

**Composite quadrature formulae for the  
approximation of wavelet coefficients of  
piecewise smooth and singular functions**

*Daan Huybrechs and Stefan Vandewalle*

*Report TW 388, April 2004 (revised October 2004)*



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

# Composite quadrature formulae for the approximation of wavelet coefficients of piecewise smooth and singular functions

*Daan Huybrechs and Stefan Vandewalle*

*Report TW 388, April 2004 (revised October 2004)*

Department of Computer Science, K.U.Leuven

## **Abstract**

The computation of wavelet coefficients of a function typically requires the computation of a large number of integrals. These integrals represent the inner product of that function with a wavelet function on different scales, or with the corresponding scaling function on a fine scale.

We develop quadrature rules for those integrals that converge fast for piecewise smooth and singular functions. They do not require the evaluation of the scaling function, and the convergence does not depend on the smoothness of that function. The analysis and computation is based completely on the filter coefficients that define the scaling function. An application is presented from the field of electromagnetics, involving the inner product of a singular function with two-dimensional tensor-product wavelets.

**Keywords :** wavelet approximation, numerical integration, singular functions  
**AMS(MOS) Classification :** Primary : 65D32, Secondary : 42C40, 35J05.

# Composite quadrature formulae for the approximation of wavelet coefficients of piecewise smooth and singular functions

Daan Huybrechs and Stefan Vandewalle

Katholieke Universiteit Leuven  
Department of Computer Science  
Celestijnenlaan 200A, B-3001 Leuven, Belgium.  
{Daan.Huybrechs,Stefan.Vandewalle}@cs.kuleuven.ac.be

## Abstract

The computation of wavelet coefficients of a function typically requires the computation of a large number of integrals. These integrals represent the inner product of that function with a wavelet function on different scales, or with the corresponding scaling function on a fine scale.

We develop quadrature rules for those integrals that converge fast for piecewise smooth and singular functions. They do not require the evaluation of the scaling function, and the convergence does not depend on the smoothness of that function. The analysis and computation is based completely on the filter coefficients that define the scaling function. An application is presented from the field of electromagnetics, involving the inner product of a singular function with two-dimensional tensor-product wavelets.

## 1 Introduction

In a variety of applications, e.g., in a multiresolution analysis or in a wavelet-Galerkin procedure for solving partial differential equations, the computation of the wavelet coefficients of a function  $f$  is required. These coefficients are defined by

$$c_{n,k} := \int_{-\infty}^{\infty} f(x)\phi_{n,k}(x) dx. \quad (1)$$

Here,  $\phi_{n,k}(x) = 2^{n/2}\phi(2^n x - k)$ , with  $\phi$  being the scaling function corresponding to some wavelet function  $\psi$ . Index  $n$  denotes the scale of the function, while index  $k$  indicates the shift. The integrals represent the  $L_2$  inner product of the function  $f$  with the scaling function, and arise naturally in most wavelet analyses. In this paper, we present a family of fast numerical quadrature rules to compute these integrals for piecewise smooth or singular functions  $f$ . Throughout this paper, we assume the scaling function has limited support, i.e.,  $\text{supp}(\phi(x)) = [s_1, s_2]$ .

General quadrature rules depend on the smoothness of the integrand. Discontinuities of the integrand or of any of its derivatives may disturb the convergence of these methods. They will fail for scaling functions with only small regularity. Also, it can be computationally expensive to evaluate  $\phi$  and  $f$ , so we would like to minimize the number of function evaluations. In some cases there is no explicit formula for  $\phi$  available.

Our goal is to develop quadrature rules that exhibit convergence characteristics that depend only on the smoothness of  $f$ , and to reuse function evaluations whenever possible. To compute the rules themselves, we will only need the coefficients  $h_k$  of the refinement equation for  $\phi$ ,

$$\phi(x) = \sqrt{2} \sum_{k=s_1}^{s_2} h_k \phi(2x - k). \quad (2)$$

Our approach is based on the method originally discussed in [3, 13]. The results of [3] were generalised in [13] to higher order interpolatory quadrature rules, and extended with a stable method to construct those quadrature rules using Chebyshev polynomials. Gauss type rules for refinable functions were developed in [2, 10]. A different method without quadrature rules, based solely on the refinement equation, was proposed in [7]. That method is an iterative scheme, that requires the solution of an eigenvalue problem. Our approach differs from the earlier work, in that it also converges for functions that are only piecewise smooth, or even singular in a finite number of points.

We recall the quadrature method of [13] for smooth functions in Section 2. In Section 3, the method is extended to cover piecewise smooth functions. In Section 4 we cover singular functions. We conclude in Sections 5 with some remarks on higher dimensional problems, and with a two-dimensional example from the field of electromagnetics. There, the boundary element method (BEM) for integral equations is used for the solution of the Helmholtz equation on a two-dimensional domain. This BEM approach requires inner products of a singular function with two-dimensional tensor-product wavelet functions.

## 2 An integration rule for smooth functions

### 2.1 The quadrature rule and its construction

The modus operandi is based on a technique described by Sweldens and Piessens in [13]. There, the authors have constructed a quadrature rule  $Q[\cdot]$  that only requires the evaluation of  $f$  in a number of quadrature points  $x_i$ ,

$$\int_{-\infty}^{\infty} f(x)\phi(x) dx \simeq Q[f(x)] = \sum_{i=1}^r c_i f(x_i). \quad (3)$$

The convergence rate of this integration rule depends on the number of abscissae  $r$  and on the smoothness properties of  $f$ , but not on the smoothness of the scaling function  $\phi$ .

The integrals of type (1) can be approximated for all values of  $n$  and  $k$  using rule (3), by

$$c_{n,k} \simeq 2^{-n/2} Q[f(2^{-n}(x+k))].$$

The abscissae  $x_i$  are chosen on a regular grid that enables reusing the values  $f(x_i)$  for neighbouring scaling functions. Points of the form  $x_i = (i-1)2^s$ , e.g., with  $s$  a negative integer, are good candidates, since any integer shift transforms the set of points  $\{x_i\}$  onto itself. A real shift  $\tau$  on the entire grid preserves that property.

These observations lead one to consider the points  $x_i = (i-1)2^s + \tau$ . In [13] it is shown that the number of shared function evaluations can be large: if  $s = 0$ , there is only one extra

evaluation of  $f$  needed for every additional coefficient. All the other values that are needed for the computation of  $c_{n,k}$  in (1), are also needed for  $c_{n,k-1}$ . The parameter  $\tau$  is an additional degree of freedom, and can be used to increase the order of the quadrature rule.

The rule (3) is derived by imposing that the quadrature is exact for all polynomials of degree lower than  $r$ ,

$$Q[P_l(x)] = M^l, \quad l = 0, 1, \dots, r-1, \quad (4)$$

with

$$M^l := \int_{-\infty}^{\infty} P_l(x) \phi(x) dx.$$

The polynomials  $P_l(x), l = 0, 1, \dots, r-1$  in (4) form a basis for the set of polynomials of degree lower than  $r$ . The unknowns are the quadrature weights, and the matrix representing the system is found by simply evaluating  $P_l(x)$  in the quadrature abscissae. Since the resulting system of equations is ill conditioned for the monomial basis  $P_l(x) = x^l$ , Sweldens and Piessens considered using the Chebyshev polynomials of the first kind,  $T_l(x)$ . These polynomials form an orthogonal basis on  $[-1, 1]$  for the weight function  $w(x) = (1 - x^2)^{-1/2}$ . When properly scaled Chebyshev polynomials are used in (4), the system is well conditioned. Scaling the interval  $[s_1, s_2]$  to  $[-1, 1]$ , we have the basis polynomials

$$P_l(x) = T_l \left( 2 \frac{x - s_1}{s_2 - s_1} - 1 \right). \quad (5)$$

By making use of the refinement equation (2) and properties of Chebyshev polynomials, an explicit formula for the moments  $M_l$  can be derived [13]. The system (4) can subsequently be solved in order to find the quadrature weights.

## 2.2 Some comments on the accuracy of the quadrature rule

With a regular grid of  $r$  abscissae, we typically expect for the corresponding quadrature rule a degree of accuracy, denoted by  $q$ , of at most  $q = r - 1$ . The integration error is then of the order  $O(h^{q+1})$ , with  $h$  being proportional to the interval length  $s_2 - s_1$ , i.e., the support of  $\phi$ . In some cases, we can achieve an order  $q = r$  with a proper choice of  $\tau$ . The number  $r$  depends on the parameter  $s$  that determines the regular grid. In [13], it is shown that the application of the rule to a scaling function  $\phi_{n,k}(x)$  on scale  $n$  leads to a relative error on the approximation of  $c_{nk}$  of  $O(2^{-n(q+1)})$ . As one might have expected, a finer scale, i.e., a reduction of the integration interval, leads to a more accurate result.

**Remark 2.1.** The technique described here leads to an interpolatory integration rule, with the scaling function itself as a weight function. In general, the weights for such rules can have alternating signs, which negatively impacts the stability of the computations. Even if the weight function is strictly positive, there will be at least one negative weight for each rule with a sufficiently high degree of accuracy. For weight functions that switch signs, as quite a few scaling functions do, each rule has negative and positive weights. For this reason, we will quantify the stability properties of the weights in the examples, presented further on by using the sum of absolute values as a measure [9].

**Remark 2.2.** If we abandon the principle of using a regular grid for the abscissae, better rules can be made by constructing Gaussian quadrature rules with  $\phi$  as weight function. Since for Gaussian rules the weight function has to be positive, for some scaling functions

$g(x) := \phi(x) + c$  is used instead. Here, the constant  $c$  is chosen such that  $g$  is a suitable nonnegative weight function. Such rules are constructed in [2]. In that setting, one loses, however, the ability to reuse function evaluations.

### 3 Improving accuracy by piecewise integration

#### 3.1 A composite quadrature approach

For a larger number of abscissae, say  $r \sim 30$ , the high order methods from [13] become unstable due to large quadrature weights with alternating signs. As with composite quadrature rules, the accuracy can be improved by splitting the integration interval, and by applying a lower order quadrature rule on each subinterval. Hence, we aim for a new rule on the subinterval  $[a, b] \subset [s_1, s_2]$

$$Q_{a,b}[f(x)] \simeq \int_a^b f(x)\phi(x) dx. \quad (6)$$

The support  $[s_1, s_2]$  of the scaling function  $\phi$  will then be divided into a sequence of intervals  $[a_i, b_i]$ . Typically, the integration subinterval  $[a_i, b_i]$  will have its endpoints on points of discontinuity of  $f$ . When the goal is to improve the integration accuracy for a smooth function  $f$ , the points  $a_i$  and  $b_i$  can be chosen to be regularly distributed. This will again enable most function evaluations to be shared.

#### 3.2 Computation of the moments

In order to find the quadrature weights, we need to compute the moments of the scaling function on the interval  $[a, b]$ ,

$$M_{a,b}^l := \int_a^b P_l(x)\phi(x)dx. \quad (7)$$

Using refinement equation (2), we have, for instance, for the zeroth order moment,

$$M_{a,b}^0 := \int_a^b \phi(x)dx = \frac{\sqrt{2}}{2} \sum_k h_k \int_{2a-k}^{2b-k} \phi(x)dx = \frac{\sqrt{2}}{2} \sum_k h_k M_{2a-k, 2b-k}^0. \quad (8)$$

This formula expresses  $M_{a,b}^0$  as a linear combination of zeroth order moments on the intervals  $[2a-k, 2b-k]$ . Applying (8) recursively for the moments in the right hand side, leads to a set of linear equations for the unknowns  $M_{a_i, b_i}^0$ , for different intervals  $[a_i, b_i]$ . Since the support of  $\phi$  is finite, only those moments where the interval intersects the support  $[s_1, s_2]$  are nonzero.

We define  $S(a, b)$  as the set of all intervals generated starting from  $[a, b]$ , by recursively adding  $[2a_i - k, 2b_i - k] \cap [s_1, s_2]$ ,  $k = s_1, \dots, s_2$ , for each interval  $[a_i, b_i]$  in the set. These intervals correspond to the unknown moments in (8). We will first generalize equation (8) to moments of higher order, and then we will discuss the size of the set  $S(a, b)$ .

#### 3.3 An algorithm based on using Chebyshev polynomials

For the computation of  $M_{a,b}^l$ , we may use the Chebyshev polynomials  $T_l(x)$  scaled to the interval  $[s_1, s_2]$ . In this way we avoid evaluating the polynomials outside the interval  $[-1, 1]$ .

One scaling can be converted to another easily by using the following relation,

$$T_n \left( \frac{x + \lambda_1}{L_1} \right) = 2^{-n} \sum_{i=0}^n w_i^{(n)} T_i \left( \frac{x + \lambda_2}{L_2} \right), \quad (9)$$

with real parameters  $\lambda_1, L_1, \lambda_2$  and  $L_2$ , and  $L_1, L_2 \neq 0$ . Note that the parameter  $w_i^{(n)}$  depends on  $\lambda_1, L_1, \lambda_2, L_2$ . A stable recursive scheme to compute the coefficients  $w_i^{(n)}$  in (9) is given in the Appendix. The scheme is based directly on the three-term recurrence relation for Chebyshev polynomials. A similar scheme, only slightly different, was used and derived in [13].

The moments can then be found in the following way. If  $l = 0$ , we solve the system of equations of type (8). For larger  $l$ , we use (9) to introduce lower order Chebyshev polynomials,

$$\begin{aligned} M_{a,b}^{l+1} &= \int_a^b T_{l+1} \left( \frac{x + \lambda_1}{L_1} \right) \phi(x) dx \\ &= \frac{\sqrt{2}}{2} \sum_k h_k \int_{2a-k}^{2b-k} T_{l+1} \left( \frac{x + k + 2\lambda_1}{2L_1} \right) \phi(x) dx \\ &= \frac{\sqrt{2}}{2} \sum_k h_k \sum_{i=0}^{l+1} \int_{2a-k}^{2b-k} 2^{-(l+1)} w_i^{(l+1)}(k) T_i \left( \frac{x + \lambda_1}{L_1} \right) \phi(x) dx \\ &= \frac{\sqrt{2}}{2} \sum_k h_k \sum_{i=0}^{l+1} 2^{-(l+1)} w_i^{(l+1)}(k) M_{2a-k, 2b-k}^i. \end{aligned}$$

The coefficients  $w_i^{(l+1)}(k)$  are written this way in order to make the dependence on the parameter  $k$  explicit in the equation, since we applied (9) with a  $k$ -dependent parameter  $\lambda_2$ . The coefficients need to be computed once for every value of  $k$ . The parameters  $\lambda_1$  and  $L_1$  are defined here such that  $T_l \left( \frac{x + \lambda_1}{L_1} \right) = T_l \left( 2 \frac{x - s_1}{s_2 - s_1} - 1 \right)$ .

A separation of the known and unknown components yields the equation

$$M_{a,b}^{l+1} - 2^{-(l+\frac{3}{2})} \sum_k h_k w_{l+1}^{(l+1)}(k) M_{2a-k, 2b-k}^{l+1} = 2^{-(l+\frac{3}{2})} \sum_k h_k \sum_{i=0}^l w_i^{(l+1)}(k) M_{2a-k, 2b-k}^i. \quad (10)$$

The right hand side of (10) is fully known and can easily be evaluated at step  $l + 1$ .

In order to find the quadrature weights for the integration on the interval  $[a, b]$ , we need to solve a system of equations similar to (4),

$$Q \left[ T_l \left( 2 \frac{x - s_1}{s_2 - s_1} - 1 \right) \right] = M_{a,b}^l.$$

The matrix entries are evaluations of Chebyshev polynomials, scaled to  $[s_1, s_2]$ , in the interval  $[a, b]$ . In order to obtain a good condition number, we will use the same matrix as in (4). The Chebyshev polynomials are then scaled to the interval  $[a, b]$ . The new right hand side can be found from the moments  $M_{a,b}^l$ , by combining (7) and (9),

$$Q \left[ T_l \left( 2 \frac{x - a}{b - a} - 1 \right) \right] = \tilde{M}_{a,b}^l := \sum_{i=0}^n 2^{-n} w_i^{(n)} M_{a,b}^i.$$

### 3.4 Computational complexity of the construction

The performance of the algorithm described above, depends on the cardinality of the set  $S(a, b)$  defined in Section 3.2. The following lemma shows that the set is finite only when  $a$  and  $b$  are rational numbers.

**Lemma 1.**  $\#S(a, b) < \infty \iff a, b \in \mathbb{Q}$ .

*Proof.* As can be seen from the recursion, each interval in  $S(a, b)$  can be written as  $[2^n a - z, 2^n b - z] \cap [s_1, s_2]$ ,  $n \in \mathbb{N}$ ,  $z \in Z$ . For  $n$  large enough, one endpoint of the interval will always be  $s_1$  or  $s_2$ .

Assume  $a$  is irrational, i.e.,  $a \in \mathbb{R} \setminus \mathbb{Q}$ . Set  $a_0 := a$ , and define  $a_i := 2a_{i-1} - k_i$  for a sequence  $\{k_i | k_i \in \mathbb{Z}\}$  such that  $a_i \in [s_1, s_2]$ . The cardinality of  $S(a, b)$  can only be finite if the sequence  $\{a_i\}$  is self-repeating. Then  $a$  has to solve  $2^n a - k = 2^m a - l$ ,  $m, n \in \mathbb{N}$ ,  $k, l \in \mathbb{Z}$ . This means  $a = \frac{k-l}{2^n - 2^m}$ . Clearly,  $a$  cannot be irrational.

Now assume  $a$  is rational. It can be written as  $a = \frac{c}{d}$ ,  $c, d \in \mathbb{Z}$ . Then  $2^n a - z = \frac{2^n c - dz}{d}$  is again a rational number with the same denominator. The cardinality of the set  $\{a_i\}$  is bounded by the total number of such rational numbers in  $[s_1, s_2]$ ,  $d(s_2 - s_1)$ . A similar reasoning for  $b$  proves boundedness of  $\#S(a, b)$ , with a cardinality that is bounded by the sum of  $\#\{a_i\}$  and a similarly defined set  $\#\{b_i\}$ .  $\square$

The lemma shows that the system to be solved for the moments of the scaling function will be the smallest for integers or rational numbers  $a$  and  $b$  with a small denominator. For irrational values, it is infinitely large; obviously the algorithm can then not be applied. However, each number on a computer is represented by a rational number  $a \simeq A2^{-N}$ ,  $A \in \mathbb{Z}$ . The interval sequence will self-repeat after  $N$  recursion steps, since then  $2^N a - z \in \mathbb{Z}$  and  $2^N b - z \in \mathbb{Z}$ . An upper bound for the cardinality of  $S(a, b)$  is then given by  $2N(s_2 - s_1)$ , i.e., the number of iterations times the number of integer shifts of  $2^i a$  and  $2^i b$  that lie in  $[s_1, s_2]$  in each iteration  $i$ . Typically, however, the size of the set is way smaller than this upper bound: this will be illustrated with some examples further.

When the construction of a quadrature rule is required in a time-critical path of a program, it may be desirable to reduce the size of the system to be solved. This can be done by computing the moments using rounded values  $\bar{a} \simeq a$  and  $\bar{b} \simeq b$  that guarantee a lower cardinality  $\#S(\bar{a}, \bar{b})$ . The constructed rule can be seen as a rule for the integration on the interval  $[\bar{a}, \bar{b}]$ . If  $\epsilon := \max(|a - \bar{a}|, |b - \bar{b}|)$  is the roundoff error, the integration error can be estimated by

$$\int_a^{\bar{a}} f(x)\phi(x)dx + \int_{\bar{b}}^b f(x)\phi(x)dx \leq 2\epsilon \max_{x \in [s, \bar{a}] \cup [b, \bar{b}]} (f(x)\phi(x)) = 2\epsilon M.$$

A good estimate for the constant  $M$  is just  $\max(f(a)\phi(a), f(b)\phi(b))$ . Experiments indicate that this error bound is sharp, giving good control of the round-off error.

### 3.5 Convergence of the quadrature rule

Define the integration error  $E_{a,b}[\cdot]$  as

$$E_{a,b}[f(x)] := \int_a^b f(x)\phi(x)dx - Q_{a,b}[f(x)].$$

Table 1: Absolute error for the integration of  $f_i$  with CDF or DB scaling functions.

$[s_1, s_2]$	CDF24 [-1, 1]		CDF24 [-1, 1]		DB2 [-2, 2]	DB3 [-3, 3]
	$f_1$	$f_1$ split	$f_2$	$f_2$ split	$f_2$ split	$f_2$ split
0	$5.6E-2$	$1.5E-2$	$5.6E-1$	$1.5E-2$	$7.1E-2$	$1.4E-2$
-1	$4.5E-4$	$1.4E-4$	$9.9E-2$	$3.0E-4$	$2.1E-4$	$5.4E-6$
-2	$8.1E-8$	$4.6E-9$	$1.5E-2$	$4.4E-8$	$4.3E-11$	$9.6E-13$
-3	$6.7E-16$	$3.3E-16$	$1.5E-1$	$1.6E-15$	$(8.6E-6)$	$(3.0E+9)$
$\sum  w_i $	4.3	2.9	4.3	2.9	78	5266

If  $f(x) \in C^{q+1}[a, b]$  then the first  $q + 1$  terms of the Taylor expansion of  $f$  around a point in  $[a, b]$  are integrated exactly, and the error will depend on the  $(q + 1)$ -th derivative of  $f$ .

To specify this further, let  $P_q(x)$  be the polynomial of degree  $q$  that interpolates the function  $f$  in  $x_1, \dots, x_r$ . Then [4],

$$\forall x \in [s_1, s_2] : \exists \xi(x) \in [s_1, s_2] : f(x) = P_q(x) + e_q(x)$$

with

$$e_q(x) = \frac{\Pi(x)}{(q+1)!} f^{(q+1)}(\xi(x)), \quad \text{and} \quad \Pi(x) = \prod_{i=1}^r (x - x_i).$$

The error is now given by

$$E_{a,b}[f(x)] = E_{a,b}[e_q(x)] = \frac{1}{(q+1)!} \int_a^b \phi(x) \Pi(x) f^{(q+1)}(\xi(x)) dx. \quad (11)$$

Estimates based on this expression are in general rather pessimistic. Moreover, the function  $\xi(x)$  is not known. However, it can be seen from (11) that the asymptotic behaviour is essentially the same as in the rule for smooth functions. The relative error of the quadrature method with the degree of accuracy  $q$  remains  $O(h^{q+1})$ , or  $O(h^r)$ ,  $r = q + 1$ . Yet, this order is to be evaluated for a smaller value of  $h$ , since  $|b - a| < |s_2 - s_1|$ .

For a smooth function  $f$ , this may not be the best solution. A more accurate result can be obtained by computing coefficients on a finer scale, and using the refinement equation to obtain values for the rougher scale. This would lead to an error of the order  $O(h^{2r})$ . If the function is only piecewise smooth however, we can split the interval  $[s_1, s_2]$  into pieces that correspond to the smooth parts of  $f$ . The convergence is then not adversely affected by discontinuities of  $f$ , or of any of its derivatives.

**Example 1.** We consider the functions  $f_1(x) := \cos(2x) + \sin(3x)$  and  $f_2(x) := \cos(|2x|) + \sin(|3x|)$ . We compare the integration rule of [13], discussed in Section 2, with the integration rule discussed in Section 3. The parameter  $s$  determines the number of abscissae  $r$  used in the interval  $[s_1, s_2]$  for the first method:  $r = 2^{-s}(s_2 - s_1) + 1$ . For the second method, the interval is split at the origin, and  $r$  abscissae are used in both intervals. Hence, there is a total of  $2r$  abscissae.

The values given in Table 1 represent the absolute error for the integrals  $\int_a^b f_i(x) \phi(x) dx$ . The values in the last row represent the maximum sum of the absolute values of the weights that

were used in the corresponding column. We consider three different scaling functions. Cohen-Daubechies-Feauveau (CDF) wavelets are biorthogonal wavelets with a finite support [5]. The primal scaling function is a B-spline, and the primal wavelet is therefore also piecewise polynomial. The numbers 2 and 4 in the notation CDF24 represent the order of the primal and dual wavelets respectively. Daubechies (DB) wavelets are orthogonal wavelets with finite support, and were first constructed in [8]. The wavelets of order 2 and 3 have very low regularity on each subinterval of their support.

The first two columns show that splitting the interval (and thus doubling the points) is not very useful for the case of a smooth function. The result is only slightly better, and does not compensate sufficiently for the extra effort. For function  $f_2$  however, which is not smooth at the origin, the regular method shows no convergence. The second method converges rapidly to almost machine precision. Similar results are obtained for the scaling functions of two different Daubechies wavelets. The values corresponding to  $s = -3$  in the last two columns indicate the presence of a large error, due to a number of abscissae greater than 30 (respectively 33 and 49). This is an illustration of the instability and poor conditioning problem mentioned in Section 3.1 and discussed in [13]. The values are given between parentheses. In order to get better accuracy results, the subintervals would have to be split into a larger number of smaller intervals.

The values in the last row, i.e., the maximum sum of the weights, are very moderate, even for the Daubechies scaling functions that switch sign. This indicates good stability properties of the constructed rules.

**Example 2.** We now look at a different example to illustrate the size of  $S(a, b)$ . The computation of the moments on the interval  $[\pi/10, \pi/4]$  for the CDF scaling functions, leads to a system with 195 unknowns with a representation in double precision. The condition number of the system to be solved in level  $l = 0$  of the algorithm is only 47. It is smaller in the next levels corresponding to the higher order moments. Rounding the interval boundaries to the nearest multiple of  $2^{-16}$  reduces the size of the system to 57 unknowns, and a maximum condition number of 27. The upper bound on the number of unknowns here is  $2N(s_2 - s_1) = 64$ . The error induced on the integration is  $5E - 6$  for the function  $f(x) = 1$ .

This example illustrates the trade-off between computation time for the construction of the rule, and the round-off error for intervals with irrational numbers as endpoints. The error can always be made as small as needed however.

Having described the convergence as a function of  $q$ , it is also of interest to consider the convergence as a function of  $n$ , which is the scale of the scaling function  $\phi_{nk}(x)$  in the integrand. This is because in most applications we would like to match the integration error of the numerical quadrature to the discretisation error of the corresponding wavelet approximation on a given scale. The values we want to compute using rule (6) are given by

$$d_{nk} = \int_{a_n}^{b_n} f(x)\phi_{nk}(x) dx.$$

For the case of smooth functions in Section 2, seeing that  $h$  is proportional to  $2^{-n}$ , one obtains the error estimate  $O(2^{-n(q+1)})$  that was mentioned in Section 2.2. Here, however, we need to consider two cases:

1. the endpoints of the integration interval change with  $n$ , such that the same part of the scaling function is covered on each scale. In this case,  $a_n = 2^{-n}(a + k)$ .

Table 2: Relative error for the integration of  $f_1$  on the interval  $[a_n, b_n]$  for DB3 wavelets.

q	$n = 0$	$n = 1$	$n = 2$	$n = 3$
0	$1.1E - 1$	$2.6E - 1$	$1.8E - 1$	$1.0E - 2$
2	$4.0E - 3$	$6.6E - 4$	$9.6E - 5$	$1.3E - 5$
4	$2.1E - 5$	$8.9E - 7$	$3.2E - 8$	$1.1E - 9$
6	$8.1E - 8$	$8.6E - 10$	$8.7E - 12$	$9.9E - 14$

2. the endpoints remain fixed as  $n$  increases, i.e.,  $a_n = a$ .

The first case occurs, e.g., when we would like to increase the accuracy of the integration, by splitting the support of the scaling function in a finite number of subintervals, regardless of the scale. The second case occurs when  $f$  has a discontinuity (in a derivative) at a fixed point  $a$  or  $b$ . We can see that the convergence in the first case will be asymptotically similar to the case of smooth functions, i.e.,  $O(2^{-n(q+1)})$ , albeit with a smaller constant since the integration interval is also smaller. In the second case, the error will still behave like  $O(h^{q+1})$ , but  $h$  does not scale as  $2^{-n}$  initially. That only happens when the scaling parameter  $n$  becomes large enough, such that the support of the scaling function is contained entirely within the fixed interval  $[a, b]$ . This reduces the integration problem to the setting in Section 2, where the integration interval does not have to be split into parts.

**Example 3.** To illustrate the above discussion, consider again the function  $f_1(x)$ . Table 2 shows the relative error for the case of Daubechies wavelets with  $k = 0$ .

The error for fixed  $q$  decreases with the expected factor  $2^{-(q+1)}$ . For fixed  $n$ , we expect a convergence rate of  $2^{-n(q+2+1)+n(q+1)} = 2^{2n}$ . For increasing  $n$ , the convergence rates in Table 2 indeed approximately improve by a factor of 4 from left to right in each column.

## 4 An integration rule for functions with singularities

### 4.1 Functions with a known singularity

The method can be extended to work for functions with an integrable singularity, e.g.,  $s(t) = |t|^\alpha$ , for  $-1 < \alpha < 0$  or  $s(t) = \log(|t|)$ . First we will assume that the singularity of  $f$  is known analytically and can be subtracted, i.e.,  $f(x) = p(x) + q(x)s(x - x')$ ,  $x' \in [s_1, s_2]$ , where  $p(x)$  is a non-singular function. We will develop a quadrature rule  $Q^s[\cdot]$  such that

$$\int_a^b f(x)\phi(x)dx = \int_a^b p(x)\phi(x)dx + \int_a^b q(x)s(x - x')\phi(x)dx \simeq Q[p] + Q^s[q],$$

with  $Q[p]$  being quadrature rule (3).

We demand for quadrature rule  $Q^s[\cdot]$  an exact integration of the functions  $s(x - x')P_l(x)$ , with  $P_l(x)$  from (5). The required moments are in their most general form given by

$$M_{a,b}^{l,m} := \int_a^b T_l\left(\frac{x + \lambda_1}{L_1}\right) s(x - m)\phi(x)dx.$$

First we discuss how to deal with the singularity. Using the refinement relation, we have

$$M^{0,m} := \int_{-\infty}^{+\infty} s(x-m)\phi(x)dx = \frac{\sqrt{2}}{2} \sum_k h_k \int_{-\infty}^{+\infty} s\left(\frac{x+k}{2} - m\right)\phi(x)dx. \quad (12)$$

Hence, in the right hand side integrals, the singularity has been shifted. For the algebraic singularity, the shifted singularity can be rewritten in the original notation  $s(x-m)$ ,

$$\left|\frac{x+k-2m}{2}\right|^\alpha = |x-(2m-k)|^\alpha 2^{-\alpha}$$

and, similarly, for the logarithmic singularity,

$$\log\left(\left|\frac{x+k-2m}{2}\right|\right) = \log(|x-(2m-k)|) - \log(2).$$

For  $s(t) = \log(|t|)$ , relation (12) becomes

$$M^{0,m} = \frac{\sqrt{2}}{2} \sum_k h_k M^{0,2m-k} - \log(2) \frac{\sqrt{2}}{2} \sum_k h_k,$$

while for  $s(t) = |t|^\alpha$  we find

$$M^{0,m} = \frac{\sqrt{2}}{2} 2^{-\alpha} \sum_k h_k M^{0,2m-k}.$$

Recursive application of the above expressions for different values of  $m$  leads again to a set of linear equations in the unknown moments  $M^{0,m^{(i)}}$ . The parameter  $m^{(i)} = 2m^{(i-1)} - k$  with  $m^{(0)} = m$  grows in principle without bound. Yet, if it is large enough the integral is no longer singular. The corresponding moments can then be computed by using the techniques of Section 2. For accurate computations, it is better to also include the nearly singular moments as unknowns in the set of equations. Good results were obtained by including the moments for all intervals that satisfy  $\text{dist}(m, [a, b]) < 1$ , i.e., when the distance of the singularity to the integration interval is of the same order as the size of the interval, which is  $O(1)$  on scale  $n = 0$ .

Combined with the approach of splitted intervals, we find a linear equation for each moment. For example, for the logarithmic singularity, we have an equation of the following type,

$$M_{a,b}^{l+1,m} = 2^{-(l+\frac{3}{2})} \sum_k h_k \sum_{i=0}^{l+1} w_i^{(l+1)}(k) (M_{2a-k,2b-k}^{i,2m-k} - \log(2) M_{2a-k,2b-k}^{l+1}). \quad (13)$$

The moments can be computed for each value of  $l$  successively, starting from  $l = 0$  and the known partial moments  $M_{a,b}^l$ .

Again, the size of the system to be solved is finite only under certain conditions. Define  $S(a, b, m)$  as the set of intervals and singularity locations corresponding to the moments found by applying (13) recursively, for which  $\text{dist}(m, [a, b]) < 1$ . The cardinality of this set is bounded only if  $m$  is a rational number.

**Lemma 2.**  $\#S(a, b, m) < \infty \iff m \in \mathbb{Q}$ .

Table 3: Absolute error for the quadrature approximation of the inner product of  $\log(|x|)f_i(x)$  with CDF24, DB2, or DB3 scaling functions.

	CDF24	CDF24	DB2	DB3
r	$\log( x )f_1$	$\log( x )f_2$	$\log( x )f_2$	$\log( x )f_2$
3	$4.1E-2$	$1.6E-2$	$8.9E-1$	$1.3E-1$
5	$2.8E-4$	$7.2E-4$	$1.4E-1$	$6.0E-2$
9	$1.8E-9$	$1.5E-7$	$5.0E-4$	$3.2E-3$
13	$1.6E-13$	$6.3E-12$	$4.2E-7$	$1.5E-5$
17	$5.5E-15$	$8.9E-15$	$1.2E-10$	$2.1E-8$
$\sum  w_i $	1.5	117	263	94

*Proof.* The intervals  $[2^na - z, 2^nb - z] \cap [s_1, s_2]$  with corresponding singularity  $2^m - z$  are in  $S(a, b, m)$  only if  $2^m - z \in [s_1 - 1, s_2 + 1]$ . Using the same line of reasoning as in Lemma 1 leads to the condition  $m \in \mathbb{Q}$  in order for this set to be finite.

Conditions on  $a$  and  $b$  are not required if  $a, b \neq m$ , since for  $n$  large enough we have that  $2^na - z - (2^nm - z) = 2^n(a - m) > s_2 - s_1 + 1$ . This means that for any  $2^nm - z \in [s_1 - 1, s_2 + 1]$ ,  $[2^na - z, 2^nb - z] \cap [s_1, s_2] = [s_1, s_2]$ .  $\square$

In a time critical code path,  $m$  can also be rounded to a near rational number. The error made is given by

$$\int_a^b f(x)\phi(x)(\log|x - m| - \log|x - \bar{m}|)dx \leq M(m - a)\log|a - m| + (a - \bar{m})\log|a - \bar{m}| + (b - m)\log|b - m| + (\bar{m} - b)\log|b - \bar{m}|, \quad (14)$$

with  $M = \max f(x)\phi(x)$ . We have  $x \log(x) - (x + \epsilon) \log(x + \epsilon) \approx \epsilon(\log(x) + 1)$ . Using this expression, we see that when  $m = a$  or  $m = b$ , expression (14) has order  $O(\epsilon \log(\epsilon))$ . Otherwise it has order  $O(\epsilon)$ .

The error  $E^s[f(x)] := \int_a^b \log(x - x')f(x)\phi(x)dx - Q^s[f(x)]$  is now given by

$$E^s[f(x)] = E^s[e_q(x)] = \frac{1}{(q+1)!} \int_a^b \log(x - x')\phi(x)\Pi(x)f^{(q+1)}(\xi(x))dx.$$

This leads, asymptotically, to the same relative error  $O(h^{q+1})$ , with  $h = 2^{-n}$  or smaller depending on the size of  $[a, b]$ , for scaling functions on scale  $n$ .

**Example 4.** Table 3 lists the absolute error of the approximation obtained by the quadrature method for two singular functions,  $\log(|x|)f_1(x)$  and  $\log(|x|)f_2(x)$ , with  $f_1(x)$  and  $f_2(x)$  as defined in Example 1. In this example, we compare different scaling functions for a fixed number of abscissae  $r$ . For the second test function, the interval  $[s_1, s_2]$  is again split at the origin in order to cope with the discontinuity of the derivatives of  $f_2(x)$ .

It is clear from Table 3 that the results converge rapidly. Note that the Daubechies wavelets require a larger number of abscissae for the same absolute error, due to their wider support. The largest system that was needed to compute the required moments for this table had only dimension 6.

Table 4: Absolute error for the quadrature approximation of the inner product with  $\log(|x|)f_i(x)$  via rules with known (A) and unknown (B) singularity.

$q$	CDF24 $\log( x )f_1$		DB2 $\log( x )f_1$		CDF24 $\log( x )f_2$	
	A	B	A	B	A	B
1	$1.9E-0$	$2.0E-2$	$2.1E-0$	$1.1E-0$	$8.3E-1$	$3.5E-1$
3	$4.8E-2$	$8.2E-4$	$8.2E-1$	$4.0E-1$	$2.8E-2$	$2.5E-2$
7	$3.4E-6$	$3.0E-6$	$3.2E-3$	$1.6E-3$	$7.6E-6$	$9.5E-5$
11	$4.4E-11$	$9.4E-11$	$2.1E-6$	$7.5E-7$	$5.0E-10$	$5.9E-8$
15	$2.0E-15$	$5.8E-14$	$3.7E-10$	$9.1E-11$	$1.4E-14$	$(9.7E-7)$
$\sum  w_i $	1.7	1625	2.0	4180	67	$9.9E8$

## 4.2 Functions with an unknown singularity

In some cases, one does not wish to substract the singularity explicitly, as in Section 4.1, and use a separate rule for the smooth and the singular parts. It is still possible to compute an efficient quadrature rule in this situation, by requiring exactness of the integration result for the functions  $T_l(x)$  and  $s(x-x')T_l(x)$ . The right hand side of the resulting set of equations is set up in a similar way as in Section 3.2 and Section 4.1, based on the moments  $M_{a,b}^l$  and  $M_{a,b}^{l,m}$ . Obviously, the number of abscissae necessary to obtain a degree of accuracy doubles, as does the size of the system, i.e.,  $r = 2q + 2$  if the above functions are used for  $l = 0, 1, \dots, q$ . Unfortunately, the matrix of the resulting system may become ill-conditioned again. The method is usable however in practice, if the required degree of accuracy for the application is not too high. If the unknown functions  $p(x)$  and  $q(x)$  are smooth, as in the example below, conditioning does not pose a significant problem. For the function  $\log(|x|)f_2(x)$ , with  $f_2(x)$  defined in Example 1, we could only obtain good rules with an order of up to 11 after we split the interval at the origin. Rules with higher accuracy require different basis functions, to avoid the ill-conditioning, or the use of higher precision arithmetic.

**Example 5.** We consider functions with known singularity, and compare the results with those obtained with the previous method. All calculations were performed in double precision; no additional measures were taken. The results are given in Table 4.

For function  $\log(|x|)f_1(x)$ , there is for most values of  $q$  no significant difference between the two methods. The method for unknown singularity requires two times the number of abscissae however (i.e.,  $r = q + 1$  for method A and  $r = 2q + 2$  for method B). That explains why in some rows the result is actually better. The degree of accuracy is the same in both cases.

The last two columns illustrate the problem of ill-conditioning. The convergence of method B stops after the fourth row. Depending on the application, the error may however be already small enough at that point.

## 5 A two-dimensional tensor-product rule example

When tensor product wavelets are used for the multiresolution analysis of two-dimensional functions, one is faced with double integrals of the form

$$c_{(nk),(n'k')} := \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y) \phi_{n,k}(x) \phi_{n',k'}(y) dx dy.$$

These can be handled by repeated one-dimensional integration, using the techniques of the previous sections. If the function  $f$  is discontinuous in some points, or has discontinuities in a derivative, the inner integration should be done with piecewise integration. If for instance  $f(x, y) = g(|x - y|)$ , then for each  $y_i$  in the outer integration, two distinct rules can be applied on the intervals  $[s_1, y_i]$  and  $[y_i, s_2]$ . In general however, if  $\phi$  is not very smooth, the result of  $g(y) = \int_{-\infty}^{+\infty} f(x, y) \phi(x) dx$  is not very smooth either. As the convergence of the outer integration depends on the smoothness of  $g$ , convergence will suffer. This problem requires then a truly two-dimensional quadrature rule with the product  $\phi(x)\phi(y)$  as a weight function. The construction of such rules is beyond the scope of the present paper. Instead, we will consider an important special case, for scaling functions that are piecewise smooth (e.g., the popular CDF-wavelets that are piecewise polynomials [5]). We illustrate this case with an application from the field of electromagnetics.

The Boundary Element Method is a numerical technique for solving integral equations [11]. It is a Galerkin scheme, where the test and trial functions are called *boundary elements*. Consider an integral operator  $(Tf)(x) = \int_S K(x, y) f(y) dy$  over a domain  $S \subset \mathbb{R}$ . The use of a Galerkin scheme with wavelet scaling functions as both test and trial functions, leads to a matrix representation of the operator, with elements of the form

$$\int_S \int_S K(x, y) \phi_{jk}(x) \phi_{j'k'}(y) dx dy.$$

When using wavelets in this way, the matrix can easily be compressed and a solution to an integral equation can be found efficiently. This technique was pioneered in [3]. See also [12] for a broader introduction to the setting described below.

In these applications, the kernel function  $K(x, y)$  is typically singular along the diagonal  $x = y$ , e.g.,  $K(x, y) \sim \log(|x - y|)$ . As a concrete example, we consider the Helmholtz operator that arises in scattering theory [6],

$$(Tf)(x) = \int_S \frac{i}{4} H_0^{(1)}(k|x - y|) f(y) dy.$$

Parameter  $k$  is the wavenumber and determines the frequency of the electromagnetic waves. The function  $H_0^{(1)}(z)$  is the Hankel function of the first kind and order zero. It is logarithmic in the origin [1],

$$H_0^{(1)}(z) \sim \frac{2i}{\pi} \log\left(\frac{z}{2}\right) J_0(z).$$

The smoothness of  $g(y) = \int_S K(x, y) \phi(x) dx$  depends on the smoothness of  $\phi$  and on the order  $r$  of the operator. In particular,  $T : H^s \rightarrow H^{s+1}$ , which means the Helmholtz operator has order  $-1$ . The function given by  $\int_a^b K(x, y) f(x) dx$ , for smooth functions  $f$  and arbitrary intervals  $[a, b] \subset S$ , is again smooth on the open interval  $(a, b)$  [11].

Assume the hat function  $\phi(x) := 1 - |x|$  is used as the scaling function for discretizing the Helmholtz operator. A possible matrix element is then given by

$$\int_{-1}^1 \int_{-1}^1 \frac{i}{4} H_0^{(1)}(k|x-y|) \phi(x) \phi(y) dx dy. \quad (15)$$

For the inner integration in  $x$ , the integral is split according to the smooth intervals of  $\phi$ . We also split at the singular point on the diagonal  $x = y$ , because  $J_0(k|x-y|)$  is not smooth there. We can use the quadrature rules for singular functions with a known singularity. The result is accurate, but not very smooth, since the hat function is not very regular. An alternative approach is to also split the integration in  $y$  according to the smooth intervals of  $\phi(y)$ . Then, one has

$$\int_{-1}^1 \int_{-1}^1 \cdot = \int_{-1}^0 \left( \int_{-1}^y \cdot + \int_y^0 \cdot + \int_0^1 \cdot \right) + \int_0^1 \left( \int_{-1}^0 \cdot + \int_0^y \cdot + \int_y^1 \cdot \right).$$

Every integrand is now a smooth function, and the result of every inner integration is smooth except possibly at the boundary points  $-1$ ,  $0$  and  $1$ . We can apply the quadrature rules for functions with unknown singularity as in Section 4.2. Table 5 lists the relative error for different values of the wavenumber  $k$ , and the number of abscissae  $r$  in one dimension. Large values of the wave number  $k$  introduce oscillations in the kernel function  $K(x, y)$ . As can be seen in the table, the relative error is larger for the oscillatory integrands, but convergence is equally good in all cases.

Table 5: Relative error of the quadrature approximation of integral (15) for different values of the wavenumber  $k$ .

$r$	$k = 2$	$k = 10$	$k = 60$
2	$1.7E - 4$	$4.7E - 3$	$1.4E - 1$
4	$1.3E - 6$	$5.2E - 6$	$7.7E - 4$
6	$1.1E - 8$	$1.5E - 8$	$2.4E - 6$
8	$1.2E - 9$	$1.3E - 9$	$9.9E - 9$

## 6 Concluding remarks

We introduced a family of quadrature rules to compute the inner product of a scaling function with a piecewise smooth function, or with a singular function. The examples show that these rules have similar convergence rates as known rules for the integration of smooth functions, even for scaling functions with very low regularity. The rules don't require the evaluation of the scaling function, neither for their construction, nor for their use in computations. This makes them suitable for general purpose routines.

One application was explored from the field of electromagnetics. The presented method for the inner product with a piecewise smooth singular function leads to fast and accurate results.

## Acknowledgements

The authors would like to thank Ronald Cools from the Computer Science Department of KU Leuven for many insightful discussions, and the anonymous referees for many constructive

remarks.

## References

- [1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, New York, 1965.
- [2] A. Barinka, T. Barsch, S. Dahlke, and M. Konik. Some remarks on quadrature formulas for refinable functions and wavelets. *Z. Angew. Math. Mech.*, 81(12):839–855, 2001.
- [3] G. Beylkin, R. R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Comm. Pure Appl. Math.*, 44:141–183, 1991.
- [4] R.L. Burden and J.D. Faires. *Numerical Analysis*. PWS-Kent Publishing Company, Boston, 4th edition, 1989.
- [5] A. Cohen, I. Daubechies, and J.-C. Feauveau. Biorthogonal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, 55:485–560, 1992.
- [6] D. Colton and R. Kress. *Integral Equation Methods in Scattering Theory*. Wiley, New York, 1983.
- [7] W. Dahmen and Charles A. Micchelli. Using the refinement equation for evaluating integrals of wavelets. *SIAM J. Numer. Anal.*, 30(2):507–537, 1993.
- [8] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, 41:909–996, 1988.
- [9] Philip J. Davis and Philip Rabinowitz. *Methods of Numerical Integration*. Academic Press Inc., 1984.
- [10] W. Gautschi, L. Gori, and F. Pitolli. Gauss quadrature for refinable weight functions. *Appl. Comput. Harmon. Anal.*, (8):249–257, 2000.
- [11] W. Hackbusch. *Integral Equations: Theory and Numerical Treatment*, volume 120 of *Internat. Series of Numer. Math.* Birkhäuser Verlag, Basel, Switzerland, 1995.
- [12] D. Huybrechs, J. Simoens, and S. Vandewalle. A note on wave number dependence of wavelet matrix compression for integral equations with oscillatory kernel. *J. Comput. Appl. Math.*, 172(2):233–246, 2004.
- [13] W. Sweldens and R. Piessens. Quadrature formulae and asymptotic error expansions for wavelet approximations of smooth functions. *SIAM Journal on Numerical Analysis*, 31(4):1240–1264, 1994.

## A A transformation rule for Chebyshev polynomials

One way to write a Chebyshev polynomial of the first kind in terms of Chebyshev polynomials of lower order and with a different shift and scaling, is given by the formula

$$T_n\left(\frac{x+\lambda_1}{L_1}\right) = \sum_{i=0}^n 2^{-n} w_i^{(n)} T_i\left(\frac{x+\lambda_2}{L_2}\right). \quad (16)$$

The coefficients  $w_i^{(n)}$  can be computed with an iterative scheme. The following derivation and algorithm is a slight modification from the work in [13]. The formula (16) is somewhat more general than the corresponding one in [13], and was needed in Section 3.3.

The derivation makes use of the well-known recurrence relation for Chebyshev polynomials of the first kind,  $T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x)$ . This recurrence relation is first applied to the left hand side of equation (16), after which  $T_{n-1}$  and  $T_{n-2}$  are replaced by using an expression similar to the right hand side of (16). This is followed by a rearrangement of the right hand side terms. More precisely,

$$\begin{aligned} T_n\left(\frac{x+\lambda_1}{L_1}\right) &= 2\frac{x+\lambda_1}{L_1} \sum_{i=0}^{n-1} 2^{-(n-1)} w_i^{(n-1)} T_i\left(\frac{x+\lambda_2}{L_2}\right) \\ &\quad - \sum_{i=0}^{n-2} 2^{-(n-2)} w_i^{(n-2)} T_i\left(\frac{x+\lambda_2}{L_2}\right) \\ &= 2\frac{L_2}{L_1} \sum_{i=0}^{n-1} 2^{-(n-1)} w_i^{(n-1)} \frac{1}{2} \left( T_{i+1}\left(\frac{x+\lambda_2}{L_2}\right) + T_{|i-1|}\left(\frac{x+\lambda_2}{L_2}\right) \right) \\ &\quad + 2 \left( 2\frac{\lambda_1}{L_1} - 2\frac{\lambda_2}{L_1} \right) \sum_{i=0}^{n-1} 2^{-(n-1)} w_i^{(n-1)} T_i\left(\frac{x+\lambda_2}{L_2}\right) \\ &\quad - 4 \sum_{i=0}^{n-2} 2^{-(n-2)} w_i^{(n-2)} T_i\left(\frac{x+\lambda_2}{L_2}\right) \\ &= 2^{-n} \left( \frac{2L_2}{L_1} \sum_{i=1}^n w_{i-1}^{(n-1)} T_i\left(\frac{x+\lambda_2}{L_2}\right) + \frac{2L_2}{L_1} \sum_{i=-1}^{n-2} w_{i+1}^{(n-1)} T_{|i|}\left(\frac{x+\lambda_2}{L_2}\right) \right. \\ &\quad + \frac{4}{L_1} (\lambda_1 - \lambda_2) \sum_{i=0}^{n-1} w_i^{(n-1)} T_i\left(\frac{x+\lambda_2}{L_2}\right) \\ &\quad \left. - 4 \sum_{i=0}^{n-2} w_i^{(n-2)} T_i\left(\frac{x+\lambda_2}{L_2}\right) \right). \end{aligned}$$

The coefficients  $w_i^{(n)}$  can be found by identifying this expression with (16). An algorithm that computes those coefficients is presented below.

```

 $w_0^{(0)} \leftarrow 1$ 
 $w_0^{(1)} \leftarrow 2(\lambda_1 - \lambda_2)/L_1$ 
 $w_1^{(1)} \leftarrow 2L_2/L_1$ 
 $w_0^{(2)} \leftarrow (-16\lambda_1\lambda_2 - 4L_2^2 + 4L_1^2 - 8\lambda_2^2 - 8\lambda_1^2)/L_1^2$ 
 $w_1^{(2)} \leftarrow 16L_2(\lambda_1 - \lambda_2)/L_1^2$ 
 $w_2^{(2)} \leftarrow 4L_2^2/L_1^2$ 
for  $p \leftarrow 2, 3, \dots$  do
   $w_0^{(p+1)} \leftarrow 4(\lambda_1 - \lambda_2)w_0^{(p)}/L_1 + 2L_2w_1^{(p)}/L_1 - 4w_0^{(p-1)}$ 
   $w_1^{(p+1)} \leftarrow 4L_2w_0^{(p)}/L_1 + 4(\lambda_1 - \lambda_2)w_1^{(p)}/L_1 + 2L_2w_2^{(p)}/L_1 - 4w_1^{(p-1)}$ 
  for  $i \leftarrow 2, 3, \dots, p-1$  do
     $w_i^{(p+1)} \leftarrow 2L_2w_{i-1}^{(p)}/L_1 + 4(\lambda_1 - \lambda_2)w_i^{(p)}/L_1 + 2L_2w_{i+1}^{(p)}/L_1 - 4w_i^{(p-1)}$ 
  end for
   $w_p^{(p+1)} \leftarrow 2L_2w_{p-1}^{(p)}/L_1 + 4(\lambda_1 - \lambda_2)w_p^{(p)}/L_1$ 
   $w_{p+1}^{(p+1)} \leftarrow 2L_2w_p^{(p)}/L_1$ 
end for

```

**Algorithm 1:** Computation of the coefficient  $w_i^{(n)}$  in formula (16).