings**, and so on:

$$\begin{array}{ll} \{x \mid f(x) = 0\} & \text{– zero-order zero-crossings of } f(x); \\ \{x \mid f'(x) = 0\} & \text{– first-order zero-crossings of } f(x), \text{ also local extrema}; \\ \{x \mid f''(x) = 0\} & \text{– second-order zero-crossings of } f(x); \\ \ldots & \\ \{x \mid f^{(n)}(x)\} & \text{– n-order zero-crossings of } f(x). \end{array} \tag{3}$$

3 Reconstruction with Modified Spline Interpolation

In this section we present algorithms for one argument function reconstruction from its zero-crossings. Three different approaches will be used – from the basic approach, trying to remember less information, to extended approaches when a larger amount of data are extracted from the original function. With all the approaches we remember end points data concerning and data on first-order zero-crossings. In extended methods, second-order zero-crossings are also considered. It is possible to exploit higher order zero-crossings and slightly improve the quality of reconstruction, but that influences the amount of zero-crossing data; therefore, this kind of extension will not be considered.

3.1 Obtaining Function Zero-Crossings

If the function is given analytically, then we can calculate its derivatives in a straightforward way. If the function f is discrete, i.e. given as a set of pairs (x, y), we will use numerical differentiation for obtaining the derivatives of the function [7,8]. We assume that points of the function f are distributed evenly, i.e. , the interval between any two adjacent points is a constant value. After calculating all the required function derivatives for each point x, we can compare adjacent values. If the sign of the derivative of the function is changing at the point x, it is considered a zero-crossing point.

$$\begin{gathered} x_i \; is \; a \; zero-crossing \; of \; order \; n \; \leftrightarrow \\ \leftrightarrow f^{(n)}(x_i) \cdot f^{(n)}(x_{i-1}) < 0 \; and \; |f^{(n)}(x_i)| < |f^{(n)}(x_{i-1})| \\ x_{i-1} \; is \; a \; zero-crossing \; of \; order \; n \; \leftrightarrow \\ \leftrightarrow f^{(n)}(x_i) \cdot f^{(n)}(x_{i-1}) < 0 \; and \; |f^{(n)}(x_i)| \geq |f^{(n)}(x_{i-1})| \end{gathered} \tag{4}$$

3.2 The Cubic Spline Interpolation Method

First of all, we briefly describe plain cubic spline interpolation method. For an explicit explanation of the cubic spline interpolation method, see [9]. The aim of the method is to find a cubic spline $S(x)$ which interpolates the given points $x_1, x_2, ..., x_n$ and the values of the original function at points $f(x_1), f(x_2), ..., f(x_n)$. The cubic spline $S(x)$ is defined as the combination of cubic polynomials $S_i(x)$.

$$S(x) = \begin{cases} S_1(x) = a_1x^3 + b_1x^2 + c_1x + d_1, & if \; x_1 \leq x \leq x_2 \\ S_2(x) = a_2x^3 + b_2x^2 + c_2x + d_2, & if \; x_2 \leq x \leq x_3 \\ ... \\ S_{n-1}(x) = a_{n-1}x^3 + b_{n-1}x^2 + c_{n-1}x + d_{n-1}(x), & if \; x_{n-1} \leq x \leq x_n \end{cases} \tag{5}$$

There are 4 unknown variables in each equation, so we have in total $4n - 4$ coefficients which we have to determine to specify the function $S(x)$. If the coefficients are chosen in a way that $S(x)$ interpolates n given points and $S(x)$, $S'(x)$

and $S''(x)$ are continuous at points $x_2, x_3, ..., x_{n-1}$, then we get an interpolating curve $S(x)$, which is called a *cubic spline*.

The following requirements are used to calculate the coefficients of the function $S(x)$ with the cubic spline interpolation method.

For the inner points x_i, $i = 2, ..., n-1$

S1.1. $S(x)$ interpolates the point $(x_i, f(x_i))$: $S_i(x_i) = f(x_i)$
S1.2. $S(x)$ is continuous at x_i: $S_{i-1}(x_i) = S_i(x_i)$
S1.3. $S'(x)$ is continuous at x_i: $S'_{i-1}(x_i) = S'_i(x_i)$
S1.4. $S''(x)$ is continuous at x_i: $S''_{i-1}(x_i) = S''_i(x_i)$

For the end points x_1 and x_n

S2.1. $S_1(x_1) = f(x_1)$ and $S_{n-1}(x_n) = f(x_n)$
S2.2. $S'_1(x_1) = f'(x_1)$ and $S'_{n-1}(x_n) = f'(x_n)$

The second requirement for the end points (S2.2) expresses the idea of the so-called *clamped* cubic spline, when function derivatives are known at the end points. Both end points give us 4 equations and each inner point gives 4 equations. Thus, we get $4n - 4$ equations in total, which is enough to calculate the unknown coefficients of the cubic spline $S(x)$.

The cubic spline interpolation method almost ideally suits our aim, except for one defect. The fact that all the given points (with the exception of the end points) are generalized zero-crossings of the function is not considered in the method. Let us see an example where all the inner points are local extrema (first-order zero-crossings). The method searches for a curve which interpolates the given points. We can get inappropriate results like in the example shown in Fig. 1. As algorithm does not take into account local extrema, the reconstructed function can be completely different; however, it interpolates the given points. That is why the cubic spline interpolation method cannot be used without changing it.

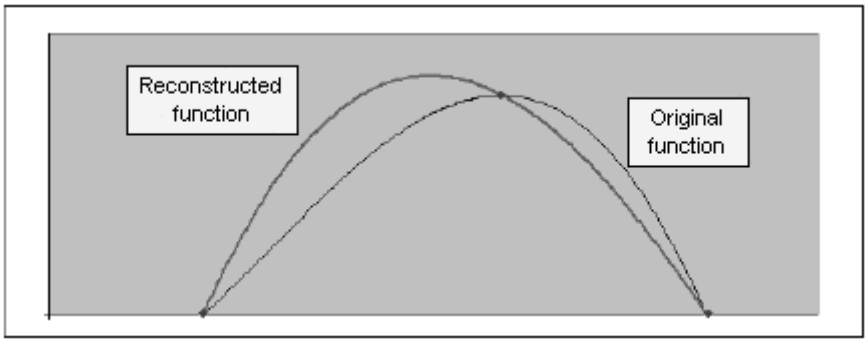

Fig. 1. Reconstruction with the plain spline interpolation method can cause inappropriate results

3.3 Function Reconstruction

Let us decide which points and which values at these points we are going to extract from the original function. We take first-order zero-crossings, i.e. local maxima and minima of the function. We remember x value and function value $f(x)$ at these points.

$$\{x_i, f(x_i) | f'(x_i) = 0, 1 \leq i < n\} \tag{6}$$

With the extended approaches, we also take second-order zero-crossings. For these points, we remember three values: x value, function value $f(x)$ and first derivative value $f'(x)$. Although not all values will be used in some approaches.

$$\{x_i, f(x_i), f'(x_i) | f''(x_i) = 0, 1 \leq i < n\} \tag{7}$$

As far as end points are concerned, it is enough to extract three values for each end point – x value, function value $f(x)$, and first derivative value $f'(x)$. Thus, we assume that we extract the following values.

$$x_1, f(x_1), f'(x_1), x_n, f(x_n), f'(x_n) \tag{8}$$

However constructing the algorithm we can get rid of some if they are redundant We now introduce three different approaches for function reconstruction from its zero-crossings.The first is a basic approach and only first-order zero-crossings are considered. The second-order zero-crossings are taken into account in the extended approaches also introduced.

Approach 1 (Basic). The following data of the local extrema of the original function and its end points were extracted from the original function:

$\{x_i, f(x_i) | f'(x_i) = 0, 1 \leq i < n\}$ – data of the local extrema,
$x_1, f(x_1), f'(x_1), x_n, f(x_n), f'(x_n)$ – data of the end points.

In the *basic approach* we define spline requirements in a slightly different way than in the plain cubic spline interpolation method. Instead of the requirement (S1.4) for the inner points in the plain cubic spline interpolation method that $S'(x)$ is continuous differentiable – $S''_i(x_i) = S''_{i-1}(x_i)$, we will require the first derivative at local extrema to be 0 (B1.3): thus, these points really are minima or maxima. Thus, the requirements we set for the polynomials are the following.

For the first-order zero-crossings (local extrema)

B1.1. $S(x)$ interpolates the point $(x_i, f(x_i))$: $S_i(x_i) = f(x_i)$
B1.2. $S(x)$ is continuous at x_i: $S_{i-1}(x_i) = S_i(x_i)$
B1.3. $S'(x)$ is 0 at x_i: $S'_i(x_i) = S'_{i-1}(x_i) = 0$

For the end points x_1 and x_n

B2.1. $S_1(x_1) = f(x_1)$ and $S_{n-1}(x_n) = f(x_n)$

B2.2. $S'_1(x_1) = f'(x_1)$ and $S'_{n-1}(x_n) = f'(x_n)$

Both end points give us 4 equations and each first-order zero-crossing point gives us 4 equations. Obviously, the number of first-order zero-crossings is $n-2$. Thus, the total count of equations is $4n-4$, which is the same as the count of unknown variables (cubic spline coefficients).
To obtain the coefficients of the polynomials, we use a linear equation solver. For a small amount of data, the Gaussian elimination method can be used. If the linear equation system is large, an iterative method for sparse linear systems [10] can be used to solve it.

Approach 2 (Extended). The following data of the first- and second-order zero-crossings of the original function and its end points were extracted from the original function:

$\{x_i, f(x_i) | f'(x_i) = 0, 1 \leq i < n\}$ – data of the local extrema,
$\{x_i | f''(x_i) = 0, 1 \leq i < n\}$ – data of the second-order zero-crossings,
$x_1, f(x_1), f'(x_1), x_n, f(x_n), f'(x_n)$ – data of the end points.

We use the same algorithm described in the *basic approach* but with different requirements. In addition we extract data about second-order zero-crossings. We search for a cubic spline $S(x)$ defined as a combination of cubic polynomials $S_i(x)$. For the second-order zero-crossings, we take all the same requirements as in the basic cubic spline interpolation algorithm, but the requirement (S1.1) for a precise function value at points $S(x_i) = f(x_i)$ is replaced with the requirement that second derivative of the function at these points is 0 (E2.3). We assume that function value at these points is not as essential as the information that the second derivative of the function is 0.

For the first-order zero-crossings (local extrema)

E1.1. $S(x)$ interpolates the point $(x_i, f(x_i))$: $S_i(x_i) = f(x_i)$
E1.2. $S(x)$ is continuous at x_i: $S_{i-1}(x_i) = S_i(x_i)$
E1.3. $S'(x)$ is 0 at x_i: $S'_i(x_i) = S'_{i-1}(x_i) = 0$

For the second-order zero-crossings

E2.1. $S(x)$ is continuous at x_i: $S_{i-1}(x_i) = S_i(x_i)$
E2.2. $S'(x)$ is continuous at x_i: $S'_i(x_i) = S'_{i-1}(x_i)$
E2.3. $S''(x)$ is continuous and equal to 0: $S''_i(x_i) = S''_{i-1}(x_i) = 0$

For the end points x_1 and x_n

E3.1. $S_1(x_1) = f(x_1)$ and $S_{n-1}(x_n) = f(x_n)$
E3.2. $S'_1(x_1) = f'(x_1)$ and $S'_{n-1}(x_n) = f'(x_n)$

Both end points give us 4 equations, each first- or second-order zero-crossing also gives us 4 equations. Thus, we get a system of $4n-4$ equations where the unknown variables are the coefficients of the spline polynomials. When the linear

equation system is solved, it gives us the coefficients of the spline and allows to define the function $S(x)$.

Approach 3 (Extended Precise). The following data of the first- and second-order zero-crossings of the original function and its end points were extracted from the original function:

$\{x_i, f(x_i) | f'(x_i) = 0, 1 \leq i < n\}$ – data of the local extrema,
$\{x_i, f(x_i), f'(x_i) | f''(x_i) = 0, 1 \leq i < n\}$ – data of the second-order zero-crossings,
$x_1, f(x_1), f'(x_1), x_n, f(x_n), f'(x_n)$ – data of the end points.

This approach is very similar to the previous approach. But here instead of requiring that the second derivative of the function $S''(x)$ is continuous and equal to 0 at the second-order zero-crossings (E2.3), we give the exact value of the function (P2.3) and the exact value of the first derivative of the function (P2.4) at these points.

For the first-order zero-crossings (local extrema)

P1.1. $S(x)$ interpolates the point $(x_i, f(x_i))$: $S_i(x_i) = f(x_i)$
P1.2. $S(x)$ is continuous at x_i: $S_{i-1}(x_i) = S_i(x_i)$
P1.3. $S'(x)$ is 0 at x_i: $S'_i(x_i) = S'_{i-1}(x_i) = 0$

For the second-order zero-crossings

P2.1. $S(x)$ is continuous at x_i: $S_{i-1}(x_i) = S_i(x_i)$
P2.2. $S'(x)$ is continuous at x_i: $S'_i(x_i) = S'_{i-1}(x_i)$
P2.3. $S(x)$ interpolates the point $(x_i, f(x_i))$: $S_i(x_i) = f(x_i)$
P2.4. $S'(x)$ interpolates the point $(x_i, f'(x_i))$: $S'_i(x_i) = f'(x_i)$

For the end points x_1 and x_n

P3.1. $S_1(x_1) = f(x_1)$ and $S_{n-1}(x_n) = f(x_n)$
P3.2. $S'_1(x_1) = f'(x_1)$ and $S'_{n-1}(x_n) = f'(x_n)$

Similarly as in the previous approaches, each inner point gives us 4 equations, but each end point gives 2. Thus, the total count of equations is $4n - 4$. Having solved this linear system, we get the values of the coefficients of the polynomials and hence the definition of the function we are looking for – $S(x)$.

3.4 Analysis of the Approaches

The *extended* and *extended precise* approaches are expected to be more precise as they demand more information about the original function. Nevertheless, now we see that actually the *basic approach* is more reliable because it guarantees that all the local extrema of the reconstructed function are exactly the same as the local extrema of the original function. The extended approaches cannot guarantee such accuracy.

Theorem 1. Let $f(x)$ be the original function and $S(x)$ – the function reconstructed from its zero-crossings with the *basic approach*. Then for $x \in [x_2, x_{n-1}]$, the local extrema of the function $S(x)$ are exactly the same as the local extrema of the original function $f(x)$, i.e., if $x \in [x_2, x_{n-1}]$, then

$$\begin{array}{l}(a)\ S'(x) = 0 \Leftrightarrow f'(x) = 0, \\ (b)\ S'(x) = 0 \Rightarrow S(x) = f(x).\end{array} \tag{9}$$

Proof. (a) The proof is acquired in two parts. Let us start with $f'(x) = 0 \Rightarrow S'(x) = 0$. As x is a first-order zero-crossing $x = x_i$ for some i, the linear equation system contains the following equation.

$$S'_i(x_i) = S'_{i-1}(x_i) = 0 \tag{10}$$

Then from Equation 10 and the definition of the cubic spline $S(x)$ it follows that $S'(x) = 0$.

Now let us prove the second part – that if $x \in [x_2, x_{n-1}]$, then $S'(x) = 0 \Rightarrow f'(x) = 0$. From the definition of the *basic approach* we know that all inner points are first-order zero-crossings:

$$f'(x_i) = S'_i(x_i) = 0 \text{ for } i = 2, ..., n-1. \tag{11}$$

This means that each function $S_i(x)$ for $i = 2, ..., n-1$ has two local extrema points – one at x_i and the other at x_{i+1}. Since all the functions $S_i(x)$ are cubic polynomials, they cannot contain more than 2 local extrema. This means that the only local extrema in each interval $[x_i, x_{i+1}]$ are x_i and x_{i+1}. All these intervals form the interval $[x_2, x_{n-1}]$. Thus, we can conclude that the only points where $S'(x) = 0$ are the points where $f'(x) = 0$ (the end points of the subintervals). Thus, for all $x \in [x_2, x_{n-1}] : S'(x) = 0 \Rightarrow f'(x) = 0$.

(b) We have proven that $S'(x) = 0 \Rightarrow f'(x) = 0$ in (a). Therefore, if $S'(x) = 0$, then x is a first-order zero-crossing of the original function - $x = x_i$ for some i, and the linear equation system contains the following equation.

$$S_i(x_i) = S_{i-1}(x_i) = f(x_i) \tag{12}$$

Then from Equation 12 and the definition of the cubic spline $S(x)$ it follows that $S(x_i) = f(x_i)$.

Theorem 1 states that in almost all of the domain (apart from two end subintervals), all the local extrema of the function reconstructed with the *basic approach* are exactly the same as the local extrema of the original function. Thus, the *basic approach* guarantees that the reconstructed function exactly reflects all the local extrema of the original function.

Now let us analyze the performance of the approaches. Both *basic* and *extended precise* approaches generate a large linear equation system. Instead, a set of linear equation systems of a constant size could be generated. For each first- and second-order zero-crossing and for the end points, we give a precise value of the function and a precise value of the derivative with both approaches.

$$\begin{array}{l}S_{i-1}(x_i) = S_i(x_i) = f(x_i) \\ S'_{i-1}(x_i) = S'_i(x_i) = f'(x_i)\end{array} \tag{13}$$

Thus, the coefficients of every two adjacent polynomials $S_{i-1}(x)$ and $S_i(x)$ do not depend on each other. That is why a separate linear equation system can be generated for each polynomial $S_i(x)$. Hence, each of these linear equation systems is of a constant size and can be computed in a constant time. It is obvious that the whole set of linear equation systems can be solved in linear time $O(n)$, where n is the count of the first- and second-order zero-crossings. Thus, there are linear time algorithms in both the *basic approach* and *extended precise approach*.

In the *extended approach*, the polynomials are connected to each other at the second-order zero-crossings. Nevertheless, the linear equation system can still be split into several small systems – one for each interval from one first-order zero-crossing to the next.

3.5 Results

In this sub-section we present the examples of function reconstruction using all three approaches. Different functions, starting with smooth analytically defined functions up to more complex functions, are reconstructed with the approaches described above. The reconstruction accuracy of each approach is discussed and compared to others.

$\boldsymbol{f(x) = (x-2)(x-9)(x-15)}$ The function is smooth and simple, it has only two local extrema points since it is a cubic polynomial. This function is interesting because it is defined in the same way as the function we get from reconstruction. That is why we can expect that the reconstruction quality will be very high. The results of the reconstruction are shown in Fig. 2. In figures below the original function is drawn black, reconstructed functions – gray, first-order zero-crossings (local extrema) are shown as black points, and second-order zero-crossings as gray points. The reconstruction quality in this case appeared to be ideal with each approach because of the fact that the data were extracted from the function of the same nature as the function we are trying to construct.

$\boldsymbol{f(x) = (2x-13)(\sin\frac{x}{5} + 2\cos\frac{x}{3} - 3\cos(\frac{x}{2} - 3))}$ Let us take a little more complex, however, also a smooth function. As shown in Fig. 3, there is a slight difference in the results of the approaches. The *basic approach* gives a slight deviation from the original function, but the *extended precise approach* – the best result. However, the difference between them is very small. Again all three methods give us good results.

$\boldsymbol{f(x) = e^{-x^2}}$ Let us take a Gaussian function whose nature is far from the nature of polynomial functions. The results of the three approaches are shown in Fig. 4. The difference between all three is remarkable. The *extended precise approach* gives a very good result in this case. The other two approaches also give

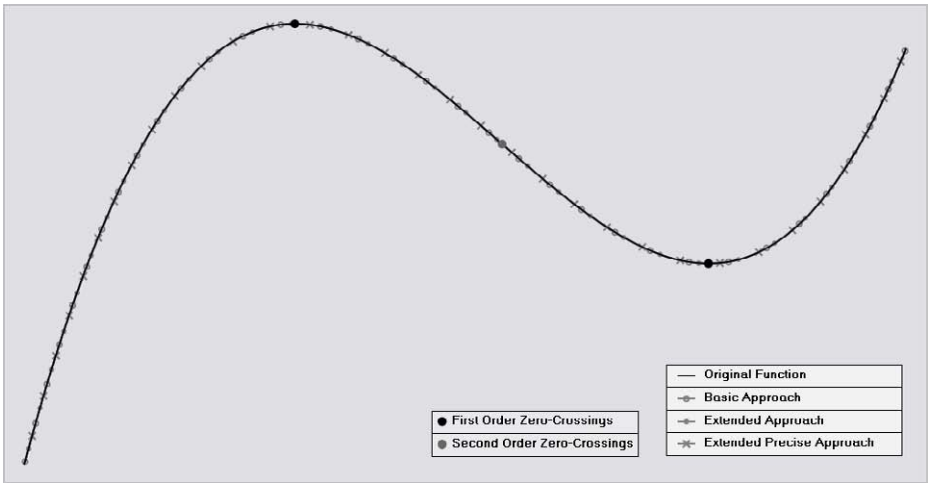

Fig. 2. Reconstruction of the function $f(x) = (x-2)(x-9)(x-15)$ at $x \in [0, 16]$ with the *basic approach*, *extended approach*, and *extended precise approach*

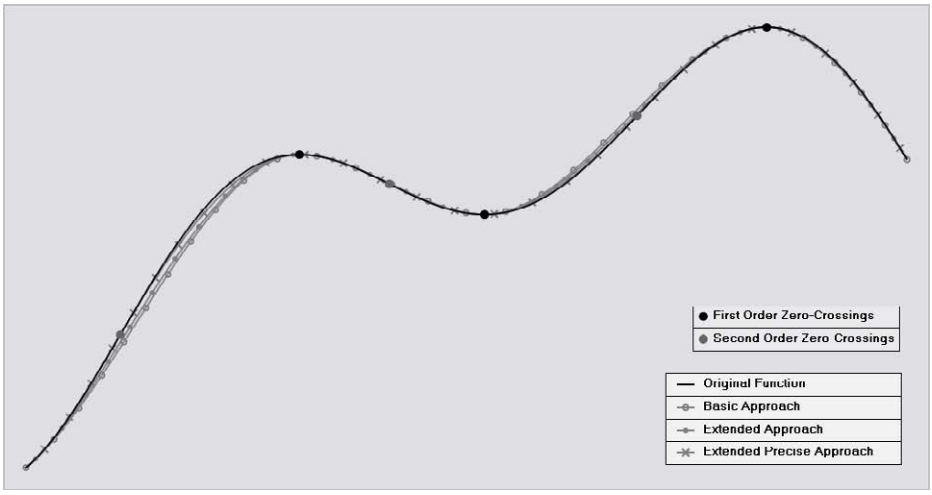

Fig. 3. Reconstruction of the function $f(x) = (2x-13)(\sin\frac{x}{5}+2\cos\frac{x}{3}-3\cos(\frac{x}{2}-3))$ at $x \in [0, 16]$ with the *basic approach*, *extended approach*, and *extended precise approach*

acceptable results; however, the reconstructed functions are a little smoothed out in comparison to the original function.

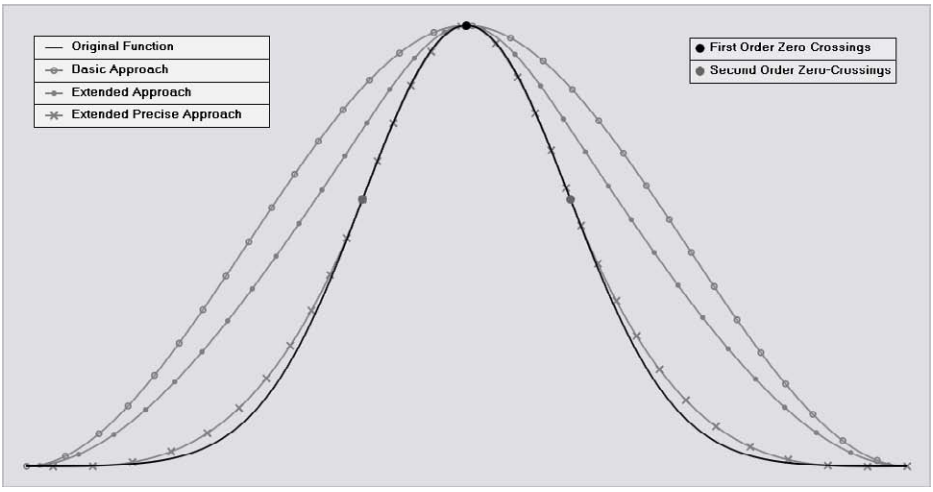

Fig. 4. Reconstruction of the function $f(x) = e^{-x^2}$ at $x \in [-3, 3]$ with the *basic approach*, *extended approach*, and *extended precise approach*

$\boldsymbol{f(x) = e^{-x^4}}$ We also take a modification of a Gaussian function with a flattened top. As we see in Fig. 5, all the approaches again give good results, yet the result of the *extended precise approach* is the most accurate.

A function with sharp angles. Let us also see the example of reconstruction of a broken-line function. This function has very little in common with polynomial functions; therefore, not very impressive results were expected. However, Fig. 6 shows that even such case is processed with a good precision. Again the *extended precise approach* gives the most accurate result. Nevertheless, the *extended approach* and the *basic approach* give almost the same result; thus, in this case the *extended approach* is not a significant improvement on the *basic approach*.

The examples above show that the quality of reconstruction is improving with an increasing amount of data stored. The most accuracy is achieved with the *extended precise approach*. However, we have seen in the analysis section that the *extended* approaches do not guarantee that all the local extrema will be identical to the local extrema of the original function. Therefore, we recommend to use the *basic approach* due to its reliability and stability.

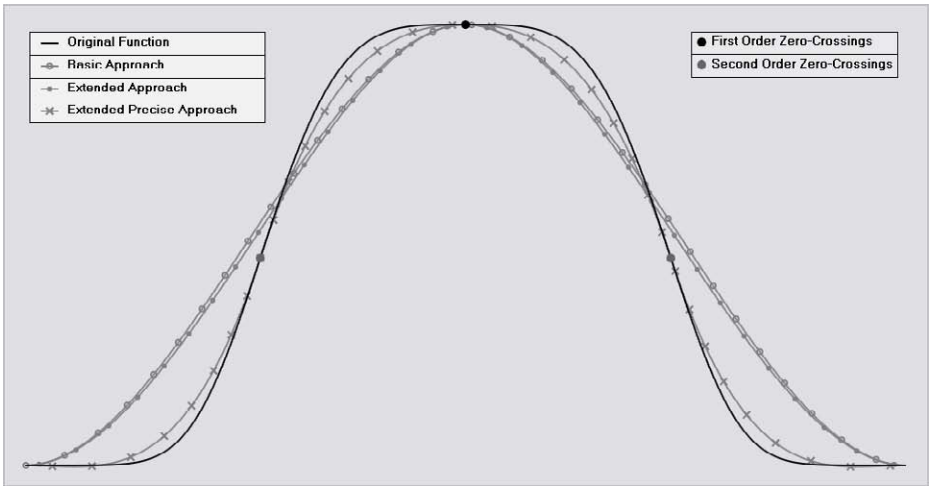

Fig. 5. Reconstruction of the function $f(x) = e^{-x^4}$ at $x \in [-2, 2]$ with the *basic approach*, *extended approach*, and *extended precise approach*

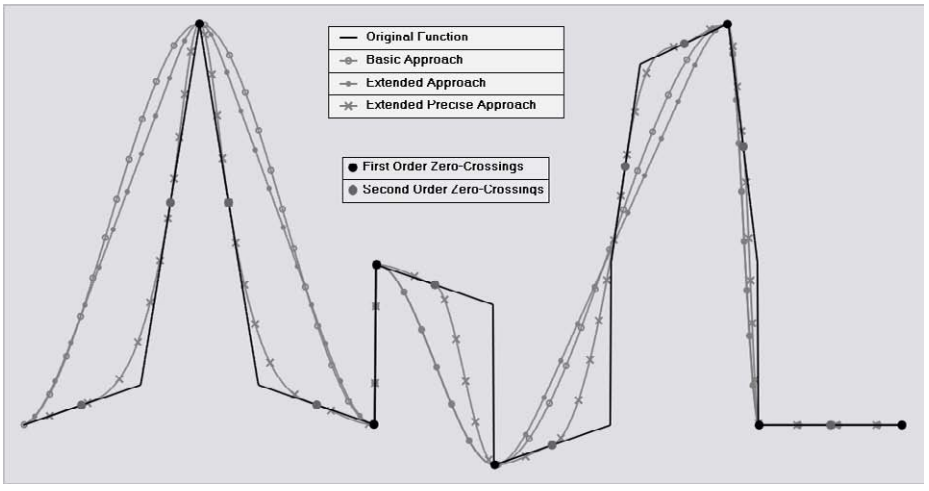

Fig. 6. Reconstruction of the broken line function with the *basic approach*, *extended approach*, and *extended precise approach*

4 Conclusion

We have introduced three approaches for function reconstruction from its zero-crossings. The most interesting among them is the *basic approach* since it is precise, stable, and computable in linear time. Previously known approaches focus on a particular type of functions, and the algorithms work well with this particular type of functions. However, in general, they are unstable and usually do not guarantee return of the result in real time.

Three main advantages of the *basic approach* can be pointed out. First, it guarantees return of the result in linear time. Second, the approach is precise, it perfectly reconstructs all the local extrema of the original function. Third, the approach is stable: it always gives the result even if the input data contain errors, i.e., the approach is tolerant to input data inaccuracy.

The presented approaches can also be extended for two-dimensional functions. Accurate reconstruction of two-dimensional functions from a relatively small set of data is useful in the area of image compression because any image can be represented as a two-argument function or a combination of several two-dimensional functions.

Reconstruction of two-dimensional functions from zero-crossings should be explored in further research. A wide range of research papers, for example, [11,12,13] are dedicated to two-dimensional function reconstruction from zero-crossings, actually from the so-called edges and ridges. The results of these methods are very good, yet the stability of the algorithms has not been proven.

The methods for reconstruction of one-dimensional functions presented in this paper appeared to be very efficient and practically useful. Therefore, we expect that these methods can most definitely be successfully extended for two-dimensional functions.

References

1. Y. V. Venkatesh. Hermite polynomials for signal reconstruction from zero-crossings. Part 1: One-dimensional signals. *IEE Proceedings I of Communications, Speech and Vision*, vol. 139 (6), 1992, pp. 587–596.
2. P. T. Boufounos, R. G. Baraniuk. Reconstructing sparse signals from their zero crossings. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008)*, pp. 3361–3364.
3. Z. Berman, J. S. Baras. Properties of the multiscale maxima and zero-crossings representations. *IEEE Transactions on Signal Processing*, vol. 41, 1993, pp. 3216–3231.
4. S. Mallat. Zero-crossings of wavelet transform. *IEEE Transactions on Information Theory*, vol. 37, 1991, pp. 1019–1033.
5. R. Hummel, R. Moniot. Reconstructions from zero crossings in scale space. *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, 1989, pp. 2111–2130.
6. S. Mallat, S. Zhong. Characterization of signals from multiscale edges. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, vol. 14 (7), 1992, pp. 710–732.

7. A. Iserles. A First Course in the Numerical Analysis of Differential Equation/ *Cambridge Texts in Applied Mathematics*, 2nd edition. Cambridge University Press, 2008.
8. J. H. Mathews, K. D. Fink. Numerical Methods Using Matlab, 4th edition. Prentice-Hall Pub., 2004.
9. G. D. Knott. Interpolating Cubic Splines. *Progress in Computer Science and Applied Logic (PCS)*. Boston: Birkhäuser, 1999.
10. Y. Saad. Iterative Methods for Sparse Linear Systems, 2d edition. Society for Industrial and Applied Mathematics, 2000.
11. J. H. Elder. Are Edges Incomplete? *International Journal of Computer Vision*, vol. 34 (2/3), 1999, pp. 97–122.
12. E. Barth, T. Caelli, C. Zetzsche. Image Encoding, Labeling and Reconstruction from Differential Geometry. *CVGIP: Graphical Models and Image Processing*, vol. 55 (6), 1993, pp. 428–446.
13. S. Carlsson. Sketch based coding of grey level images. *Signal Processing*, vol. 15, 1988, pp. 57–83.