Ain Shams Engineering Journal 15 (2024) 102657

Contents lists available at ScienceDirect

Ain Shams Engineering Journal

journal homepage: https://www.sciencedirect.com

Full Length Article

Performance enhancement of high degree Charlier polynomials using multithreaded algorithm

Basheera M. Mahmmod [a], Wameedh N. Flayyih [a], Sadiq H. Abdulhussain [a], Firas A. Sabir [a], Bilal Khan [b,*], Muntadher Alsabah [e], Abir Hussain [c,d]

[a] Department of Computer Engineering, University of Baghdad, Iraq
[b] School of Computer Science and Engineering, California State University San Bernardino, San Bernardino, United States
[c] School of Computer Science and Mathematics, Liverpool John Moores University, UK
[d] Department of Electrical Engineering, University of Sharjah, Sharjah, United Arab Emirates
[e] Medical Technical College, Al-Farahidi University, Baghdad 10071, Iraq

ARTICLE INFO

Keywords:
High-degree polynomial
Balanced processing
Charlier polynomials
Charlier moments
Multithread processing
Orthogonal polynomials

ABSTRACT

Discrete orthogonal polynomials (DOPs) have gained significant research attention owing to their crucial role in digital signal processing applications such as computer vision, pattern recognition, and compression. However, the computation of DOP coefficients often incurs a substantial computational burden, exacerbating for higher-degree moments along with the resulting numerical errors. To address this challenge, this paper exploits the inherent parallelism in Charlier polynomial coefficient calculations to achieve enhanced polynomial performance. Independent calculations are distributed among threads, making efficient use of the available processing resources. Two algorithms are presented, the first algorithm evenly distributes the rows in a sequential manner (straightforward). Additionally, to achieve a more equitable distribution of coefficient calculations, this paper proposes alternative distribution approaches, aimed at balancing processing load among threads. Through extensive comparative experiments, we have confirmed that the proposed approaches achieved high performance across different degrees (1540 to 7370) and at different numbers of threads (2 to 256). The results show processing time in the multithreaded case is improved by up to 9.1 times with respect to the unthreaded case. Furthermore, by increasing the number of threads from 2 to 256, the trend indicates that the most significant improvement occurs in the range of 32 to 128 threads, confirming the robustness of the proposed algorithm. These findings signify the importance of this paper.

1. Introduction

Discrete orthogonal polynomials (DOPs) play a pivotal role in various scientific domains, particularly in digital signal analyses, including image and speech processing [1–4]. DOPs are invaluable tools for tackling a wide range of challenges because they are able to represent data without redundancy, do not necessitate prior information, and exhibit robustness against noise [5]. In addition, these polynomials are extensively used in a variety of applications, including edge detection [6–8], information concealment [9], speech enhancement [10,11], face recognition [12,13], and video content analysis [14]. Basis functions of orthogonal polynomials (OPs) can be utilized to estimate the solution of differential equations [15,16]. It is worth noting that by utilizing the characteristics of orthogonal polynomials to extract features and reduce dimensionality, they find utility in different applications, including data clustering [17,18], tracking [19], and action recognition [20]. Additionally, orthogonal polynomials can be used as basis functions or approximation tools within the genetic algorithm to improve the accuracy and efficiency of the optimization process [21–24]. Researchers have employed Charlier polynomials in various contexts. In [25], the speech enhancement algorithm is presented using Charlier polynomials. Li et al. [26] applies Charlier polynomials to reduce the model degree of discrete-time bilinear systems, while, Kang-Li et al. [27] utilize Charlier polynomials for model reduction of discrete time-delay systems.

* Corresponding author.
E-mail addresses: basheera.m@coeng.uobaghdad.edu.iq (B.M. Mahmmod), wam.nazar@coeng.uobaghdad.edu.iq (W.N. Flayyih), sadiqhabeeb@coeng.uobaghdad.edu.iq (S.H. Abdulhussain), firas.a.saber@coeng.uobaghdad.edu.iq (F.A. Sabir), bilal.khan@csusb.edu (B. Khan), muntadher.m@uoalfarahidi.edu.iq (M. Alsabah), A.Hussain@ljmu.ac.uk, abir.hussain@sharjah.ac.ae (A. Hussain).

https://doi.org/10.1016/j.asej.2024.102657
Received 3 April 2023; Received in revised form 11 January 2024; Accepted 14 January 2024
Available online 20 February 2024

The computation of DOP coefficients (DOPCs) is time-consuming and subject to numerical errors [28], primarily attributed to the involvement of Gamma functions and hypergeometric series. Consequently, DOPCs are typically computed using the three-term recurrence (TTR). Numerous studies [29,30] have shown that the values of DOPC provide numerical propagation error for high-degrees with (>172). To this end, novel recurrence techniques have been proposed to address propagation errors for discrete Chebyshev and Krawtchouk polynomials [31,32].

Charlier polynomials constitute a family of orthogonal polynomials defined on the real line with respect to the Poisson distribution. Notable characteristics of Charlier polynomials include their three-term recurrence relation, which involves the coefficients of the preceding polynomial terms, and their orthogonality property concerning the Poisson distribution. These polynomials are commonly used in approximation theory, probability theory, and quantum mechanics, offering solutions to various mathematical problems and facilitating the analysis of statistical distributions [33–35].

The recurrence relation and moment computing algorithms have been extensively explored for Charlier polynomials (CHPs). While recurrence algorithms have been proposed in both n- and x-directions, they encounter limitations when generating high-degree polynomials. This limitation arises from the initial values and the number of recurrence times, making the generation impractical. Some algorithms have attempted to lower the computational cost of Charlier moments (CHMs) [28] by utilizing either the n- or x-direction recurrence algorithm.

Recently, a recurrence technique based on the Gram-Schmidt orthonormalization process (GSOP) has been introduced [36]. However, this method comes with a high calculation cost due to its involvement in multiple iterations and inner products. In a previous work [28], the authors addressed the issue of size and reduced the computational cost by identifying suitable initial values for various CHP parameter values (a) and leveraging symmetry relations. However, they have stated that further reductions in computational cost were needed to enhance CHP generation performance. To date, efficiency and error minimization have predominantly been focused in the related literature [37]. However, performance, on the other hand, is a key issue in CHP computation in high-degree polynomials. Motivated by the challenges raised, the present study aims to provide an improved recurrence relation that efficiently computes CHPs for higher-degree polynomials. The main objective is to explore the available parallelism in the CHP computations. To our knowledge, no previous research has considered the integration of multithreading in CHP computation. In essence, the coefficients are distributed among independent threads to enable parallel processing, and the results are finally combined together. Various threads are considered to identify the optimal outcome. Furthermore, the exploitation of symmetry properties along the primary diagonal serves to reduce the number of coefficients requiring processing, thereby further reducing the overall processing time.

2. Materials and methods

In this section, the materials and the details of our methodology will be introduced. First, the preliminaries are provided, and then, the functions of CHP and their moments are described. Finally, the methodology of our proposed approach is explained in detail.

2.1. Preliminaries

CHPs and their moments (CHMs) have been widely used in different applications [38] such as image classification, compressive sensing, compression, and encryption. This section provides the definition of CHPs as well as the calculation of CHMs. Existing TTR relationships are also discussed. The definition of CHP $\tilde{C}_n^a(x)$ for the nth degree (order) is derived from [34] as given below

$$\tilde{C}_n^a(x) = {}_2F_0\left(\begin{matrix}-n,-x\\-\end{matrix}\middle|-\frac{1}{a}\right), \quad n,x=0,1,\ldots,N,\ a>0, \tag{1}$$

where a represents the parameter of the CHP which influences the distribution of moments, parameter n represents the degree of the polynomials, x is the signal index, N denotes the polynomial size, and ${}_2F_0$ is the hypergeometric series [34]. It is noteworthy that the degree of orthogonal polynomials directly impacts the level of detail and accuracy in representing the signal. Higher-degree orthogonal polynomials allow for finer resolution and more precise representation of the signal, capturing intricate details and variations. On the other hand, lower-degree orthogonal polynomials provide a coarser representation that may not capture fine details but can still convey the overall structure of the signal. The choice of the orthogonal polynomial degree depends on the application task, computational considerations, and the need to address potential numerical instabilities.

The normalized and weighted CHP can be expressed as:

$$\tilde{C}_n^a(x) = \sum_{k=0}^{\infty} \frac{(-n)_k\,(-x)_k}{k!}\left(-\frac{1}{a}\right)^k. \tag{2}$$

CHP satisfies the orthogonality conditions as explained in [39]:

$$\sum_{x=0}^{\infty} \tilde{C}_n^a(x)\,\tilde{C}_m^a(x) = a^{-n}\,e^a\,n!\,\delta_{nm}, \tag{3}$$

where δ_{nm} denotes the Kronecker delta function.

The use of (2) to compute CHP coefficients (CHPCs) causes numerical instability because of the binomial and gamma functions. Therefore, to address this issue a WNCHP (weighted and normalized CHP) is proposed. The WNCHP of degree n can be written as

$$C_n^a(x) = \sqrt{\frac{\omega_C(x;a)}{\rho_C(n;a)}}\,\tilde{C}_n^a(x), \tag{4}$$

where $\omega_C(x;a)$ and $\rho_C(n;a)$ represent the weight and norm functions of the CHPs.

Charlier Moments, also known as Charlier transform coefficients, are scalar values used as effective tools for object representation and image analysis. These moments provide a concise and non-redundant description of signals, enabling the characterization and analysis of objects in various domains, such as computer vision and pattern recognition. By capturing essential information, Charlier moments enable efficient representation and meaningful analysis of signals (1D such as speech, and 2D such as image). For a 2D signal $f(x,y)$ with a size of $N_1 \times N_2$, 2D CHMs ϕ_{nm} can be calculated as follows:

$$\phi_{nm} = \sum_{x=0}^{N_1-1}\sum_{y=0}^{N_2-1} f(x,y)\,C_n^a(x)\,C_m^a(y), \tag{5}$$

$n = 0,1,\ldots,Ord_1$ and $m = 0,1,\ldots,Ord_2$,

where Ord_1 and Ord_2 represent the maximal signal characterization degree. The procedure for reconstructing the 2D signal $\bar{f}(x,y)$ from CHMs is as follows:

$$\bar{f}(x,y) = \sum_{n=0}^{Ord_1-1}\sum_{m=0}^{Ord_2-1} \phi_{nm}\,C_n^a(x)\,C_m^a(y), \tag{6}$$

$x = 0,1,\ldots,N_1-1$ and $y = 0,1,\ldots,N_2-1$.

When all moments are employed in the reconstruction procedure, the reconstructed signal is $\bar{f}(x,y) = f(x,y)$.

2.2. Methodology

In this section, the proposed algorithm used to compute CHP is introduced. For more clarification, Fig. 1 depicts the proposed recurrence

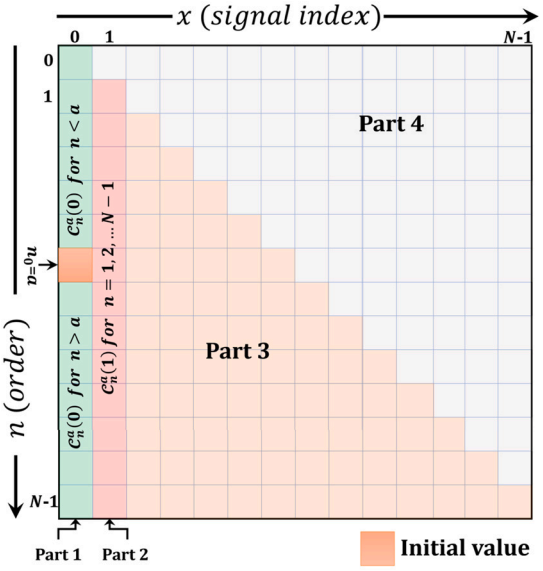


Fig. 1. CHPs parts of the proposed methodology.

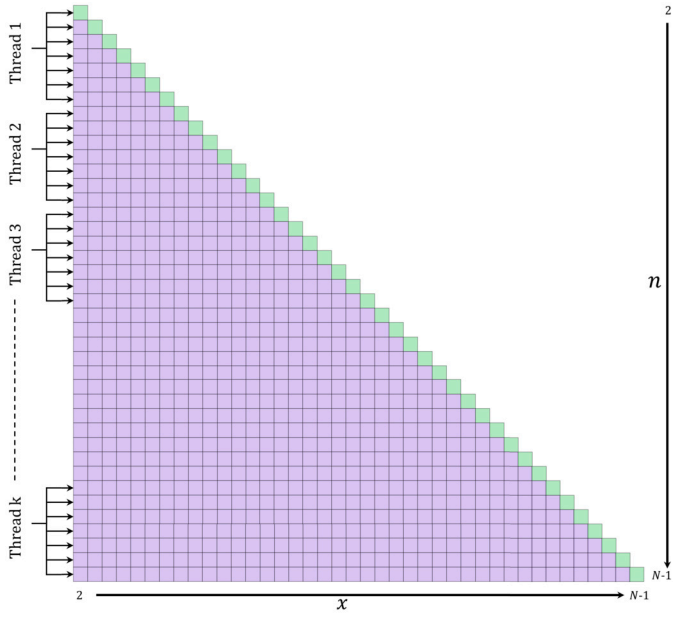


Fig. 2. Distribution of unbalanced threads.

algorithm. The first step is to compute an initial value at location n_0 [28]:

$$C_{n_0}^a(0) = \sqrt{e^{(-a+(a+1)log(a)-log\Gamma(a))}}, \tag{7}$$

where $n_0 = a$ (see Fig. 1). This enables the computation for a broad range of parameter values (a) as well as for high-degrees polynomial [28].

To compute the rest of the CHP values at $x = 0$ ($C_n^a(0)$), the two-term recurrence relation is utilized. This computation involves two distinct ranges, as illustrated in part 1 in Fig. 1. For the first range ($n < a$), i.e., $n = a-1, a-2, \ldots, 0$, the following formula can be used:

$$C_n^a(0) = \sqrt{\frac{n}{a}} C_{n+1}^a(0). \tag{8}$$

For the second range ($n > a$), i.e. $n = a+1, a+2, \ldots, N-1$, the following two-term recurrence method is used, which is given as

$$C_n^a(0) = \sqrt{\frac{n}{a}} C_{n-1}^a(0). \tag{9}$$

After computing the coefficients at $x = 0$, the coefficients at $x = 1$ (refer to part 2 in Fig. 1) are computed using the following two-term recurrence relation:

$$C_n^a(1) = \frac{a-n}{\sqrt{n}} C_n^a(0). \tag{10}$$

To compute the values of CHP in the range $n = 0, 1, \ldots, N-1$, and $x = 2, 3, \ldots, N-1$ (see part 3 and 4 in Fig. 1), we leverage a recently introduced symmetry relation (duality relation) described in [28]. This relation reduces the workload to ~50% of the coefficients. To adopt the duality relation, the coefficients in Part 3 are first computed using the recurrence relation, and then the symmetry relation is applied to compute the CHP coefficients in Part 4. The utilized symmetry relation can be expressed as

$$C_n^a(x) = C_x^a(n), \tag{11}$$

$n = 0, 1, \ldots, N-1$, and $x = 0, 1, \ldots, N-1$.

Accordingly, the coefficients that need to be computed are in the range $x = 2, 3, \ldots, N-1$ and $n = x, x+1, \ldots, N-1$, which is the triangular stage. In this stage, the three terms recurrence relation can be used:

$$C_n^a(x+1) = \beta_1 C_n^a(x) + \beta_2 C_n^a(x-1), \tag{12}$$

where $x = 1, 2, \ldots, N-1$, and $n = x, x+1, \ldots, N-1$. However, this relation is only applied to part 3 (see Fig. 1).

$$\begin{aligned} \beta_1 &= \frac{a-n+a}{\sqrt{a(x+1)}}, \\ \beta_2 &= -\sqrt{\frac{x}{(x+1)}}. \end{aligned} \tag{13}$$

2.3. Exploiting the parallelism in the triangular region

Multithread processing refers to a computing approach that involves the simultaneous execution of multiple threads within an algorithm. Multithreading allows for concurrent execution of multiple tasks, where each task runs independently and concurrently, potentially utilizing separate processor cores or computing resources [40]. In the proposed algorithm, the coefficients in each row can be calculated independently from the other rows. This allows parallel computations of the rows by distributing them among threads. These threads can run simultaneously on different cores, which achieves parallelism.

Each thread will be given a number of rows to process, and since data in each row is independent of other rows, data consistency is preserved. The final results will be forwarded when all threads complete their calculations and the complete matrix is constructed. Two approaches are considered when distributing the rows among the threads. In the first approach, the $(N-2)$ rows are distributed among the k different threads in sequence as shown in Fig. 2, where each thread takes the same number of rows B, where $B = (N-2)/k$ represents the bunch of rows that each thread should process. Thread 1 is responsible for rows 1 to B, Thread 2 for rows $B+1$ to $2B$, and so on. It can be noticed that the number of coefficients in each row increases as the row number increases, where row 1 has the lowest number of coefficients. This approach leads to a non-uniform distribution of process load between threads since the number of coefficients to be calculated by the first thread will be the lowest whereas the last thread will have the highest load.

Mathematically, the number of coefficients that need to be computed for the ith thread is:

$$Th_i = (i-1)\left(\frac{N-2}{k}\right)^2 + \frac{1}{2}\left(\frac{N-2}{k}\right)^2, \tag{14}$$

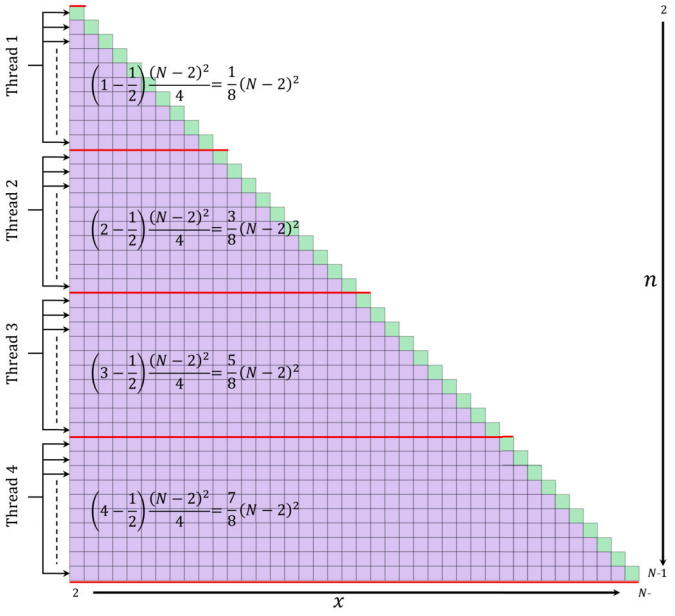


Fig. 3. UnBalanced approach coefficients distribution (four threads case study).

$$Th_i = \left(i-\frac{1}{2}\right)\left(\frac{N-2}{k}\right)^2, \tag{15}$$

where parameter Th_i denotes the ith thread, N denotes the polynomial size, and k represents the number of threads. For more clarification, Fig. 3 shows an example for $k=4$. It can be noticed that thread 1 computes $\frac{1}{2}\left(\frac{N-2}{4}\right)^2$ coefficients; whereas thread 4 computes $(4-1/2)\left(\frac{N-2}{4}\right)^2$.

It is worth noting that reducing the bunch size would increase the number of threads. This should enhance the performance but at the same time will increase the overhead induced by the thread generation and context switching.

The second approach aims to provide a balanced processing load between the threads by dividing the set of rows for each thread (see Fig. 4). Each thread will be given half the bunch size ($M=B/2$) from the top rows and the other half (M) from the bottom rows, in a symmetric manner. This ensures that each thread will have an almost identical processing load by having an equal number of coefficients to be calculated, namely $M(N-1)$ coefficients. This balance is independent of the number of threads, as can be given by the following for the ith thread:

$$Th_{i_{upper}} = (i-1)M^2 + 1/2M^2, \tag{16}$$

$$Th_{i_{lower}} = (2k-i-1)M^2 + 1/2M^2, \tag{17}$$

$$Th_i = 2(k-1)M^2 + M^2. \tag{18}$$

By substituting $k=\frac{N-2}{2M}$

$$Th_i = M(N-1), \tag{19}$$

$$Th_i = \frac{(N-2)(N-1)}{2k}, \tag{20}$$

which proves that the number of coefficients is independent of the thread number.

For the case of 4 threads ($k=4$) as shown in Fig. 5, each thread will be given $(N-2)/4$ rows arranged in $(N-2)/8$ rows from upper half and the same number from the bottom half, but with a different number of coefficients. Each thread will process $(N-2)(N-1)/8$ coefficients.

Since the data in the matrix is symmetric across the primary diagonal, this work proposes to write the second copy of each coefficient immediately after the first copy is written. This will take advantage of the availability of data that will be cached, thus reducing the memory access by avoiding reading from the memory at a later point in time. This symmetry is embedded in all steps including the threads of part 4 in Fig. 1.

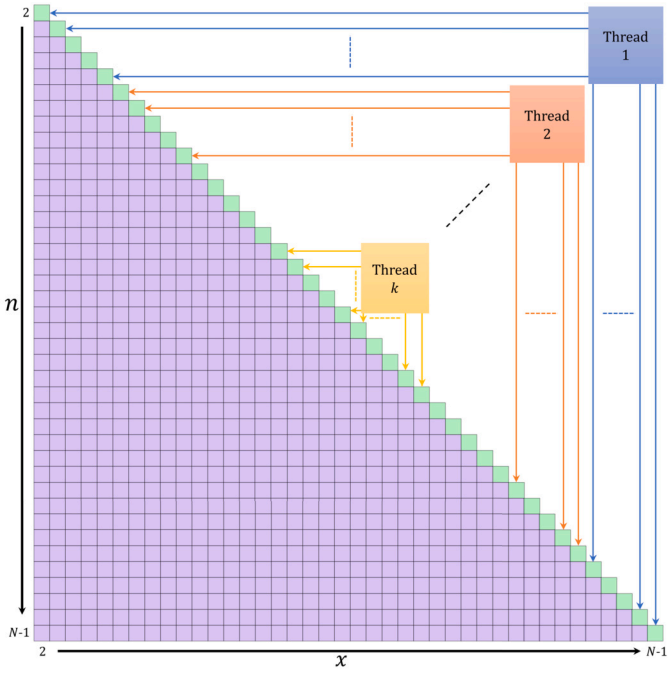


Fig. 4. Distribution of balanced threads.

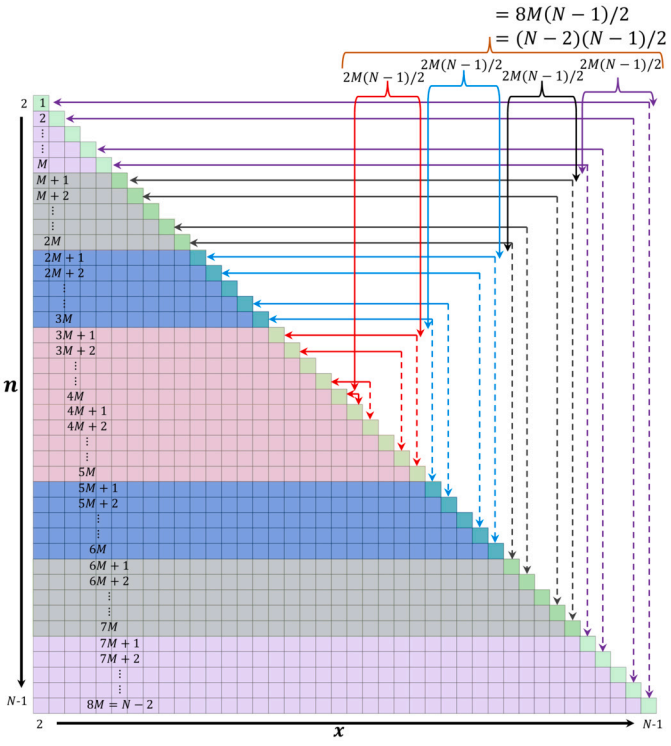


Fig. 5. Distribution of coefficients among four threads in the balanced approach ($k=4$).

3. Results and discussion

The proposed approaches were benchmarked with the unthreaded case in [28] for different polynomial sizes. Furthermore, the number of rows allocated to each thread (bunch size) is changed to see its effect on

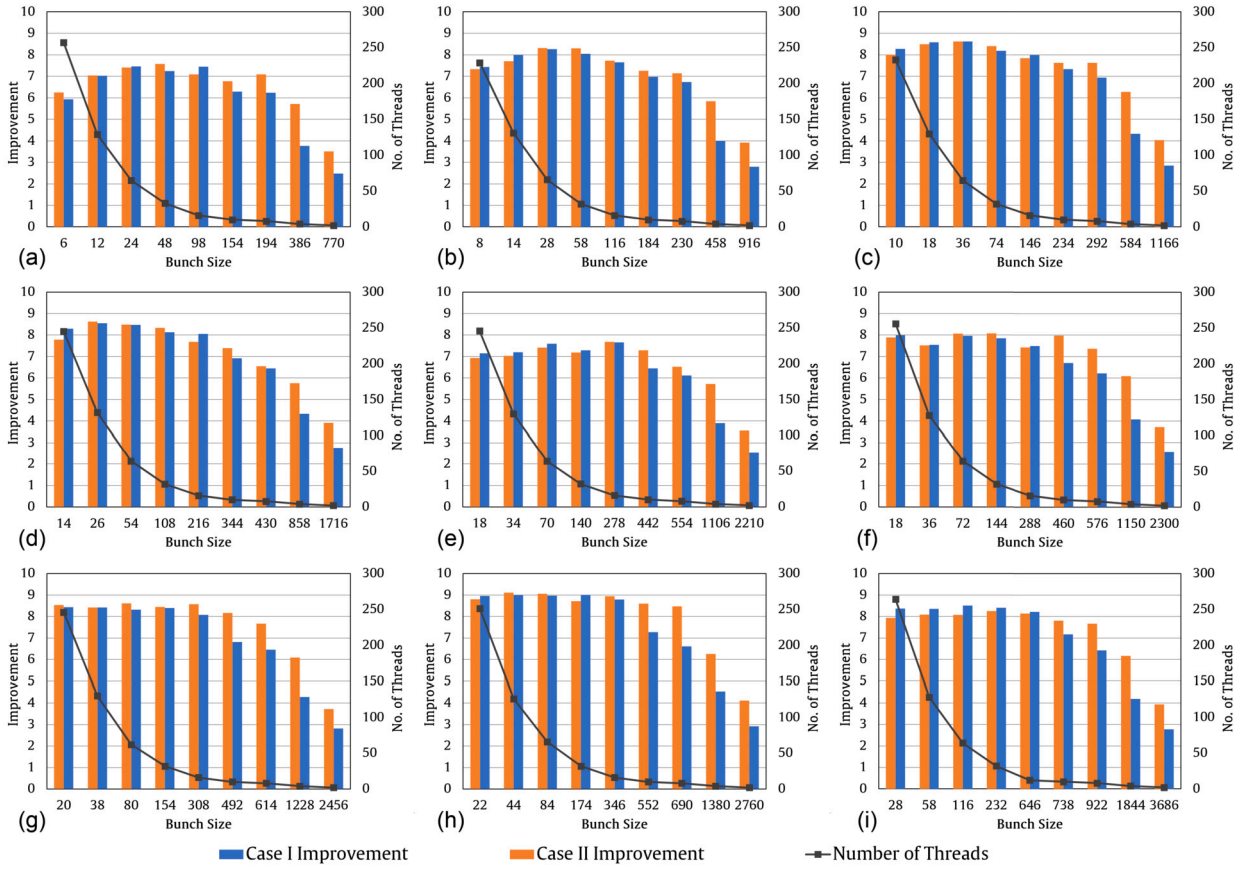


Fig. 6. Proposed algorithm improvement over the work in [12] for polynomial-size of (a) 1540, (b) 1830, (c) 2330, (d) 3430, (e) 4420, (f) 4600, (g) 4910, (h) 5520, and (i) 7370.

Algorithm 1 Compute CHP Coefficients using Multithreading.

Input: N, a, k
N Size of CHP, a parameter of CHP, k Number of threads
Output: $C_n^a(x)$ which represents the CHP.

$C_n^a(x) \leftarrow$ empty array of size $N \times N$
function COMPUTE_CHP_COEFFICIENTS(N, p, k)
Computing initial values for CHP
  Compute $C_n^a(0)$ for $n = a-1, a-2, \ldots, 0$ using (8)
  Compute $C_n^a(0)$ for $n = a+1, a+2, \ldots, N-1$ using (9)

Compute initial set at $x = 1$ and $n = 2, 3, \ldots, N-1$
  for n in range 2 to $N-1$ **do**
    Compute $C_n^a(1)$ using (10)
  end for
  Compute coefficients in the range $x = 0, 1$ and $n = 0, 1, \ldots, N-1$ using similarity relation using (11).
  if algorithm_choice = Unbalanced **then**
    Call ALGORITHM_UNBALANCED($N, a, k, C_n^a(x)$)
  else if algorithm_choice = Balanced **then**
    Call ALGORITHM_BALANCED($N, a, k, C_n^a(x)$)
  end if
end function

Algorithm 2 Compute CHP Coefficients using Unbalanced approach.

Input: N, a, k, $C_n^a(x)$
Output: $C_n^a(x)$

function ALGORITHM_UNBALANCED($N, p, k, C_n^a(x)$)
Generate threads
  $Thread\_pool \leftarrow$ thread pool with k threads
  Compute Bunch of Rows (B) $B = (N-2)/k$
  for i in range 0 to $k-1$ **do**
    Set the start row and end row for thread i
    $r_{start} = 2 + i \cdot B$ and $r_{end} = (i+1) \cdot B + 1$
    $rows = r_{start}$ to r_{end}
    $enqueue(Thread\_pool$, COMPUTE_COEFFICIENTS, $rows)$
  end for
  $wait(Thread\_pool)$ ▷ wait for all threads to complete
  return $C_n^a(x)$
end function
function COMPUTE_COEFFICIENTS($rows$)
  for k in range $rows$ **do**
    for x in range 2 to k **do**
      Compute coefficients using recurrence relation using (12).
      Compute coefficients using similarity relation using (11).
    end for
  end for
end function

the achieved improvement. As demonstrated in Fig. 6, the largest bunch sizes have not achieved the highest improvement. This comes from the fact that the resultant number of threads is very small which gives lower flexibility in making use of the processor time and resources. It is noteworthy that threads with a low number of coefficients as in thread 1 will finish the computations much faster than thread k, and there will be no other threads to utilize the idle resources that were allocated to thread 1. The improvement of the balanced approach (case II) clearly outperforms that of the unbalanced approach (case I) at large bunch sizes. This is consistent with the main objective of this approach since a large bunch size brings larger allocation differences between threads if this approach is not applied. Having a small number of unbalanced threads leads to non-optimal utilization of the processing resources and time, and this is what the second approach (balancing approach) was designed to resolve. In general, the maximum improvement in all cases ranged from 7.5 to 9.1 with respect to the non-threaded case.

Table 1
Size generated using the proposed algorithm and its improvement over existing algorithms.

a in terms of N	Value of a	TTR-n	TTR-x	Method in [28]	Proposed	Best improvement of the proposed and [28] over TTR-n and TTR-x
$N/6$	257	80	76	1540	1540	19.25
$N/5$	366	92	90	1830	1830	19.89
$N/4$	583	116	116	2330	2330	20.09
$N/3$	1114	168	162	3430	3430	20.42
$N/2$	3685	162	172	7370	7370	42.85
$2N/3$	3680	154	172	5520	5520	32.09
$3N/4$	3683	150	172	4910	4910	28.55
$4N/5$	3680	148	172	4600	4600	26.74
$5N/6$	3684	148	172	4420	4420	25.7

Algorithm 3 Compute CHP Coefficients using Balanced approach.

Input: N, a, k, $C_n^a(x)$
Output: $C_n^a(x)$
function ALGORITHM_BALANCED($N, p, k, C_n^a(x)$)
Generate threads
  $Thread\_pool \leftarrow$ thread pool with k threads
  Compute Bunch of Rows (B) $B = (N-2)/k$
  for i in range 0 to $k-1$ **do**
    Set the start row and end row for thread i
    $r_{start1} = 2 + i \cdot B/2$ and $r_{end1} = (i+1) \cdot B/2 + 1$
    $r_{start2} = N - 1 - (i+1) \cdot B/2$ and $r_{end2} = N - 1 - i \cdot B/2$
    $rows = r_{start1}$ to $r_{end1} \cup r_{start2}$ to r_{end2}
    $enqueue(Thread\_pool$, COMPUTE_COEFFICIENTS, $rows)$
  end for
  $wait(Thread\_pool)$ ▷ wait for all threads to complete
  return $C_n^a(x)$
end function
function COMPUTE_COEFFICIENTS($rows$)
  for k in range $rows$ **do**
    for x in range 2 to k **do**
      Compute coefficients using recurrence relation using (12).
      Compute coefficients using similarity relation using (11).
    end for
  end for
end function

On the other hand, a very small bunch size, which results in a large number of threads, achieved higher improvement as compared to large bunch sizes but was not the highest. Instead, the maximum improvement ranged from 32 threads to 128 threads as can be seen in Fig. 7.

This clearly indicates that a very high number of threads with a small workload does not give the optimal performance, which could be due to the imposed overhead of the threads generation with respect to the actual workload in each thread, in addition to the overhead of context switching. This is clear in the improvement drop in small polynomial sizes (Fig. 7 a to c) since the bunch size on the left side of the x-axis is small and the number of threads is high. This drop has less effect in the cases (d to f) and there is no drop in the higher polynomial size (g to i) since the bunch size is larger than in previous cases.

The same technique introduced in [41] is used here to evaluate the capacity of the proposed algorithm, which outperforms the existing algorithms in terms of maximum produced threads. Comparison results of the proposed method and the existing methods are listed in Table 1. The results include the maximum size for TTR-n, TTR-x, [28], and the proposed algorithms. The experiment is performed for different values of a for the CHP. The findings reveal that the proposed approach outperforms previous algorithms in terms of the size of CHP. In addition, the maximum size generated by the proposed algorithm and [28] are equivalent. The numbers reveal that for $a = N/6$, the minimum improvement is 19.25, and at $a = N/2$, the maximum improvement is 42.85.

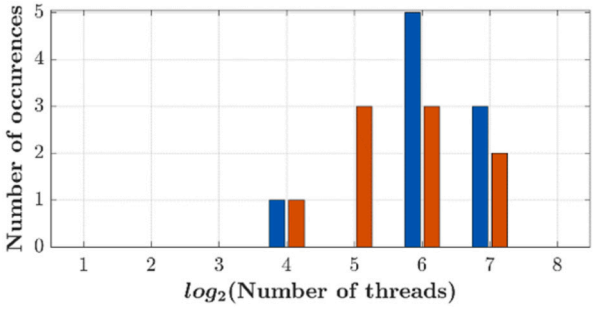


Fig. 7. Maximum improvement frequency distribution with respect to the number of threads.

4. Conclusion

The coefficients of Discrete orthogonal polynomials (DOPs) hold significant importance in digital signal processing applications such as computer vision. Calculating the coefficients of Charlier polynomials represents a performance bottleneck that this paper addressed by implementing multithreading. This threaded approach involves the distribution of independent coefficients among different threads, harnessing the parallel processing capabilities of multicore resources. Additionally, this paper takes into account, the symmetry across the primary diagonal to further enhance performance. Two distribution approaches have been proposed to divide the processing load represented by the coefficients calculations among the threads. The first approach assigns a consecutive set of rows to each thread, resulting in an unbalanced load distribution between threads. In contrast, the second approach aims to balance the load by allocating two subsets of rows to each thread. The results showed a substantial improvement in performance compared to the unthreaded case, reaching up to 9.1 times improvement. In general, the second approach achieved higher improvement, especially at a smaller number of threads due to its ability to highly utilize the processing resources. The results also confirmed that a very large number of threads, each having a small processing load, did not achieve the highest improvement due to the overheads induced. The maximum improvement was observed when the number of threads ranged from 32 to 128. Threading may also be applied to other discrete orthogonal polynomials after analyzing their characteristics and then deriving a suitable distribution approach to maximize the enhancement.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Pee C-Y, Ong S, Raveendran P. Numerically efficient algorithms for anisotropic scale and translation Tchebichef moment invariants. Pattern Recognit Lett 2017;92:68–74. https://doi.org/10.1016/j.patrec.2017.04.008.

[2] Abood ZI. Image compression using 3-d two-level techniques. J Eng 2013;19(11):1407–24.

[3] Abou-Loukh SJ, Abdul-Razzaq SM. Isolated word speech recognition using mixed transform. J Eng 2013;19(10):1271–86.

[4] Abou-Loukh SJ, Jameel JS. Compression of an ecg signal using mixed transforms. J Eng 2014;20(06):109–23.

[5] Abdulhussain SH, Ramli AR, Hussain AJ, Mahmmod BM, Jassim WA. Orthogonal polynomial embedded image kernel. In: Proceedings of the international conference on information and communication technology - ICICT'19, ICICT'19. New York, USA: ACM Press, New York; 2019. p. 215–21.

[6] Abdulhussain SH, Ramli AR, Mahmmod BM, Al-Haddad SAR, Jassim WA. Image edge detection operators based on orthogonal polynomials. Int J Image Data Fusion 2017;8(3):293–308. https://doi.org/10.1080/19479832.2017.1326405.

[7] Fahmy MM. Palmprint recognition based on mel frequency cepstral coefficients feature extraction. Ain Shams Eng J 2010;1(1):39–47.

[8] Alhawamda H, Taib B, Eshkuvatov Z, Ibrahim R. A new class of orthogonal polynomials for solving logarithmic singular integral equations. Ain Shams Eng J 2020;11(2):489–94.

[9] Radeaf HS, Mahmmod BM, Abdulhussain SH, Al-Jumaeily D. A steganography based on orthogonal moments. In: Proceedings of the international conference on information and communication technology - ICICT'19, ICICT'19. New York, USA: ACM, New York; 2019. p. 147–53.

[10] Mahmmod BM, Ramli AR, Baker T, Al-Obeidat F, Abdulhussain SH, Jassim WA. Speech enhancement algorithm based on super-Gaussian modeling and orthogonal polynomials. IEEE Access 2019;7:103485–504. https://doi.org/10.1109/ACCESS.2019.2929864.

[11] Hussein HA, Hameed SM, Mahmmod BM, Abdulhussain SH, Hussain AJ. Dual stages of speech enhancement algorithm based on super Gaussian speech models. J Eng 2023;29(09):1–13.

[12] Akhmedova F, Liao S. Face recognition with discrete orthogonal moments. In: Recent advances in computer vision. Springer; 2019. p. 189–209.

[13] Abdulhussain SH, Mahmmod BM, AlGhadhban A, Flusser J. Face recognition algorithm based on fast computation of orthogonal moments. Mathematics 2022;10(15):2721. https://doi.org/10.3390/math10152721. https://www.mdpi.com/2227-7390/10/15/2721.

[14] Abdulhussain SH, Rahman Ramli A, Mahmmod BM, Iqbal Saripan M, Al-Haddad S, Baker T, et al. A fast feature extraction algorithm for image and video processing. In: 2019 international joint conference on neural networks (IJCNN). IEEE; 2019. p. 1–8.

[15] Mizel AKE. Orthogonal functions solving linear functional differential equations using Chebyshev polynomial. Baghdad Sci J 2008;5(1):143–8.

[16] Beukers F. Special functions. Encyclopedia of mathematics and its applications, vol. 71. Cambridge University Pres. ISBN 0-521-62321-9, 1999. By George E. Andrews, Richard Askey and Ranjan Roy. 664 pp., £ 55.00 (us $85.00); Bull Lond Math Soc 2001;33(1):116–27.

[17] Nurhidayat I, Pimpunchat B, Noeiaghdam S, Fernández-Gámiz U. Comparisons of svm kernels for insurance data clustering. Emerg Sci J 2022;6(4):866–80.

[18] Abdulhussain SH, Mahmmod BM, Flusser J, AL-Utaibi KA, Sait SM. Fast overlapping block processing algorithm for feature extraction. Symmetry 2022;14(4):715. https://doi.org/10.3390/sym14040715. https://www.mdpi.com/2073-8994/14/4/715.

[19] Bagherzadeh SZ, Toosizadeh S. Eye tracking algorithm based on multi model Kalman filter. HighTech Innov J 2022;3(1):15–27.

[20] Sarabu A, Santra AK. Human action recognition in videos using convolution long short-term memory network with spatio-temporal networks. Emerg Sci J 2021;5(1):25–33.

[21] Abo-Hammour Z, Alsmadi O, Momani S, Abu Arqub O. A genetic algorithm approach for prediction of linear dynamical systems. Math Probl Eng 2013;2013.

[22] Abo-Hammour Z, Abu Arqub O, Momani S, Shawagfeh N. Optimization solution of Troesch's and Bratu's problems of ordinary type using novel continuous genetic algorithm. Discrete Dyn Nat Soc 2014:15.

[23] Abu Arqub O, Abo-Hammour Z, Momani S, Shawagfeh N. Solving singular two-point boundary value problems using continuous genetic algorithm. In: Abstract and applied analysis. Hindawi; 2012. p. 25.

[24] Arqub OA, Abo-Hammour Z. Numerical solution of systems of second-order boundary value problems using continuous genetic algorithm. Inf Sci 2014;279:396–415.

[25] Jerjees SA, Mohammed HJ, Radeaf HS, Mahmmod BM, Abdulhussain SH. Deep learning-based speech enhancement algorithm using Charlier transform. In: 2023 15th international conference on developments in eSystems engineering (DeSE). IEEE; 2023. p. 100–5.

[26] Li Y, Jiang Y, Yang P. Time domain model order reduction of discrete-time bilinear systems with Charlier polynomials. Math Comput Simul 2021;190:905–20.

[27] Xu K-L, Jiang Y-L, Li Z, Li L. Model reduction of discrete time-delay systems based on Charlier polynomials and high-order Krylov subspaces. Linear Algebra Appl 2023;661:222–46.

[28] Abdul-Hadi AM, Abdulhussain SH, Mahmmod BM. On the computational aspects of Charlier polynomials. Cogent Eng 2020;7(1):1763553. https://doi.org/10.1080/23311916.2020.1763553.

[29] Abdulhussain SH, Mahmmod BM, Baker T, Al-Jumeily D. Fast and accurate computation of high-order tchebichef polynomials. Concurr Comput Pract Exp Dec. 2022;34(27). https://doi.org/10.1002/cpe.7311. https://onlinelibrary.wiley.com/doi/10.1002/cpe.7311.

[30] Abdulhussain SH, Ramli AR, Al-Haddad SAR, Mahmmod BM, Jassim WA. Fast recursive computation of Krawtchouk polynomials. J Math Imaging Vis 2018;60(3):285–303. https://doi.org/10.1007/s10851-017-0758-9.

[31] Andrews GE, Askey R, Roy R. Special functions. Encyclopedia of mathematics and its applications, vol. 71. Cambridge: Cambridge University Press; 1999.

[32] AL-Utaibi KA, Abdulhussain SH, Mahmmod BM, Naser MA, Alsabah M, Sait SM. Reliable recurrence algorithm for high-order Krawtchouk polynomials. Entropy 2021;23(9):1162. https://doi.org/10.3390/e23091162.

[33] Ismail MEH. Classical and quantum orthogonal polynomials in one variable. Encyclopedia of mathematics and its applications, vol. 98. Cambridge: Cambridge University Press; 2005.

[34] Koekoek R, Lesky PA, Swarttouw RF. Hypergeometric orthogonal polynomials and their q-analogues. Springer monographs in mathematics. Berlin: Springer-Verlag; 2010.

[35] Nikiforov AF, Uvarov VB, Suslov SK. Classical orthogonal polynomials of a discrete variable. In: Springer series in computational physics. Berlin: Springer-Verlag; 1991. p. 18–54.

[36] Daoui A, Yamni M, El ogri O, Karmouni H, Sayyouri M, Qjidaa H. Stable computation of higher order Charlier moments for signal and image reconstruction. Inf Sci 2020;521:251–76. https://doi.org/10.1016/j.ins.2020.02.019.

[37] Razian SA, MahvashMohammadi H. Optimizing raytracing algorithm using cuda. Emerg Sci J 2017;1(3):167–78.

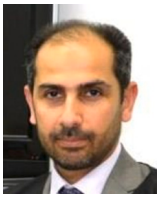
[38] Daoui A, Karmouni H, Sayyouri M, Qjidaa H. Efficient methods for signal processing using Charlier moments and artificial bee colony algorithm. Circuits Syst Signal Process 2022;41(1):166–95.

[39] Karmouni H, Hmimid A, Jahid T, Sayyouri M, Qjidaa H, Rezzouk A. Fast and stable computation of the Charlier moments and their inverses using digital filters and image block representation. Circuits Syst Signal Process 2018;37(9):4015–33. https://doi.org/10.1007/s00034-018-0755-2.

[40] Amamiya S, Amamiya M, Hasegawa R, Fujita H. A continuation-based noninterruptible multithreading processor architecture. J Supercomput 2009;47:228–52.

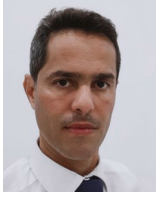
[41] Chong C-W, Raveendran P, Mukundan R. Translation and scale invariants of Legendre moments. Pattern Recognit 2004;37(1):119–29. https://doi.org/10.1016/j.patcog.2003.06.003.

Basheera M. Mahmmod received the Ph.D. degree in computer and embedded system engineering from UPM, in 2018. She is currently a Lecturer at the Department of Computer Engineering, University of Baghdad. Her research interests include signal processing, speech enhancement, and computer vision.



Wameedh N. Flayyih received the B.Sc. degree in computer engineering from University of Baghdad, Iraq, in 2001. He received the M.Sc. degree in computer and control engineering from University of Baghdad, Iraq, in 2004. In 2005, he joined the Computer Engineering Department in the University of Baghdad as a faculty member. His research interests include computer architecture, microprocessors, VLSI design, on-chip communication, and fault tolerance.

Sadiq H. Abdulhussain received the Ph.D. degree in computer and embedded system engineering from UPM, in 2018. Since 2008, he has been a Faculty Member with the University of Baghdad. His research interests include image processing, computer vision, and signal processing.



Firas A. Sabir received the PhD Degree in electrical and electronic engineering from the University of Technology, Baghdad, in 2008. He is an associate professor in the computer engineering department university of Baghdad. His areas of specialty and interest include the next generation of mobile communication, cloud radio access networks, signal processing, and self-organized networks.

Bilal Khan holds a Ph.D. in computer science (artificial intelligence) with the thesis title "Game theoretic coalitional routing in cooperative vehicular ad hoc networks", MSc in pervasive computing and Master of science in computer science. While serving as an assistant researcher at UCLA from 2013 to 2020, Dr. Khan developed the most robust and comprehensive decision support system as an online nanoinformatics platform to assist regulatory bodies for the management of nanotechnology. The nanoinformatics platform consists of machine learning and data mining models that are supported by the largest database of nanomaterials. Bilal maintained the platform via a high performance computing cluster that was designed from scratch (with 24 compute nodes and 115TB of storage space for high performance computations and simulations). At UCLA, Dr. Khan also led the development of a cyber-infrastructure for a virtual water district for efficient use and consumption of water in small rural agricultural communities. Additionally, Dr. Khan developed data-driven approaches (using machine and deep learning techniques) for water use patterns in the small communities as online applications.

Muntadher Alsabah received the B.S. degree in electrical engineering from the University of Baghdad, Iraq, in 2007, and the M.S. degree in wireless communication engineering from University Putra Malaysia (UPM), Malaysia, in 2014, and the Ph.D. degree in wireless communication from the University of Sheffield, Sheffield, U.K. in 2020. His current research interests include wireless communications, multiple-input-multiple-output (MIMO) systems, mmWave frequencies, image processing, signal processing and in applying the artificial intelligence and machine learning techniques in wireless communications systems.

Abir Hussain is currently a Professor of machine learning with the Department of Computer Science, Liverpool John Moores University. Her research interests include machine learning algorithms and their applications to medical, image and signal processing, and data analysis.