Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 195 (2008) 326–345
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

Numerical stability of fast computation algorithms of Zernike moments

G.A. Papakostas [a,*], Y.S. Boutalis [a], C.N. Papaodysseus [b], D.K. Fragoulis [b]

[a] Democritus University of Thrace, Department of Electrical and Computer Engineering, 67100 Xanthi, Greece
[b] National Technical University of Athens, School of Electrical and Computer Engineering, 15773 Athens, Greece

Abstract

A detailed, comparative study of the numerical stability of the recursive algorithms, widely used to calculate the Zernike moments of an image, is presented in this paper. While many papers, introducing fast algorithms for the computation of Zernike moments have been presented in the literature, there is not any work studying the numerical behaviour of these methods. These algorithms have been in the past compared to each other only according to their computational complexity, without been given the appropriate attention, as far as their numerical stability is concerned, being the most significant part of the algorithms' reliability. The present contribution attempts to fill this gap in the literature, since it mainly demonstrates that the usefulness of a recursive algorithm is defined not only by its low computational complexity, but most of all by its numerical robustness.

This paper exhaustively compares some well known recursive algorithms for the computation of Zernike moments and sets the appropriate conditions in which each algorithm may fall in an unstable state. The experiments show that any of these algorithms can be unstable under some conditions and thus the need to develop more stable algorithms is of major importance.

Keywords: Zernike moments; Recursive algorithm; Finite precision error; Numerical stability

1. Introduction

Although a long time has passed, since the first introduction of orthogonal moments in image processing by Teague [1], the scientific interest in the usage of moments in engineering life is still increased. This happens due to their ability to uniquely describe a signal and thus they are used as discriminative features in image representation and pattern recognition applications.

There are some research topics about orthogonal moments in which scientists focus their attention lately. Many researchers all over the world try to develop fast algorithms [2] that accelerate the computation of orthogonal moments and make their hardware implementation an easy task. Most of the fast algorithms

* Corresponding author.
E-mail addresses: gpapakos@ee.duth.gr (G.A. Papakostas), ybout@ee.duth.gr (Y.S. Boutalis), cpapaod@cs.ntua.gr (C.N. Papaodysseus), dfrag@mail.ntua.gr (D.K. Fragoulis).

0096-3003/$ - see front matter
doi:10.1016/j.amc.2007.04.110

presented until now, are making use of recursive equations that permit the computation of a high order moment kernels by using kernels of lower orders. This methodology has proved to be efficient, especially when an entire set of moments is necessary to be calculated.

Moreover, since orthogonal moments have as kernel functions orthogonal polynomials, which construct orthogonal basis, are very useful to uniquely represent patterns in pattern recognition tasks. For these applications some very promising attempts [3–5] to embody scale, rotation and translation invariances in the computation of orthogonal moments have been performed.

Recently, the disadvantages of orthogonal moments having continuous orthogonal polynomials as kernel functions have been studied and more numerically accurate orthogonal moments, with discrete polynomials [6–8] are introduced as alternative to the continuous ones. It has been proved that the moments with discrete polynomials are more suitable in image processing, since the domain of an image is discrete.

However, although a brief study about the accuracy of the orthogonal moments in respect to some approximation errors, due to the transformation from the continuous space to the discrete [9] and the accuracy of the moments into scale, translation and rotation transformations [10,11] have been done, there is not any work which deals with the finite precision errors presented in the computation of orthogonal moments. The occurrence of finite precision errors is of major importance, especially in the fast recursive algorithms since a possible error in a step of the algorithm may be accumulated iteration by iteration, by resulting to unreliable outcomes.

A first investigation of the mechanisms that produce and propagate finite precision errors in Zernike moments computation, using the well known "q-recursive" algorithm, has been successfully performed and presented by the authors in [12]. In that work a detailed numerical analysis of the way the finite precision errors are being generated and the appropriate conditions under which these errors occurred were discussed.

The present paper comes to complete the previous work [12] by comparing some very popular recursive algorithms widely used in Zernike moments computation, in terms of finite precision errors. This comparative study from a different point of view, the algorithms' numerical robustness, in conjunction with the comparison already been done in [2] in respect to their computational complexity, constitutes a complete study of the computational behaviour that each recursive algorithm presents.

By keeping in mind the above objectives the paper is organized by presenting the description of each recursive algorithm in section one, by analyzing the algorithms' performance, according to the finite precision errors being generated and finally by discussing the performance of each algorithm by means of their numerical stability.

2. Computing the Zernike moments

Zernike moments (ZMs) are the most widely used family of orthogonal moments due to their properties, of being invariant to an arbitrary rotation of the object that they describe and that a perfect reconstruction of an image from its moments is possible. They are used, after making them invariant to scale and translation, as object descriptors in pattern recognition applications [13–19] and in image retrieval tasks [20,21] with considerable results.

The introduction of ZMs in image analysis was made by Teague [1], using a set of complex polynomials, which form a complete orthogonal set over the interior of the unit circle $x^2 + y^2 = 1$. These polynomials [13,14] have the form

$$V_{pq}(x,y) = V_{pq}(r,\theta) = R_{pq}(r)\exp(jq\theta), \tag{1}$$

where p is a non-negative integer and q positive and negative integers subject to the constraints $p - |q|$ even and $|q| \leqslant p$, r is the length of vector from the origin $(\bar{x},\bar{y})$ to the pixel (x,y) and θ the angle between vector r and x axis in counter-clockwise direction. $R_{pq}(r)$, are the Zernike radial polynomials [22] in (r,θ) polar coordinates defined as

$$R_{pq}(r) = \sum_{s=0}^{\frac{p-|q|}{2}} (-1)^s \cdot \frac{(p-s)!}{s!\left(\frac{p+|q|}{2}-s\right)!\left(\frac{p-|q|}{2}-s\right)!} r^{p-2s}. \tag{2}$$

Note that $R_{p,-q}(r) = R_{pq}(r)$

The polynomials of Eq. (1) are orthogonal and satisfy the orthogonality principle

$$\int\int_{x^2+y^2\leqslant 1} V^*_{nm}(x,y)\cdot V_{pq}(x,y)\,\mathrm{d}x\,\mathrm{d}y = \frac{\pi}{n+1}\delta_{np}\delta_{mq},$$

where $\delta_{\alpha\beta}=1$ for $\alpha=\beta$ and $\delta_{\alpha\beta}=0$ otherwise, is the *Kronecker* symbol.

The Zernike moment of order p with repetition q for a continuous image function $f(x,y)$, that vanishes outside the unit disk is

$$Z_{pq} = \frac{p+1}{\pi}\int\int_{x^2+y^2\leqslant 1} f(x,y)V^*_{pq}(r,\theta)r\,\mathrm{d}r\,\mathrm{d}\theta. \tag{3}$$

For a digital image, the integrals are replaced by summations [13,14] to get

$$Z_{pq} = \frac{p+1}{\pi}\sum_x\sum_y f(x,y)V^*_{pq}(r,\theta), \quad x^2+y^2\leqslant 1. \tag{4}$$

Suppose that one knows all moments Z_{pq} of $f(x,y)$ up to a given order $p_{\max}$. It is desired to reconstruct a discrete function $\hat{f}(x,y)$ whose moments exactly match those of $f(x,y)$ up to the given order $p_{\max}$. Zernike moments are the coefficients of the image expansion into orthogonal Zernike polynomials as can be seen in the following reconstruction equation:

$$\hat{f}(x,y) = \sum_{p=0}^{p_{\max}}\sum_q Z_{pq}V_{pq}(r,\theta) \tag{5}$$

with q having similar constraints as in (1). Note that as $p_{\max}$ approaches infinity $\hat{f}(x,y)$ will approach $f(x,y)$.

The method that computes the Zernike moments and uses Eq. (2) to evaluate the Zernike polynomials, is called *Direct Method*.

As can be seen from Eq. (2) there are a lot of factorial computations, operations that consume too much computer time. For this reason, as it has already been discussed in the introduction, recursive algorithms for the computation of the radial polynomials (2) have been developed [2].

The most well known recursive algorithms for Zernike moments computation, are described in the next sections and their main features are discussed.

2.1. Kintner's algorithm

Kintner was the first who studied the properties of the Zernike polynomials [23] and introduced a recurrent relation [24]. In the following the recursive algorithm for Zernike polynomials computation, proposed by Kintner, is described.

Algorithm

- $p=q$ Direct Method using Eq. (2)
- $p-q=2$ Direct Method using Eq. (2)
- otherwise $R_{pq}(r) = \dfrac{(K_2r^2+K_3)R_{(p-2)q}(r)+K_4R_{(p-4)q}(r)}{K_1}$, with (6a)

$$K_1 = \frac{(p+q)(p-q)(p-2)}{2}, \quad K_2 = 2p(p-1)(p-2),$$

$$K_3 = -q^2(p-1)-p(p-1)(p-2), \quad K_4 = \frac{-p(p+q-2)(p-q-2)}{2}. \tag{6b}$$

As it can be seen in the above equations, Kintner's algorithm cannot be applied in cases where $p=q$ and $p-q=2$, thus the *Direct Method* has to be used. This algorithm is faster than the *Direct Method* but it still includes factorial calculations which take too computational time.

2.2. Modified Kintner's algorithm

A modified version of Kintner's recursive algorithm described in Section 2.1 has been introduced in [2]. This algorithm eliminates the usage of the *Direct Method* in ill-posed cases where the recursive equation can not be applied. For these cases other factorial free equations can be used, in order to accelerate the overall computation time.

Algorithm

- $p = q \quad R_{pp}(r) = r^p,$ (7a)
- $p - q = 2 \quad R_{p(p-2)}(r) = pR_{pp}(r) - (p-1)R_{(p-2)(p-2)}(r),$ (7b)
- otherwise $R_{pq}(r) = \dfrac{(K_2 r^2 + K_3)R_{(p-2)q}(r) + K_4 R_{(p-4)q}(r)}{K_1},$ with (7c)

$$K_1 = \frac{(p+q)(p-q)(p-2)}{2}, \quad K_2 = 2p(p-1)(p-2),$$

$$K_3 = -q^2(p-1) - p(p-1)(p-2), \quad K_4 = \frac{-p(p+q-2)(p-q-2)}{2}. \tag{7d}$$

This algorithm has been proved to be of significant efficiency in computing an entire set of Zernike polynomials up to a specific order or a Zernike polynomial of an individual order [2].

2.3. Prata's algorithm

Prata [25] proposed, a recursive algorithm that enables the higher order polynomials to be derived from the lower order ones. This algorithm is slower than Kintner's algorithm [2] and it is not possible to use it in cases where $q = 0$ and $p = q$, as shown below, where the *Direct Method* has to be used.

Algorithm

- $q = 0$ Direct Method using Eq. (2)
- $p = q \; R_{pp}(r) = r^p,$ (8a)
- otherwise $R_{pq}(r) = L_1 R_{(p-1)(q-1)}(r) + L_2 R_{(p-2)q}(r),$ with, (8b)

$$L_1 = \frac{2rp}{p+q}, \quad L_2 = -\frac{p-q}{p+q}. \tag{8c}$$

Prata's recursive algorithm is of the same complexity $O(p^2)$ for specific order p, as Kintner's one, but its overall performance is quite low [2].

2.4. "q-recursive" algorithm

Recently, a novel recursive algorithm with remarkable performance, called "q-recursive" algorithm, has been introduced in [2].

Algorithm

- $p = q \; R_{pp}(r) = r^p,$ (9a)
- $p - q = 2 \; R_{p(p-2)}(r) = pR_{pp}(r) - (p-1)R_{(p-2)(p-2)}(r),$ (9b)

- otherwise $R_{p(q-4)} = H_1 R_{pq}(r + \left(H_2 + \frac{H_3}{r^2}\right) R_{p(q-2)}$, with (9c)

$$H_1 = \frac{q(q-1)}{2} - qH_2 + \frac{H_3(p+q+2)(p-q)}{8},$$

$$H_2 = \frac{H_3(p+q)(p-q+2)}{4(q-1)} + (q-2), \tag{9d}$$

$$H_3 = \frac{-4(q-2)(q-3)}{(p+q-2)(p-q+4)}.$$

A thorough study [2] of the computational performance of the recursive algorithms presented above has shown that "q-recursive" algorithm significantly outperforms the other methods in all test cases. These results establish this algorithm as the most efficient recursive algorithm for Zernike polynomials computation, in terms of CPU execution time.

3. Numerical stability analysis

Although a detailed study of the computational speed of the previously presented recursive algorithms has been presented in [2], there is not any work dealing with the analysis of their numerical behaviour. The numerical stability of a recursive algorithm seems to be more important than its computation speed. For example, the appearance of finite precision errors is of major importance, since a possible error in a step of the algorithm may be accumulated iteration by iteration, by resulting to unreliable quantities. So, an efficient recursive algorithm must quickly compute the desirable quantities by ensuring the algorithm's stability. The results of a fast but unstable algorithm can not be safely used, no matter how fast they have been derived.

A first investigation about the possible finite precision errors generation and propagation, in the "q-recursive" algorithm, has been done by the authors in [12]. In this section a complementary analysis of the numerical stability of the rest recursive algorithms is taking place. The algorithms are compared in respect to their numerical behaviour and the cases where these algorithms fall in unstable situations are defined.

An efficient methodology, which explores the way the finite precision errors are generated and propagated, during a recursive algorithm, has been introduced and used in very popular signal processing algorithms [26–31]. This methodology employs a number of fundamental propositions demonstrating the way the four operations addition, multiplication, division and subtraction, influence the generation and transmission of the quantization error. These propositions are described in the following.

3.1. General remarks

The propositions stated in this paper hold true independently of the radix of the arithmetic system. However, the numerical error generation and propagation will be studied in the decimal representation, because the decimal arithmetic system is far more familiar and clear to users. In this arithmetic system, precision comparison between two numbers will be made in accordance with the following:

Definition 1. Consider two numbers, n_1 and n_2, written in the canonical exponential form, with the same number, n, of decimal digits in the mantissa, i.e.

$$n_1 = d_1 d_2 d_3, \ldots, d_n \times 10^{\tau}, \quad n_2 = \delta_1 \delta_2 \delta_3, \ldots, \delta_n \times 10^{\rho},$$

with

$$\tau \geqslant \rho.$$

Then, these two numbers differ by K decimal digits, if and only if

$$||n_1| - |n_2|| = d \times 10^{\tau-(n-K)}, \quad 1 \leqslant d < 10.$$

For example, according to this definition, the two numbers 1.234567 and 1.234912 differ indeed by 3 decimal digits, but the following two 1.000002 and 0.999996 differ by 1 decimal digit, as one might intuitively expect.

Definition 2. Let all quantities be written in the canonical exponential form, with n decimal digits in the mantissa. Suppose that the correct value of an arbitrary quantity α is α_c, if all calculations were made with infinite precision. Then, one may define that the quantity α has been computed with precisely the last λ decimal digits erroneous, if and only if, α and α_c differ λ digits according to Definition 1.

As will be evident from the subsequent analysis, all the formulas, that constitute a certain iterative algorithm, are not equivalent from the point of view of the finite precision error generation and propagation.

Notation. For any quantity a expressed in the canonical exponential form, we shall write: (i) man(a) for the mantissa of a, and (ii) $E(a)$ for the exponent of a

Proposition 1. *Let all the involved quantities be computed with finite precision of n decimal digits in the mantissa, and consider any quantity computed by means of a formula of the type*

Multiplication $x = y \cdot z$.

Suppose that, due to the previous finite precision calculations, the quantity y has been computed with precisely the last λ decimal digits erroneous, while z has been computed with up to λ decimal digits erroneous. Then,

(i) *if* $|man(y) \cdot man(z)| \geqslant 10$, *then x is computed with precisely the last λ or $\lambda - 1$ decimal digits erroneous.*
(ii) *if* $|man(y) \cdot man(z)| < 10$, *then x is computed with precisely the last λ or $\lambda + 1$ decimal digits erroneous.*

Proposition 2. *Let all the following quantities be computed with finite precision of n decimal digits in the mantissa, and consider any quantity computed through a formula of the type*:

Division $x = \dfrac{y}{z}$.

Suppose that, due to the previous finite precision calculations, the quantity y has been computed with precisely the last λ decimal digits erroneous, while z has been computed with up to λ decimal digits erroneous. Then,

(iii) *if* $|man(y)/man(z)| \geqslant 1$, *then x is computed with precisely the last λ or $\lambda - 1$ decimal digits erroneous.*
(iv) *if* $|\ man(y)/man(z)| < 1$, *then x is computed with precisely the last λ or $\lambda + 1$ decimal digits erroneous.*

Proposition 3. *Let all the involved quantities be computed with finite precision of n decimal digits in the mantissa, and consider any quantity calculated through a formula of the type*

Subtraction $x = y - z$,

with

$$y \cdot z > 0,$$

where

$$x = x_1, x_2, x_3, \ldots, x_n \times 10^{\delta}, \quad y = y_1, y_2, y_3, \ldots, \quad y_n \times 10^{\tau}, \quad z = z_1 z_2 z_3 \ldots z_n \times 10^{\rho}$$

are such that

$$\delta < \max\{\tau, \rho\} \Longleftrightarrow E(x) < \max\{E(y), E(z)\}.$$

Let

$$d = |\max\{\tau, \rho\} - \delta|.$$

Moreover, suppose that, due to the previous finite precision calculations, the higher order quantity say y has been computed with precisely the last λ decimal digits erroneous, while z has been computed with a number of erroneous decimal digits equal to, or smaller than λ. Then x is computed with the last $(\lambda + d)$ decimal digits erroneous.

Proposition 4 (The numerical error relaxation shift). *Let all quantities be computed with finite precision of n decimal digits in the mantissa, and consider any quantity x computed through a sum of two quantities, that is,*

Addition $x = y + z$.

Suppose, moreover, that z has its last λ *decimal digits erroneous and that the exponent of z is by v smaller than the exponent of y, that is,*

$$v = E(y) - E(z) > 0.$$

Then z transfers to x only $\lambda - v$ *erroneous decimal digits if* $\lambda - v > 0$ *or it does not transfer finite precision error at all, if* $\lambda - v \leqslant 0$.

3.2. Finite precision errors of the recursive algorithms

In order to analyze the numerical behaviour of the recursive algorithms presented in Section 2, we proceed according to the following steps [31], for each algorithm:

Step 1: We execute the algorithm with n digits precision for the mantissa
Step 2: In parallel, we execute the algorithm with $2n$ digits precision for the mantissa
Step 3: We cast any quantity z_{2n} computed by $2n$ decimal digits precision to a quantity $\bar{z}_n$ of n decimal digits precision.
Step 4: We compare quantities $\bar{z}_n$ and z_n, according to Definition 1 and in this way we obtain the exact number of erroneous decimal digits with which each quantity z_n is computed.

By applying the above methodology and keeping in mind the definitions and propositions previously discussed, each recursive algorithm is studied and the corresponding results are presented next. In the forthcoming tables the finite precision error, in erroneous digits, is being measured according to Definition 1 and is depicted in bold face.

3.2.1. Kintner's algorithm

Kintner's algorithm, presents two kind of numerical errors. The first one is the overflow error due to range limitations of data type used. For example, the float (seven digit precision) data type has a valid range $(1.18 \times 10^{-45}, 3.40 \times 10^{45})$, while the double (15 digit precision) data type has a valid range $(2.23 \times 10^{-308}, 1.79 \times 10^{308})$, in the case of IBM PC compatible computers. The second type of error is the finite precision error (FPE),

Case 1: $p = q$ *or* $p - q = 2$

In this case, the computation of the radial polynomial $R_{pq}(r)$, is performed using the direct method (2). The direct method presents overflow errors due to the limitations on representing the factorials of a big numbers $(n!)$ and the radius in the powers of big numbers (r^a). Table 1 illustrates this behaviour, of the direct method, where the values $0.00000000 \times 10^{00}$ corresponds to an overflowed quantities.

Case 2: $p \neq q$ *or* $p - q \neq 2$

In this case the radial polynomials $R_{pq}(r)$ are computed by using Eq. (6a). By applying the algorithm described in Section 3.2, it is concluded that the quantities K_1, K_2, K_3, K_4, do not present any finite precision error.

However, the term $K_2 r^2$ generates a finite precision error of 2 erroneous digits. This error is quite small and it cannot lead the whole algorithm to destroy, as it is shown in Table 2.

The term $K_2 r^2 + K_3$, seems to be a candidate source of generating finite precision error, since there are cases where the erroneous digits are many, e.g. 7 digits. The mathematical operation that is responsible for the generation of this error is the subtraction between numbers with common digits, as Table 3 presents.

The finite precision error will be significantly increased by leading the algorithm to unreliable results, if the following condition is satisfied:

$$K_2 r^2 + K_3 \simeq 0, \iff r \simeq \sqrt{\frac{1}{2} + \frac{q^2}{2p(p-2)}}. \tag{10}$$

Table 1
Computation accuracy $R_{pq}(r)$ quantity, for $p = q$ or $p - q = 2$

Input values				Output value
p	q	r		R_{pq}
23	23	0.00	64bit	$9.99999486 \times 10^{-47}$
			32bit	$0.00000000 \times 10^{00}$
				9 digits error
23	23	0.01	64bit	$1.00000000 \times 10^{-46}$
			32bit	$0.00000000 \times 10^{00}$
				9 digits error
24	24	0.01	64bit	$1.00000000 \times 10^{-48}$
			32bit	$0.00000000 \times 10^{00}$
				9 digits error
25	25	0.01	64bit	$1.00000000 \times 10^{-50}$
			32bit	$0.00000000 \times 10^{00}$
				9 digits error
26	24	0.01	64bit	$-2.49974000 \times 10^{-47}$
			32bit	$0.00000000 \times 10^{00}$
				9 digits error
26	26	0.01	64bit	$1.00000000 \times 10^{-52}$
			32bit	$0.00000000 \times 10^{00}$
				9 digits error
27	27	0.01	64bit	$1.00000000 \times 10^{-54}$
			32bit	$0.00000000 \times 10^{00}$
				9 digits error

Table 2
Computation accuracy of $K_2 r^2$ quantity

Input values					Output value
p	q	r		K_2	$K_2 r^2$
4	0	0.01	64bit	$4.80000000 \times 10^{-01}$	$4.80000000 \times 10^{-03}$
			32bit	$4.80000000 \times 10^{-01}$	$4.79999976 \times 10^{-03}$
				0 digits error	**2 digits error**
4	0	0.03	64bit	$4.80000000 \times 10^{-01}$	$4.32000000 \times 10^{-02}$
			32bit	$4.80000000 \times 10^{-01}$	$4.32000011 \times 10^{-02}$
				0 digits error	**2 digits error**
5	1	0.05	64bit	$1.20000000 \times 10^{02}$	$3.00000000 \times 10^{-01}$
			32bit	$1.20000000 \times 10^{02}$	$2.99999982 \times 10^{-01}$
				0 digits error	**2 digits error**
6	0	0.05	64bit	$2.40000000 \times 10^{02}$	$6.00000000 \times 10^{-01}$
			32bit	$2.40000000 \times 10^{02}$	$5.99999964 \times 10^{-01}$
				0 digits error	**2 digits error**
6	2	0.05	64bit	$2.40000000 \times 10^{02}$	$6.00000000 \times 10^{-01}$
			32bit	$2.40000000 \times 10^{02}$	$5.99999964 \times 10^{-01}$
				0 digits error	**2 digits error**
7	1	0.09	64bit	$4.20000000 \times 10^{02}$	$3.40200000 \times 10^{00}$
			32bit	$4.20000000 \times 10^{02}$	$3.40200019 \times 10^{00}$
				0 digits error	**2 digits error**
7	3	0.09	64bit	$4.20000000 \times 10^{02}$	$3.40200000 \times 10^{00}$
			32bit	$4.20000000 \times 10^{02}$	$3.40200019 \times 10^{00}$
				0 digits error	**2 digits error**

However, condition (10) can't be satisfied for any values of p and q, since the resulted radius r takes values greater than 1, which is outside the region of the unit disk and where the radial polynomials are undefined.

Thus, we can conclude that while $K_2r^2 + K_3$ generates finite precision error, this error can't destroy the algorithm.

The next term of (6a), $(K_2r^2 + K_3)/K_1$, behaves similar to the $K_2r^2 + K_3$ term. The error produced in the previous state is carried out to the next quantity, as Table 4 shows.

In the case of $(K_2r^2 + K_3)/K_1 \cdot R_{(p-2)q}(r)$ term, the produced finite precision error, is very critical, about 8 erroneous digits, since the computed quantities differ significantly.

This error is not produced by the multiplication but it is rather carried by the previous state, due to the subtraction $(K_2r^2 + K_3)$ or the overflow of $R_{(p-2)q}(r)$ term, as presented in Table 5.

Table 6 illustrates the finite precision errors generated when the K_4/K_1 quantity is being computed. As it can be seen, this error is of small magnitude, 2 digits error.

As it can be seen in Table 7, the $K_4/K_1 \cdot R_{(p-4)q}(r)$ term, is computed with the same number of erroneous digits as $R_{(p-4)q}(r)$ and thus the multiplication operation in this case doesn't constitute a finite precision error source.

Finally, the radial polynomial of pth order and qth repetition, which is computed by using Eq. (6a), presents high finite precision errors, in some cases this error is of 8 erroneous digits. From Table 8, one realizes that the error increases when the subtraction of two values with common digits is performed.

This ill-posed subtraction affects the final result and an investigation of the appropriate conditions according to which this operation may be an important source of errors, has to be performed.

The subtraction that is responsible for generating high finite precision errors and affects the stability of the algorithm is performed between two values having common digits.

Table 3
Computation accuracy of $K_2r^2 + K_3$ quantity

Input values						Output value
p	q	r		K_2r^2	K_3	$K_2r^2 + K_3$
5	1	0.73	64bit	$6.39480000 \times 10^{01}$	$-6.40000000 \times 10^{01}$	$-5.20000000 \times 10^{-02}$
			32bit	$6.39479980 \times 10^{01}$	$-6.40000000 \times 10^{01}$	$-5.20029068 \times 10^{-02}$
				2 digits error	**0 digits error**	**5 digits error**
14	4	0.74	64bit	$2.39191680 \times 10^{03}$	$-2.39200000 \times 10^{03}$	$-8.32000000 \times 10^{-02}$
			32bit	$2.39191680 \times 10^{03}$	$-2.39200000 \times 10^{03}$	$-8.33253860 \times 10^{-02}$
				0 digits error	**0 digits error**	**7 digits error**
15	1	0.71	64bit	$2.75238600 \times 10^{03}$	$-2.74400000 \times 10^{03}$	$8.38600000 \times 10^{00}$
			32bit	$2.75238623 \times 10^{03}$	$-2.74400000 \times 10^{03}$	$8.38613510 \times 10^{00}$
				2 digits error	**0 digits error**	**5 digits error**
15	7	0.79	64bit	$3.40758600 \times 10^{03}$	$-3.41600000 \times 10^{03}$	$-8.41400000 \times 10^{00}$
			32bit	$3.40758618 \times 10^{03}$	$-3.41600000 \times 10^{03}$	$-8.41383934 \times 10^{00}$
				2 digits error	**0 digits error**	**5 digits error**
16	10	0.85	64bit	$4.85520000 \times 10^{03}$	$-4.86000000 \times 10^{03}$	$-4.80000000 \times 10^{00}$
			32bit	$4.85520020 \times 10^{03}$	$-4.86000000 \times 10^{03}$	$-4.79982376 \times 10^{00}$
				2 digits error	**0 digits error**	**5 digits error**
28	18	0.85	64bit	$2.84029200 \times 10^{04}$	$-2.84040000 \times 10^{04}$	$-1.08000000 \times 10^{00}$
			32bit	$2.84029219 \times 10^{04}$	$-2.84040000 \times 10^{04}$	$-1.07896900 \times 10^{00}$
				2 digits error	**0 digits error**	**6 digits error**
30	26	0.95	64bit	$4.39698000 \times 10^{04}$	$-4.39640000 \times 10^{04}$	$5.80000000 \times 10^{00}$
			32bit	$4.39697969 \times 10^{04}$	$-4.39640000 \times 10^{04}$	$5.79872227 \times 10^{00}$
				2 digits error	**0 digits error**	**6 digits error**

Table 4
Computation accuracy of $(K_2r^2 + K_3)/K_1$ quantity

Input values						Output value
p	q	r		$K_2r^2 + K_3$	K_1	$(K_2r^2 + K_3)/K_1$
11	3	0.74	64bit	$4.24800000 \times 10^{00}$	$5.04000000 \times 10^{02}$	$8.42857143 \times 10^{-03}$
			32bit	$4.24794292 \times 10^{00}$	$5.04000000 \times 10^{02}$	$8.42845906 \times 10^{-03}$
				4 digits error	**0 digits error**	**5 digits error**
14	4	0.74	64bit	$-8.32000000 \times 10^{-02}$	$1.08000000 \times 10^{03}$	$-7.70370370 \times 10^{-05}$
			32bit	$-8.33253860 \times 10^{-02}$	$1.08000000 \times 10^{03}$	$-7.71531340 \times 10^{-05}$
				7 digits error	**0 digits error**	**7 digits error**
15	7	0.79	64bit	$-8.41400000 \times 10^{00}$	$1.14400000 \times 10^{03}$	$-7.35489510 \times 10^{-03}$
			32bit	$-8.41383934 \times 10^{00}$	$1.14400000 \times 10^{03}$	$-7.35475449 \times 10^{-03}$
				5 digits error	**0 digits error**	**5 digits error**
15	7	0.79	64bit	$-8.41400000 \times 10^{00}$	$1.14400000 \times 10^{03}$	$-7.35489510 \times 10^{-03}$
			32bit	$-8.41383934 \times 10^{00}$	$1.14400000 \times 10^{03}$	$-7.35475449 \times 10^{-03}$
				5 digits error	**0 digits error**	**5 digits error**
16	10	0.85	64bit	$-4.80000000 \times 10^{00}$	$1.09200000 \times 10^{03}$	$-4.39560440 \times 10^{-03}$
			32bit	$-4.79982376 \times 10^{00}$	$1.09200000 \times 10^{03}$	$-4.39544301 \times 10^{-03}$
				5 digits error	**0 digits error**	**5 digits error**
28	18	0.85	64bit	$-1.08000000 \times 10^{00}$	$5.98000000 \times 10^{03}$	$-1.80602007 \times 10^{-04}$
			32bit	$-1.07896900 \times 10^{00}$	$5.98000000 \times 10^{03}$	$-1.80429604 \times 10^{-04}$
				6 digits error	**0 digits error**	**6 digits error**
30	26	0.95	64bit	$5.80000000 \times 10^{00}$	$3.13600000 \times 10^{03}$	$1.84948980 \times 10^{-03}$
			32bit	$5.79872227 \times 10^{00}$	$3.13600000 \times 10^{03}$	$1.84908230 \times 10^{-03}$
				6 digits error	**0 digits error**	**5 digits error**

Table 5
Computation accuracy of $(K_2r^2 + K_3)/K_1 \cdot R_{(p-2)q}(r)$ quantity

Input values						Output value
p	q	r		$(K_2r^2 + K_3)/K_1$	$R_{(p-2)q}(r)$	$(K_2r^2 + K_3)/K_1 \cdot R_{(p-2)q}(r)$
13	1	0.98	64bit	$1.69707013 \times 10^{00}$	$3.38944528 \times 10^{-03}$	$5.75212635 \times 10^{-03}$
			32bit	$1.69707012 \times 10^{00}$	$3.39005049 \times 10^{-03}$	$5.75315347 \times 10^{-03}$
				0 digits error	**5 digits error**	**6 digits error**
14	4	0.74	64bit	$-7.70370370 \times 10^{-05}$	$-1.20536977 \times 10^{-01}$	$9.28581158 \times 10^{-06}$
			32bit	$-7.71531340 \times 10^{-05}$	$-1.20536923 \times 10^{-01}$	$9.29980160 \times 10^{-06}$
				7 digits error	**2 digits error**	**7 digits error**
18	0	0.99	64bit	$1.81371111 \times 10^{00}$	$-1.42994931 \times 10^{-03}$	$-2.59351494 \times 10^{-03}$
			32bit	$1.81371105 \times 10^{00}$	$-1.42934895 \times 10^{-03}$	$-2.59242603 \times 10^{-03}$
				1 digits error	**5 digits error**	**6 digits error**
24	8	0.92	64bit	$1.23248636 \times 10^{00}$	$-3.98715398 \times 10^{-06}$	$-4.91411291 \times 10^{-06}$
			32bit	$1.23248649 \times 10^{00}$	$-3.72576210 \times 10^{-06}$	$-4.59195144 \times 10^{-06}$
				2 digits error	**8 digits error**	**8 digits error**
27	23	0.01	64bit	$-1.25201960 \times 10^{01}$	$-2.39975000 \times 10^{-45}$	$3.00453404 \times 10^{-44}$
			32bit	$-1.25201960 \times 10^{01}$	$-2.80259693 \times 10^{-45}$	$3.50324616 \times 10^{-44}$
				0 digits error	**8 digits error**	**8 digits error**
29	5	0.98	64bit	$1.76902861 \times 10^{00}$	$-2.72819333 \times 10^{-04}$	$-4.82625207 \times 10^{-04}$
				$1.76902854 \times 10^{00}$	$-2.73729936 \times 10^{-04}$	$-4.84236080 \times 10^{-04}$
			32bit	**1 digits error**	**6 digits error**	**7 digits error**
29	23	0.01	64bit	$-8.72070636 \times 10^{00}$	$2.99935004 \times 10^{-44}$	$-2.61564509 \times 10^{-43}$
			32bit	$-8.72070599 \times 10^{00}$	$2.94272678 \times 10^{-44}$	$-2.56437619 \times 10^{-43}$
				2 digits error	**7 digits error**	**7 digits error**

Table 6
Computation accuracy of K_4/K_1 quantity

Input values					Output value
p	q		K_4	K_1	K_4/K_1
5	1	64bit	$-2.00000000 \times 10^{01}$	$3.60000000 \times 10^{01}$	$-5.55555556 \times 10^{-01}$
		32bit	$-2.00000000 \times 10^{01}$	$3.60000000 \times 10^{01}$	$-5.55555582 \times 10^{-01}$
			0 digits error	**0 digits error**	**2 digits error**
6	0	64bit	$-4.80000000 \times 10^{01}$	$7.20000000 \times 10^{01}$	$-6.66666667 \times 10^{-01}$
		32bit	$-4.80000000 \times 10^{01}$	$7.20000000 \times 10^{01}$	$-6.66666687 \times 10^{-01}$
			0 digits error	**0 digits error**	**2 digits error**
7	1	64bit	$-8.40000000 \times 10^{01}$	$1.20000000 \times 10^{02}$	$-7.00000000 \times 10^{-01}$
		32bit	$-8.40000000 \times 10^{01}$	$1.20000000 \times 10^{02}$	$-6.99999988 \times 10^{-01}$
			0 digits error	**0 digits error**	**2 digits error**
8	2	64bit	$-1.28000000 \times 10^{02}$	$1.80000000 \times 10^{02}$	$-7.11111111 \times 10^{-01}$
		32bit	$-1.28000000 \times 10^{02}$	$1.80000000 \times 10^{02}$	$-7.11111128 \times 10^{-01}$
			0 digits error	**0 digits error**	**2 digits error**
8	4	64bit	$-8.00000000 \times 10^{01}$	$1.44000000 \times 10^{02}$	$-5.55555556 \times 10^{-01}$
		32bit	$-8.00000000 \times 10^{01}$	$1.44000000 \times 10^{02}$	$-5.55555582 \times 10^{-01}$
			0 digits error	**0 digits error**	**2 digits error**
9	1	64bit	$-2.16000000 \times 10^{02}$	$2.80000000 \times 10^{02}$	$-7.71428571 \times 10^{-01}$
		32bit	$-2.16000000 \times 10^{02}$	$2.80000000 \times 10^{02}$	$-7.71428585 \times 10^{-01}$
			0 digits error	**0 digits error**	**2 digits error**
9	3	64bit	$-1.80000000 \times 10^{02}$	$2.52000000 \times 10^{02}$	$-7.14285714 \times 10^{-01}$
		32bit	$-1.80000000 \times 10^{02}$	$2.52000000 \times 10^{02}$	$-7.14285731 \times 10^{-01}$
			0 digits error	**0 digits error**	**2 digits error**

Table 7
Computation accuracy of $K_4/K_1 \cdot R_{(p-4)q}(r)$ quantity

Input values						Output value
p	q	r		K_4/K_1	$R_{(p-4)q}(r)$	$K_4/K_1 \cdot R_{(p-4)q}(r)$
20	12	0.98	64bit	$-7.81250000 \times 10^{-01}$	$1.40621237 \times 10^{-04}$	$-1.09860341 \times 10^{-04}$
			32bit	$-7.81250000 \times 10^{-01}$	$1.41222539 \times 10^{-04}$	$-1.10330111 \times 10^{-04}$
				0 digits error	**6 digits error**	**6 digits error**
22	6	0.79	64bit	$-8.93750000 \times 10^{-01}$	$-2.82887650 \times 10^{-04}$	$2.52830837 \times 10^{-04}$
			32bit	$-8.93750012 \times 10^{-01}$	$-2.83055298 \times 10^{-04}$	$2.52980681 \times 10^{-04}$
				2 digits error	**6 digits error**	**6 digits error**
26	8	0.92	64bit	$-9.06318083 \times 10^{-01}$	$-3.98715398 \times 10^{-06}$	$3.61362975 \times 10^{-06}$
			32bit	$-9.06318069 \times 10^{-01}$	$-3.72576210 \times 10^{-06}$	$3.37672554 \times 10^{-06}$
				2 digits error	**8 digits error**	**8 digits error**
26	18	0.92	64bit	$-7.75568182 \times 10^{-01}$	$-2.73373473 \times 10^{-04}$	$2.12019768 \times 10^{-04}$
			32bit	$-7.75568187 \times 10^{-01}$	$-2.73571844 \times 10^{-04}$	$2.12173618 \times 10^{-04}$
				1 digits error	**6 digits error**	**6 digits error**
26	22	0.01	64bit	$-5.19097222 \times 10^{-01}$	$1.00000000 \times 10^{-44}$	$-5.19097222 \times 10^{-45}$
			32bit	$-5.19097209 \times 10^{-01}$	$9.80908925 \times 10^{-45}$	$-5.60519386 \times 10^{-45}$
				2 digits error	**7 digits error**	**8 digits error**
29	23	0.01	64bit	$-6.88509022 \times 10^{-01}$	$-2.39975000 \times 10^{-45}$	$1.65224953 \times 10^{-45}$
			32bit	$-6.88509047 \times 10^{-01}$	$-2.80259693 \times 10^{-45}$	$1.40129846 \times 10^{-45}$
				2 digits error	**8 digits error**	**8 digits error**
30	26	0.02	64bit	$-5.16581633 \times 10^{-01}$	$6.71088640 \times 10^{-45}$	$-3.46672065 \times 10^{-45}$
			32bit	$-5.16581655 \times 10^{-01}$	$7.00649232 \times 10^{-45}$	$-4.20389539 \times 10^{-45}$
				2 digits error	**8 digits error**	**8 digits error**

These values are the $(K_2r^2 + K_3)/K_1 \cdot R_{(p-2)q}(r)$ and $K_4/K_1 \cdot R_{(p-4)q}(r)$ terms. By analyzing the conditions where these two values are made almost equal, for specific order and repetition, we have

For $p = 4$, $q = 0$, by using (6) we have,

$$R_{40} = \frac{K_2r^2 + K_3}{K_1}R_{20} + \frac{K_4}{K_1}R_{00},$$

where

$$K_1 = 16, \quad K_2 = 48, \quad K_3 = -24, \quad K_4 = -8 \quad \text{and} \quad R_{20} = 2r^2 - 1, \quad R_{00} = 1.$$

The error is presented when

$$\frac{K_2r^2 + K_3}{K_1}R_{20} = -\frac{K_4}{K_1}R_{00} \Longleftrightarrow \left(K_2r^2 + K_3\right)R_{20} = -K_4R_{00} \Longleftrightarrow$$
$$(48r^2 - 24)(2r^2 - 1) = 8 \Longleftrightarrow 6r^4 - 6r^2 + 1 \simeq 0 \Longleftrightarrow$$
$$r \simeq \sqrt{\frac{3 \pm \sqrt{3}}{6}}.$$

By proceeding in the same way, the radius values for which the subtraction of two common numbers may generate high finite precision errors, for several orders and repetitions, have been derived and presented in Table 9.

From the above analysis is concluded that for every usage of the recursive Eq. (6a), there are combinations between the moment order p, the repetition q and the radial, for which a significant finite precision error is being generated.

This error increases when a subtraction of common real numbers is presented, by resulting to totally unreliable radial polynomial values.

Table 8
Computation accuracy of $R_{pq}(r)$ quantity, for $p \neq q$ and $p - q \neq 2$

Input values						Output value
p	q	r		$(K_2r^2 + K_3)/K_1 \cdot R_{(p-2)q}(r)$	$K_4/K_1 \cdot R_{(p-4)q}(r)$	$R_{pq}(r)$
16	12	0.98	64bit	$4.16520924 \times 10^{-01}$	$-4.16380302 \times 10^{-01}$	$1.40621237 \times 10^{-04}$
			32bit	$4.16521460 \times 10^{-01}$	$-4.16380405 \times 10^{-01}$	$1.41064025 \times 10^{-04}$
				3 digits error	**3 digits error**	**6 digits error**
18	6	0.79	64bit	$-7.59320343 \times 10^{-02}$	$7.56491467 \times 10^{-02}$	$-2.82887650 \times 10^{-04}$
			32bit	$-7.59320781 \times 10^{-02}$	$-8.99671465 \times 10^{-02}$	$-2.83046189 \times 10^{-04}$
				3 digits error	**4 digits error**	**6 digits error**
22	8	0.92	64bit	$2.79200177 \times 10^{-01}$	$-2.79204164 \times 10^{-01}$	$-3.98715398 \times 10^{-06}$
			32bit	$2.79200405 \times 10^{-01}$	$-2.79204130 \times 10^{-01}$	$-3.74297929 \times 10^{-06}$
				3 digits error	**2 digits error**	**8 digits error**
22	18	0.92	64bit	$1.16210871 \times 10^{-01}$	$-1.16484244 \times 10^{-01}$	$-2.73373473 \times 10^{-04}$
			32bit	$1.16210751 \times 10^{-01}$	$-1.16484277 \times 10^{-01}$	$-2.73526326 \times 10^{-04}$
				3 digits error	**2 digits error**	**6 digits error**
24	4	0.65	64bit	$-8.64667230 \times 10^{-02}$	$8.66212905 \times 10^{-02}$	$1.54567488 \times 10^{-04}$
			32bit	$-8.64667222 \times 10^{-02}$	$8.66211578 \times 10^{-02}$	$1.54434267 \times 10^{-04}$
				1 digits error	**4 digits error**	**6 digits error**
27	23	0.01	64bit	$3.00453404 \times 10^{-44}$	$-5.18400000 \times 10^{-47}$	$2.99935004 \times 10^{-44}$
			32bit	$3.50324616 \times 10^{-44}$	$0.00000000 \times 10^{00}$	$3.50324616 \times 10^{-44}$
				8 digits error	**overflow**	**8 digits error**
29	23	0.01	64bit	$-2.61564509 \times 10^{-43}$	$1.65224953 \times 10^{-45}$	$-2.59912260 \times 10^{-43}$
			32bit	$-2.56437619 \times 10^{-43}$	$1.40129846 \times 10^{-45}$	$-2.55036321 \times 10^{-43}$
				7 digits error	**8 digits error**	**7 digits error**

An important observation that is derived from the above Table 9, is that as the moment order increases, the number of the radial values for which the appropriate conditions are satisfied, is also dramatically increased.

This unstable behaviour constitutes a major drawback of this algorithm, since its numerical stability can be easily altered, even for low moment orders.

3.2.2. Modified Kintner's algorithm

The modified Kintner's algorithm, as proposed in [2], differs from the original one in using more simple equations instead of the direct method (2), in cases of $p = q$ (7a) and $p - q = 2$ (7b).

Case 1: $p = q$

The usage of (7a) instead of the direct method for $p = q$, avoids the overflows which occurred due to the computation of the factorials of big numbers. However, overflow can be still presented when computing the radius to the power of big numbers, as it can be seen in Table 10.

Case 2: $p - q = 2$

In this case it is possible to generate significant finite precision errors, when real numbers with common digits are being subtracted according to the recursive (7b). By applying the methodology of Section 3.2, the finite precision error for specific radius, moment order and repetition are extracted and illustrated in Table 11.

The condition under which the subtraction would be performed between common real numbers is obtained as,

Table 9
High finite precision errors in $R_{pq}(r)$ quantity, of Eq. (6a)

Input values						Output value
p	q	r		$(K_2r^2 + K_3)/K_1 \cdot R_{(p-2)q}(r)$	$K_4/K_1 \cdot R_{(p-4)q}(r)$	$R_{pq}(r)$
4	0	$\sqrt{\frac{3+\sqrt{3}}{6}}$	64bit	$5.00000000 \times 10^{-01}$	$-5.00000000 \times 10^{-01}$	$-1.66533454 \times 10^{-16}$
			32bit	$5.00000060 \times 10^{-01}$	$-5.00000000 \times 10^{-01}$	$8.52900399 \times 10^{-08}$
				2 digits error	**0 digits error**	**17 digits error**
4	0	$\sqrt{\frac{3-\sqrt{3}}{6}}$	64bit	$5.00000000 \times 10^{-01}$	$-5.00000000 \times 10^{-01}$	$0.00000000 \times 10^{00}$
			32bit	$4.99999970 \times 10^{-01}$	$-5.00000000 \times 10^{-01}$	$-1.79482313 \times 10^{-08}$
				2 digits error	**0 digits error**	**18 digits error**
5	1	$\sqrt{\frac{6+\sqrt{6}}{10}}$	64bit	$5.10672812 \times 10^{-01}$	$-5.10672812 \times 10^{-01}$	$4.44089210 \times 10^{-16}$
			32bit	$5.10672927 \times 10^{-01}$	$-5.10672808 \times 10^{-01}$	$1.18265987 \times 10^{-07}$
				3 digits error	**1 digits error**	**18 digits error**
5	1	$\sqrt{\frac{6-\sqrt{6}}{10}}$	64bit	$3.31034213 \times 10^{-01}$	$-3.31034213 \times 10^{-01}$	$1.66533454 \times 10^{-16}$
			32bit	$3.31034243 \times 10^{-01}$	$-3.31034184 \times 10^{-01}$	$5.12383522 \times 10^{-08}$
				2 digits error	**2 digits error**	**17 digits error**
6	0	$\sqrt{\frac{1}{2}}$	64bit	$0.00000000 \times 10^{00}$	$0.00000000 \times 10^{00}$	$0.00000000 \times 10^{00}$
			32bit	$0.00000000 \times 10^{00}$	$2.28190284 \times 10^{-08}$	$2.28190284 \times 10^{-08}$
				0 digits error	**1 digits error**	**18 digits error**
6	0	$\sqrt{\frac{5+\sqrt{15}}{10}}$	64bit	$5.16397779 \times 10^{-01}$	$-5.16397779 \times 10^{-01}$	$9.99200722 \times 10^{-16}$
			32bit	$5.16398013 \times 10^{-01}$	$-5.16397834 \times 10^{-01}$	$1.60331666 \times 10^{-07}$
				3 digits error	**2 digits error**	**18 digits error**
6	0	$\sqrt{\frac{5-\sqrt{15}}{10}}$	64bit	$-5.16397779 \times 10^{-01}$	$5.16397779 \times 10^{-01}$	$-1.11022302 \times 10^{-16}$
			32bit	$-5.16397834 \times 10^{-01}$	$5.16397774 \times 10^{-01}$	$-3.62354520 \times 10^{-08}$
				2 digits error	**1 digits error**	**17 digits error**
6	2	$\sqrt{\frac{10+\sqrt{10}}{15}}$	64bit	$4.93585412 \times 10^{-01}$	$-4.93585412 \times 10^{-01}$	$-1.22124533 \times 10^{-15}$
			32bit	$4.93585229 \times 10^{-01}$	$-4.93585408 \times 10^{-01}$	$-1.81533622 \times 10^{-07}$
				3 digits error	**1 digits error**	**17 digits error**
6	2	$\sqrt{\frac{10-\sqrt{10}}{15}}$	64bit	$2.56414588 \times 10^{-01}$	$-2.56414588 \times 10^{-01}$	$1.66533454 \times 10^{-16}$
			32bit	$2.56414592 \times 10^{-01}$	$-2.56414592 \times 10^{-01}$	$-2.00921910 \times 10^{-08}$
				1 digits error	**1 digits error**	**17 digits error**

Table 10
Computation accuracy of $R_{pp}(r)$ quantity, for $p = q$

Input values			Output value
p	r		$R_{pp}(r)$
23	0.01	64bit	$1.00000000 \times 10^{-46}$
		32bit	$0.00000000 \times 10^{00}$
			9 digits error
24	0.01	64bit	$1.00000000 \times 10^{-48}$
		32bit	$0.00000000 \times 10^{00}$
			9 digits error
25	0.01	64bit	$1.00000000 \times 10^{-50}$
		32bit	$0.00000000 \times 10^{00}$
			9 digits error
26	0.01	64bit	$1.00000000 \times 10^{-52}$
		32bit	$0.00000000 \times 10^{00}$
			9 digits error
27	0.01	64bit	$1.00000000 \times 10^{-54}$
		32bit	$0.00000000 \times 10^{00}$
			9 digits error
27	0.02	64bit	$1.34217728 \times 10^{-46}$
		32bit	$0.00000000 \times 10^{00}$
			9 digits error
28	0.01	64bit	$1.00000000 \times 10^{-56}$
		32bit	$0.00000000 \times 10^{00}$
			9 digits error

Table 11
High finite precision errors in $R_{p(p-2)}(r)$ quantity, of Eq. (7b)

Input values						Output value
p	q	r		$pR_{pp}(r)$	$(p-1)R_{(p-2)(p-2)}(r)$	$R_{p(p-2)}$
2	0	$\sqrt{\frac{1}{2}}$	64bit	$1.00000000 \times 10^{00}$	$1.00000000 \times 10^{00}$	$0.00000000 \times 10^{00}$
			32bit	$9.99999940 \times 10^{-01}$	$1.00000000 \times 10^{00}$	$-5.96046448 \times 10^{-08}$
				1 digits error	**0 digits error**	**18 digits error**
3	1	$\sqrt{\frac{2}{3}}$	64bit	$1.63299316 \times 10^{00}$	$1.63299316 \times 10^{00}$	$0.00000000 \times 10^{00}$
			32bit	$1.63299346 \times 10^{00}$	$1.63299322 \times 10^{00}$	$1.78813934 \times 10^{-07}$
				2 digits error	**1 digits error**	**18 digits error**
4	2	$\sqrt{\frac{3}{4}}$	64bit	$2.25000000 \times 10^{00}$	$2.25000000 \times 10^{00}$	$-4.44089210 \times 10^{-16}$
			32bit	$2.24999976 \times 10^{00}$	$2.25000000 \times 10^{00}$	$-2.38418579 \times 10^{-07}$
				2 digits error	**0 digits error**	**18 digits error**
5	3	$\sqrt{\frac{4}{5}}$	64bit	$2.86216701 \times 10^{00}$	$2.86216701 \times 10^{00}$	$-4.44089210 \times 10^{-16}$
			32bit	$2.86216688 \times 10^{00}$	$2.86216688 \times 10^{00}$	$-1.19209290 \times 10^{-07}$
				2 digits error	**2 digits error**	**18 digits error**
6	4	$\sqrt{\frac{5}{6}}$	64bit	$3.47222222 \times 10^{00}$	$3.47222222 \times 10^{00}$	$0.00000000 \times 10^{00}$
			32bit	$3.47222257 \times 10^{00}$	$3.47222233 \times 10^{00}$	$1.78813934 \times 10^{-07}$
				2 digits error	**2 digits error**	**18 digits error**
7	5	$\sqrt{\frac{6}{7}}$	64bit	$4.08116615 \times 10^{00}$	$4.08116615 \times 10^{00}$	$-1.77635684 \times 10^{-15}$
			32bit	$4.08116627 \times 10^{00}$	$4.08116627 \times 10^{00}$	$2.38418579 \times 10^{-07}$
				2 digits error	**2 digits error**	**17 digits error**
8	6	$\sqrt{\frac{7}{8}}$	64bit	$4.68945312 \times 10^{00}$	$4.68945312 \times 10^{00}$	$0.00000000 \times 10^{00}$
			32bit	$4.68945408 \times 10^{00}$	$4.68945408 \times 10^{00}$	$1.19209290 \times 10^{-07}$
				2 digits error	**2 digits error**	**18 digits error**

$$pR_{pp}(r) - (p-1)R_{(p-2)(p-2)}(r) = pr^{p} - (p-1)r^{(p-2)} \simeq 0 \Longleftrightarrow r \simeq \sqrt{\frac{p-1}{p}}, \quad \text{or} \quad r \simeq \sqrt{\frac{q+1}{p}}. \tag{11}$$

The above analysis shows that while the modified Kitner's method permits the calculation of the radial polynomials by avoiding any factorial computations, there is an increased possibility the algorithm lost its stability.

The modified Kintner's algorithm can be unstable more often than the conventional one, since extra conditions, when $p - q = 2$, where ill-posed subtractions can exist for many combinations of p, q and r, as shown in Table 11.

3.2.3. Prata's algorithm

Prata's algorithm can be considered a combination of Kintner's and modified Kintner's algorithms, since it uses the *Direct* method (2) to compute the radial polynomials for $q = 0$ as Kintner's do and Eq. (7a) in the case of $p = q$, as modified Kintner's one. In the cases, where $q \neq 0$ and $p \neq q$, a recursive formula (8b) is used.

By analyzing the way the L_1 and L_2 (8c) quantities are being computed, it is concluded that these terms do not generate significant finite precision errors, as presented in Tables 12 and 13.

However, the previous analysis on the Kintner's like algorithms, where finite precision errors are possible to be generated, due to ill-posed subtractions, focuses the attention to the study of the subtraction in Eq. (8b).

As Tables 14 and 15 show, while the terms being subtracted do not generate finite precision errors greater than 3 digits, their subtraction can lead to high finite precision errors, under certain conditions (Table 16).

The finite precision error is extremely increased when for specific moment order p and repetition q, the radial polynomial is computed for a pixel having radius r, for which the following holds:

Table 12
Computation accuracy of L_1 quantity

Input values				Output value
p	q	r		L_1
3	1	0.03	64bit	$4.50000000 \times 10^{-02}$
			32bit	$4.49999981 \times 10^{-02}$
				2 digits error
3	1	0.04	64bit	$6.00000000 \times 10^{-02}$
			32bit	$5.99999987 \times 10^{-02}$
				2 digits error
9	3	0.53	64bit	$7.95000000 \times 10^{-01}$
			32bit	$7.94999897 \times 10^{-01}$
				3 digits error
18	2	0.53	64bit	$9.54000000 \times 10^{-01}$
			32bit	$9.53999877 \times 10^{-01}$
				3 digits error
18	4	0.53	64bit	$8.67272727 \times 10^{-01}$
			32bit	$8.67272615 \times 10^{-01}$
				3 digits error
18	6	0.53	64bit	$7.95000000 \times 10^{-01}$
			32bit	$7.94999897 \times 10^{-01}$
				3 digits error
23	11	0.72	64bit	$9.74117647 \times 10^{-01}$
			32bit	$9.74117756 \times 10^{-01}$
				3 digits error

Table 13
Computation accuracy of L_2 quantity

Input values			Output value
p	q		L_2
5	1	64bit	$-6.66666667\times10^{-01}$
		32bit	$-6.66666687\times10^{-01}$
			2 digits error
8	2	64bit	$-6.00000000\times10^{-01}$
		32bit	$-6.00000024\times10^{-01}$
			2 digits error
9	1	64bit	$-8.00000000\times10^{-01}$
		32bit	$-8.00000012\times10^{-01}$
			2 digits error
9	5	64bit	$-2.85714286\times10^{-01}$
		32bit	$-2.85714298\times10^{-01}$
			2 digits error
10	2	64bit	$-6.66666667\times10^{-01}$
		32bit	$-6.66666687\times10^{-01}$
			2 digits error
11	1	64bit	$-8.33333333\times10^{-01}$
		32bit	$-8.33333313\times10^{-01}$
			2 digits error
11	3	64bit	$-5.71428571\times10^{-01}$
		32bit	$-5.71428597\times10^{-01}$
			2 digits error

$$R_{31} = L_1R_{20} + L_2R_{11} = \frac{3}{2}r\cdot(2r^2-1) + \left(-\frac{1}{2}\right)r = 3r^3 - 2r \simeq 0 \Longleftrightarrow r(3r^2-2)\simeq 0 \Longleftrightarrow r \simeq \sqrt{\frac{2}{3}},$$

$$R_{42} = L_1R_{31} + L_2R_{22} = \frac{4}{3}r\cdot(3r^3-2r) + (-\frac{1}{3})r^2 = 4r^4 - 3r^2 \simeq 0 \Longleftrightarrow r^2(4r^2-3)\simeq 0 \Longleftrightarrow r \simeq \sqrt{\frac{3}{4}},$$

$$R_{53} = L_1R_{42} + L_2R_{33} = \frac{5}{4}r\cdot(4r^4-3r^2) + \left(-\frac{1}{4}\right)r^3 = 5r^5 - 4r^3 \simeq 0 \Longleftrightarrow r^3(5r^2-4)\simeq 0 \Longleftrightarrow r \simeq \sqrt{\frac{4}{5}},$$

$$R_{64} = L_1R_{53} + L_2R_{44} = \frac{6}{5}r\cdot(5r^5-4r^3) + \left(-\frac{1}{5}\right)r^4 = 6r^6 - 5r^4 \simeq 0 \Longleftrightarrow r^4(6r^2-5)\simeq 0 \Longleftrightarrow r \simeq \sqrt{\frac{5}{6}}.$$

From the above, it is obvious that the radius for which the subtraction generates finite precision errors, satisfies the following generic equation:

$$r = \sqrt{\frac{p-1}{p}}, \quad \text{or} \quad r = \sqrt{\frac{q+1}{p}} \tag{12}$$

for a given moment order p and repetition q.

The above Table 16 presents some possible combinations of p, q and r, which satisfy Eq. (12) and for which the finite precision errors generated are significant and drop the algorithm to unstable situations.

3.2.4. "q-recursive" algorithm

This algorithm constitutes the most recently introduced [2], among the algorithms which are studied in this work and it provides a fast way to compute the radial polynomials of order p and repetition q. It is proved that this algorithm outperforms the other ones, by a significant factor [2].

Table 14
Computation accuracy of $L_1 \cdot R_{(p-1)(q-1)}$ quantity

Input values						Output value
p	q	r		L_1	$R_{(p-1)(q-1)}$	$L_1 \cdot R_{(p-1)(q-1)}$
7	1	0.30	64bit	$5.25000000 \times 10^{-01}$	$-1.48420000 \times 10^{-01}$	$-7.79205000 \times 10^{-02}$
			32bit	$5.25000036 \times 10^{-01}$	$-1.48420006 \times 10^{-01}$	$-7.79205114 \times 10^{-02}$
				2 digits error	**1 digits error**	**3 digits error**
7	1	0.37	64bit	$6.47500000 \times 10^{-01}$	$1.31866228 \times 10^{-01}$	$8.53833827 \times 10^{-02}$
			32bit	$6.47500038 \times 10^{-01}$	$1.31866232 \times 10^{-01}$	$8.53833929 \times 10^{-02}$
				2 digits error	**1 digits error**	**3 digits error**
7	3	0.93	64bit	$1.30200000 \times 10^{00}$	$-6.67874483 \times 10^{-02}$	$-8.69572576 \times 10^{-02}$
			32bit	$1.30200005 \times 10^{00}$	$-6.67874515 \times 10^{-02}$	$-8.69572684 \times 10^{-02}$
				1 digits error	**2 digits error**	**3 digits error**
10	8	0.59	64bit	$6.55555556 \times 10^{-01}$	$-1.21125156 \times 10^{-01}$	$-7.94042692 \times 10^{-02}$
			32bit	$6.55555487 \times 10^{-01}$	$-1.21125154 \times 10^{-01}$	$-7.94042572 \times 10^{-02}$
				2 digits error	**1 digits error**	**3 digits error**
12	8	0.20	64bit	$2.40000000 \times 10^{-01}$	$4.15846400 \times 10^{-04}$	$9.98031360 \times 10^{-05}$
			32bit	$2.40000010 \times 10^{-01}$	$4.15846414 \times 10^{-04}$	$9.98031464 \times 10^{-05}$
				1 digits error	**2 digits error**	**3 digits error**
13	1	0.14	64bit	$2.60000000 \times 10^{-01}$	$3.25954487 \times 10^{-01}$	$8.47481667 \times 10^{-02}$
			32bit	$2.60000020 \times 10^{-01}$	$3.25954497 \times 10^{-01}$	$8.47481787 \times 10^{-02}$
				2 digits error	**1 digits error**	**3 digits error**
13	7	0.07	64bit	$9.10000000 \times 10^{-02}$	$-9.67637837 \times 10^{-06}$	$-8.80550432 \times 10^{-07}$
			32bit	$9.10000056 \times 10^{-02}$	$-9.67637880 \times 10^{-06}$	$-8.80550544 \times 10^{-07}$
				2 digits error	**2 digits error**	**3 digits error**

Recently, this algorithm has been studied by the authors [12], for its numerical stability in terms of finite precision errors generation and propagation.

More precisely, in [12] has been proved that Eq. (9b), generates significant amount of erroneous digits when the following condition is satisfied

$$r \simeq \sqrt{\frac{p-1}{p}}, \quad \text{for } p \geqslant 2. \tag{13}$$

Additionally, the $H_2 + H_3/r^2$ quantity of (9c) is possible to generate considerable finite precision error, when pixels having radius satisfying the following equation exist.

$$r \simeq \sqrt{\frac{4(q-1)(q-3)}{(3-q)(p+q)(p-q+2)+(p+q-2)(p-q+4)(q-1)}}. \tag{14}$$

4. Discussion

The stability analysis of the fast algorithms used for computing the Zernike moments presented previously, leads up to significant conclusions about the numerical behaviour and appropriateness of each algorithm.

The four algorithms studied in this work, Kintner's, Modified Kintner's, Prata's and q-recursive algorithms, produce overflow but more important finite precision errors, under certain conditions. The mathematical operation, which is responsible for the generation of finite precision errors in all cases, is the subtraction performed between real numbers with common digits, as defined by Proposition 3.

Table 15
Computation accuracy of $L_2 \cdot R_{(p-2)q}$ quantity

Input values						Output value
p	Q	r		L_2	$R_{(p-2)q}$	$L_2 \cdot R_{(p-2)q}$
3	1	0.01	64bit	$-5.00000000 \times 10^{-01}$	$1.00000000 \times 10^{-02}$	$-5.00000000 \times 10^{-03}$
			32bit	$-5.00000000 \times 10^{-01}$	$9.99999978 \times 10^{-03}$	$-4.99999989 \times 10^{-03}$
				0 digits error	**1 digits error**	**2 digits error**
3	1	0.09	64bit	$-5.00000000 \times 10^{-01}$	$9.00000000 \times 10^{-02}$	$-4.50000000 \times 10^{-02}$
			32bit	$-5.00000000 \times 10^{-01}$	$9.00000036 \times 10^{-02}$	$-4.50000018 \times 10^{-02}$
				0 digits error	**2 digits error**	**2 digits error**
4	2	0.03	64bit	$-3.33333333 \times 10^{-01}$	$9.00000000 \times 10^{-04}$	$-3.00000000 \times 10^{-04}$
			32bit	$-3.33333343 \times 10^{-01}$	$8.99999985 \times 10^{-04}$	$-3.00000014 \times 10^{-04}$
				1 digits error	**2 digits error**	**2 digits error**
5	1	0.03	64bit	$-6.66666667 \times 10^{-01}$	$-5.99190000 \times 10^{-02}$	$3.99460000 \times 10^{-02}$
			32bit	$-6.66666687 \times 10^{-01}$	$-5.99189997 \times 10^{-02}$	$3.99460010 \times 10^{-02}$
				2 digits error	**1 digits error**	**2 digits error**
9	5	0.64	64bit	−2.85714286e-001	−3.36381839e-001	9.61090967e-002
			32bit	−2.85714298e-001	−3.36381853e-001	9.61091071e-002
				2 digits error	**2 digits error**	**3 digits error**
11	3	0.64	64bit	−5.71428571e-001	1.55900488e-001	−8.90859932e-002
			32bit	−5.71428597e-001	1.55900493e-001	−8.90860036e-002
				2 digits error	**1 digits error**	**3 digits error**
11	3	0.86	64bit	−5.71428571e-001	−1.62654230e-001	9.29452744e-002
			32bit	−5.71428597e-001	−1.62654236e-001	9.29452851e-002
				2 digits error	**1 digits error**	**3 digits error**

Table 16
High finite precision errors in $R_{pq}(r)$ quantity, of Eq. (8b)

Input values						Output value
p	q	r		$L_1 \cdot R_{(p-1)(q-1)}$	$L_2 \cdot R_{(p-2)q}$	R_{pq}
3	1	$\sqrt{\frac{2}{3}}$	64bit	$4.08248290 \times 10^{-01}$	$-4.08248290 \times 10^{-01}$	$-1.11022302 \times 10^{-16}$
			32bit	$4.08248305 \times 10^{-01}$	$-4.08248305 \times 10^{-01}$	$0.00000000 \times 10^{-00}$
				2 digits error	**2 digits error**	**18 digits error**
4	2	$\sqrt{\frac{3}{4}}$	64bit	$2.50000000 \times 10^{-01}$	$-2.50000000 \times 10^{-01}$	$-3.05311332 \times 10^{-16}$
			32bit	$2.49999985 \times 10^{-01}$	$-2.50000000 \times 10^{-01}$	$-1.49011612 \times 10^{-08}$
				2 digits error	**0 digits error**	**17 digits error**
5	3	$\sqrt{\frac{4}{5}}$	64bit	$1.78885438 \times 10^{-01}$	$-1.78885438 \times 10^{-01}$	$-8.32667268 \times 10^{-17}$
			32bit	$1.78885430 \times 10^{-01}$	$-1.78885445 \times 10^{-01}$	$-1.49011612 \times 10^{-08}$
				1 digits error	**1 digits error**	**18 digits error**
6	4	$\sqrt{\frac{5}{6}}$	64bit	$1.38888889 \times 10^{-01}$	$-1.38888889 \times 10^{-01}$	$8.32667268 \times 10^{-17}$
			32bit	$1.38888896 \times 10^{-01}$	$-1.38888881 \times 10^{-01}$	$1.49011612 \times 10^{-08}$
				1 digits error	**1 digits error**	**18 digits error**

Specifically Modified Kintner's algorithm, is highly unstable as compared to the other methods, since it presents overflow errors (for $p = q$) and for any other combination of p, q, finite precision errors, that can destroy the algorithm by resulting to unreliable quantities. While the modified Kintner's algorithm has some interesting properties, according to its computational complexity compared with the conventional one [2], it is numerically more unstable. It is preferable to use the standard Kintner's method than the modified one, since the

benefits in computation speed are of less importance than the stability and reliability the radial polynomials being computed.

Prata's and q-recursive methods are more stable than the Kintner's type algorithms, by introducing less ill-posed subtractions for fewer conditions. Particularly, Prata's algorithm behaves unstably only for one condition (12), while q-recursive for the conditions (13) and (14). If one is to select between these more stable algorithms and by taking into account that q-recursive algorithm is very fast, its complexity is of an order lower than the other ones [2], the q-recursive algorithm is a good choice.

While the overflow errors can be handled, by restricting to low moment orders up to 20, the finite precision errors could be probably overcome by modifying the algorithm definitions and introducing algorithms that tackle both the computational efficiency and the stability ensuring.

5. Conclusions

A detailed study of the numerical stability of some very popular recursive algorithms for the fast computation of Zernike moments has taken place in the previous sections. The appropriate conditions in which each algorithm falls in unstable states, where the computed radial polynomials take unreliable values were defined. The above investigation draws the useful conclusion that all the algorithms analysed, present the possibility to alter their numerical stability, under certain conditions.

The analysis presented in this work constitutes a first attempt to compare these algorithms not according to their computational complexity, as this task has been already done, but in respect to their numerical behaviour.

Conclusively, one should take under consideration not only the mathematical properties of the algorithm, but also the numerical implementation restrictions, in each development of any recursive algorithm.

References

[1] M. Teague, Image analysis via the general theory of moments, J. Opt. Soc. Am. 70 (8) (1980) 920–930.
[2] C.W. Chong, P. Raveendran, R. Mukundan, A comparative analysis of algorithms for fast computation of Zernike moments, Pattern Recogn. 36 (2003) 731–742.
[3] C.W. Chong, P. Raveendran, R. Mukundan, Translation invariants of Zernike moments, Pattern Recogn. 36 (2003) 1765–1773.
[4] C.W. Chong, P. Raveendran, R. Mukundan, Translation and scale invariants of Legendre moments, Pattern Recogn. 37 (2004) 119–129.
[5] C.W. Chong, P. Raveendran, R. Mukundan, The scale invariants of pseudo-Zernike moments, Pattern Anal. Appl. 6 (2003) 176–184.
[6] R. Mukundan, S.H. Ong, P.A. Lee, Image analysis by Tchebichef moments, IEEE Trans. Image Process. 10 (9) (2001) 1357–1364.
[7] R. Mukundan, S.H. Ong, P.A. Lee, Discrete vs. continuous orthogonal moments for image analysis, in: Proceedings of International Conference on Imaging Science Systems and Technology, vol. 1, 2001, pp. 23–29.
[8] P.T. Yap, P. Raveendran, S.H. Ong, Image analysis by Krawtchouk moments, IEEE Trans. Image Process. 12 (11) (2003) 1367–1377.
[9] S.X. Liao, M. Pawlak, On the accuracy of Zernike moments for image analysis, IEEE PAMI 20 (12) (1998) 1358–1364.
[10] S. Rodtook, S.S. Makhanov, Numerical experiments on the accuracy of rotation moment invariants, Image Vis. Comput. 23 (2005) 577–586.
[11] N.K. Kamila, S. Mahapatra, S. Nanda, Invariance image analysis using modified Zernike moments, Pattern Recogn. Lett. 26 (2005) 747–753.
[12] G.A. Papakostas, Y.S. Boutalis, C.N. Papaodysseus, D.K. Fragoulis, Numerical error analysis in Zernike moments computation, Image Vis. Comput. 24 (2006) 960–969.
[13] A. Khotanzad, J.-H. Lu, Classification of invariant image representations using a neural network, IEEE Trans. Acoust., Speech Sign. Process. 38 (6) (1990) 1028–1038.
[14] A. Khotanzad, Y.H. Hong, Invariant image recognition by Zernike moments, IEEE Trans. Pattern Anal. Machine Intell. PAMI-12 (5) (1990) 489–497.
[15] G.A. Papakostas, D.A. Karras, B.G. Mertzios, Image coding using a wavelet based Zernike moments compression technique, in: 14th International Conference on Digital Signal Processing (DSP2002), vol. II, 1–3 July 2002, Santorini-Hellas, Greece, pp. 517–520.
[16] G.A. Papakostas, Y.S. Boutalis, B.G. Mertzios, Evolutionary selection of Zernike moment sets in image processing, in: 10th International Workshop on Systems, Signals and Image Processing (IWSSIP'03), 10–11 September 2003, Prague, Czech Republic.
[17] G.A. Papakostas, D.A. Karras, B.G. Mertzios, Y.S. Boutalis, An efficient feature extraction methodology for computer vision applications using wavelet compressed Zernike moments, ICGST International Journal on Graphics, Vision and Image Processing, Special Issue: Wavelets and Their Applications SI1 (2005) 5–15.
[18] M. Zhenjiang, Zernike moment-based image shape analysis and its application, Pattern Recogn. Lett. 21 (2) (2000) 169–177.

[19] C. Kan, M.D. Srinath, Invariant character recognition with Zernike and orthogonal Fourie–Mellin moments, Pattern Recogn. 35 (1) (2002) 143–154.
[20] T.W. Lin, Y.F. Chou, A comparative study of Zernike moments for image retrieval, in: Proceedings of 16th IPPR Conference on Computer Vision, Graphics and Image Processing (CVGIP 2003), 2003, pp. 621–629.
[21] D.G. Sim, H.K. Kim, R.H. Park, Invariant texture retrieval using modified Zernike moments, Image Vision Comput. 22 (4) (2004) 331–342.
[22] F. Zernike, Beugungstheorie des Schneidenverfahrens und seiner verbesserten Form, der Phasenkonstrastmethode, Physica 1 (1934) 689–701.
[23] E.C. Kintner, On the mathematical properties of the Zernike polynomials, Opt. Acta 23 (8) (1976) 679–680.
[24] E.C. Kintner, A recurrence relation for calculating the Zernike polynomials, Opt. Acta 23 (6) (1976) 499–500.
[25] A. Prata, W.V.T. Rusch, Algorithm for computation of Zernike polynomials expansion coefficients, Appl. Opt. 28 (1989) 749–754.
[26] C.N. Papaodysseus, E.B. Koukoutsis, C.N. Triantafyllou, Error sources and error propagation in the Levinson–Durbin algorithm, IEEE Trans. Signal Process. 41 (4) (1993).
[27] C.N. Papaodysseus, G. Carayannis, E.B. Koukoutsis, E. Kayafas, Comparing LS FIR filtering and l-step ahead linear prediction, IEEE Trans. Signal Process. 41 (2) (1993).
[28] C. Papaodysseus, E. Koukoutsis, C. Vassilatos, Error propagation and methods of error correction in LS FIR, IEEE Trans. Signal Process. 42 (5) (1994).
[29] C. Papaodysseus, E. Koukoutsis, G. Stavrakakis, C.C. Halkias, Exact analysis of the finite precision error generation and propagation in the FAEST and the fast transversal algorithms: A general methodology for developing robust RLS schemes, Math. Comput. Simulat. 44 (1997) 29–41.
[30] Y. Boutalis, C. Papaodyseus, E. Koukoutsis, A new multichannel recursive least squares algorithm for very robust and efficient adaptive filtering, J. Algorithms 37 (2000) 283–308.
[31] C. Papaodysseus, C. Alexiou, Th. Panagopoulos, G. Rousopoulos, D. Kravaritis, A novel general methodology for studying and remedying finite precision error with application in Kalman filtering, Stochastic Environ. Res. Risk Assess. 17 (2003) 1–19.