IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 8 May 2023, accepted 30 May 2023, date of publication 5 June 2023, date of current version 12 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3283028

RESEARCH ARTICLE

An Automatic Method for Generating Symbolic Expressions of Zernike Circular Polynomials

HONG-YAN ZHANG, YU ZHOU, (Member, IEEE), AND FU-YUN LI
School of Information Science and Technology, Hainan Normal University, Haikou 571158, China

Corresponding author: Hong-Yan Zhang (hongyan@hainnu.edu.cn)

This work was supported in part by the Hainan Provincial Natural Science Foundation of China under Grant 2019RC199, in part by the Hainan Provincial Education and Teaching Reform Project of Colleges and Universities under Grant Hnjg2019-46, and in part by the National Natural Science Foundation of China under Grant 62167003.

ABSTRACT Zernike circular polynomials (ZCP) play a significant role in optics engineering. The symbolic expressions for ZCP are valuable for theoretic analysis and engineering designs. However, there are still two problems which remain open: firstly, there is a lack of sufficient mathematical formulas of the ZCP for optics designers; secondly the formulas for inter-conversion of Noll's single index and Born-Wolf's double indices of ZCP are neither uniquely determinate nor satisfactory. An automatic method for generating symbolic expressions for ZCP is proposed based on five essential factors: the new theorems for converting the single/double indices of the ZCP, the robust and effective numeric algorithms for computing key parameters of ZCP, the symbolic algorithms for generating mathematical expressions of ZCP, and meta-programming & LaTeX programming for generating the table of ZCP. The theorems, method, algorithms and system architecture proposed are beneficial to both optics design process, optics software, computer-output typesetting in publishing industry as well as STEM education.

INDEX TERMS Zernike circular polynomial, symbolic computation, mathematical table, computer-output typesetting, LaTeX programming, STEM education.

I. INTRODUCTION

In optics engineering, the *Zernike circular polynomials* (ZCP), also named Zernike polynomials for simplicity, are essential for representing aberrations in imaging system, optics design and optical testing [1], [2], [3], [4], [5], [6], [7], [8], [9]. Mathematically, ZCP are a sequence of bivariate polynomials defined on the unit disk

$$\begin{aligned}\mathcal{D} &= \left\{(x, y) : 0 \le x^2 + y^2 \le 1\right\} \\ &= \{(\rho, \theta) : 0 \le \rho \le 1, 0 \le \theta \le 2\pi\}\end{aligned}$$

derived from the circular pupils of imaging system. Named after Frits Zernike, they play an important role in beam optics and optics design [10], atmospheric turbulence [11] and image processing [12]. The ZCP are orthogonal functions which represent balanced aberrations that yield minimum variance [9]. The computation of ZCP are significant for theoretic analysis and engineering applications. The numerical algorithms are discussed in lots of literature such as [4], [12], [13], and [14]. However, the symbolic computation for automatically generating the mathematical expressions for the ZCP is missing. It may be inconvenient or troublesome for the optics designers to look for the formulas of ZCP with "high" orders. Actually, there are less 28 $\sim$ 40 and 70 expressions for the ZCP in ZEMAX and CodeV respectively.

The purpose of this paper is to propose an automatic method for generating mathematical expressions for the ZCP. FIGURE 1 illustrates the framework of our work. There are five modules for the system architecture:

- Principle of Index Conversion, which includes our new theorems and proofs about the single/double index for the ZCP;
- Numeric computation, which computes key parameters robustly and effectively for large integers so as to avoid the overflow problem in computing factorials;
- Symbolic computation, which deals with the algorithms for automatic generating symbolic expressions of the ZCP;

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Selim Habib.

- Meta-programming and LaTeX programming, which generates the long table of ZCP and outputs the corresponding pdf file for printing, reading and reference;
- System setting, which sets the necessary information for the mathematical formula of ZCP and the geometric configuration of the long table of mathematical expressions.

The contents of this paper are organized as follows: Section II deals with the preliminaries of backgrounds; Section III copes with the principle of inter-conversion of the single/double index for computing ZCP; Section IV focuses on how to generate long table of mathematical expressions of ZCP; Section V is the conclusion.

II. PRELIMINARIES

A. FUNDAMENTALS OF MATHEMATICS

1) NOTATIONS FOR INTEGERS

For an integer $n \in \mathbb{Z} = \{0, \pm1, \pm2, \cdots\}$, it is even if and only if $2 \mid n$, i.e., 2 divides n or equivalently $n \equiv 0 \pmod 2$; otherwise, it is odd if and only if $2 \nmid n$ or equivalently $n \equiv 1 \pmod 2$. The sets of even and odd integers are denoted by

$$\mathbb{Z}_{\text{odd}} = \left\{i \in \mathbb{Z} : 2 \nmid i\right\} = \{\pm1, \pm3, \pm5, \cdots\}$$

and

$$\mathbb{Z}_{\text{even}} = \{i \in \mathbb{Z} : 2 \mid i\} = \{0, \pm2, \pm4, \pm6, \cdots\}$$

respectively. The set of positive integers are $\mathbb{N} = \{1, 2, 3, \cdots\}$ and the set of non-negative integers is $\mathbb{Z}^+ = \mathbb{N} \cup \{0\} = \{0, 1, 2, \cdots\}$. Similarly, the set of negative integers is $\mathbb{Z}^- = -\mathbb{N} = \{-1, -2, -3, \cdots\}$. In consequence, the sets of non-negative even integers and non-negative odd integers can be denoted by $\mathbb{Z}^+_{\text{even}}$ and $\mathbb{Z}^+_{\text{odd}}$ respectively.

For any real number $x \in \mathbb{R}$, the unique integer $n = \lfloor x \rfloor \in \mathbb{Z}$ such that $n \le x < n+1$ is called the floor of x. Similarly, the integer $n' = \lceil x \rceil \in \mathbb{Z}$ such that $n' - 1 < x \le n'$ is called the ceiling of x.

2) EXPRESSION OF GENERAL POLYNOMIALS

A polynomial $T_n(x)$ with argument x and degree n can be written by

$$T_n(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + \cdots + a_i x^i + \cdots + a_n x^n \tag{1}$$

If $a_i = 0$ for all odd i, then $T_n(x)$ is an even polynomial. Similarly, if $a_i = 0$ for all even i, then $T_n(x)$ is an odd polynomial. Futhermore, some of the coefficients $a_0, a_1, \cdots, a_n$ may be missing if they equal zero. In consequence, we can denote a polynomial as

$$T(x) = \sum_{s=0}^{k} c_s x^{p_s}, \quad p_s \in \mathbb{Z}^+ \tag{2}$$

where $p_s \in \mathbb{Z}^+$. In other words, a polynomial can be described by the sequence of coefficients $\mathtt{c} = \langle c_0, \cdots, c_k \rangle$ and the corresponding sequence of power indices $\mathtt{p} = \langle p_0, \cdots, p_k \rangle$.

3) BINOMIAL AND TRI-NOMIAL COEFFICIENTS

In general, for any real number $\alpha \in \mathbb{R}$ and non-negative $i \in \mathbb{Z}^+ = \{0, 1, 2, \cdots\}$ the generalized binomial expansion can be expressed by

$$(1+x)^\alpha = \sum_{i=0}^{\infty} \binom{\alpha}{i} x^i, \quad x \in \text{ROC} = (-1, 1)$$

in which ROC is the region of convergence and

$$\binom{\alpha}{i} = \frac{\alpha(\alpha-1)\cdots(\alpha-i+1)}{i!} \tag{3}$$

is the binomial coefficient [15], [16], [17]. Particularly, if $\alpha \in \mathbb{Z}^+$ is a positive integer, we have

$$\binom{\alpha}{i} = \frac{\alpha!}{i!(\alpha-i)!}. \tag{4}$$

For any $r \in \mathbb{Z}^+$ we have

$$(x_1 + x_2 + x_3)^r = \sum_{i_1+i_2+i_3=r} \binom{r}{i_1, i_2, i_3} x_1^{i_1} x_2^{i_2} x_3^{i_3}$$

where

$$\binom{r}{i_1, i_2, i_3} = \frac{(i_1+i_2+i_3)!}{i_1! i_2! i_3!}, \quad i_1 + i_2 + i_3 = r \tag{5}$$

is the tri-nomial coefficients, which is symmetric for a permutation of i_1, i_2, i_3 and can be expressed in terms of binomial coefficients:

$$\binom{r}{i_1, i_2, i_3} = \binom{i_1+i_2}{i_1}\binom{i_1+i_2+i_3}{i_1+i_2}. \tag{6}$$

Therefore, for $r = n-s, p = 3, i_1 = s, i_2 = k-s, i_3 = n-k-s, i_1+i_2+i_3 = n-s$, we can obtain

$$\binom{n-s}{s, k-s, n-k-s} = \binom{k}{s}\binom{n-s}{k}. \tag{7}$$

When computing the value of binomial coefficients with computer programs, we should keep an eye on the overflow problem since the factorial $r!$ increases rapidly with the integer r. We can avoid such a risk by reformulating the binomial coefficient properly, i.e., for any $i \in \mathbb{Z} = \{0\} \cup \mathbb{N}$,

$$\binom{\alpha}{i} = \begin{cases} 1, & i = 0; \\ \prod_{t=0}^{i-1} \frac{\alpha - t}{i - t}, & i \ge 1. \end{cases} \tag{8}$$

The value of $\binom{\alpha}{i}$ can be computed with **Algorithm** 4 robustly and fastly.

4) KRONECKER SYMBOL

The notation

$$\delta_{ij} = \begin{cases} 0, & i \ne j \\ 1, & i = j \end{cases} \tag{9}$$

is called the Kronecker symbol. Particularly, $\delta_{m0} = 1$ for $m = 0$ and $\delta_{m0} = 0$ otherwise.

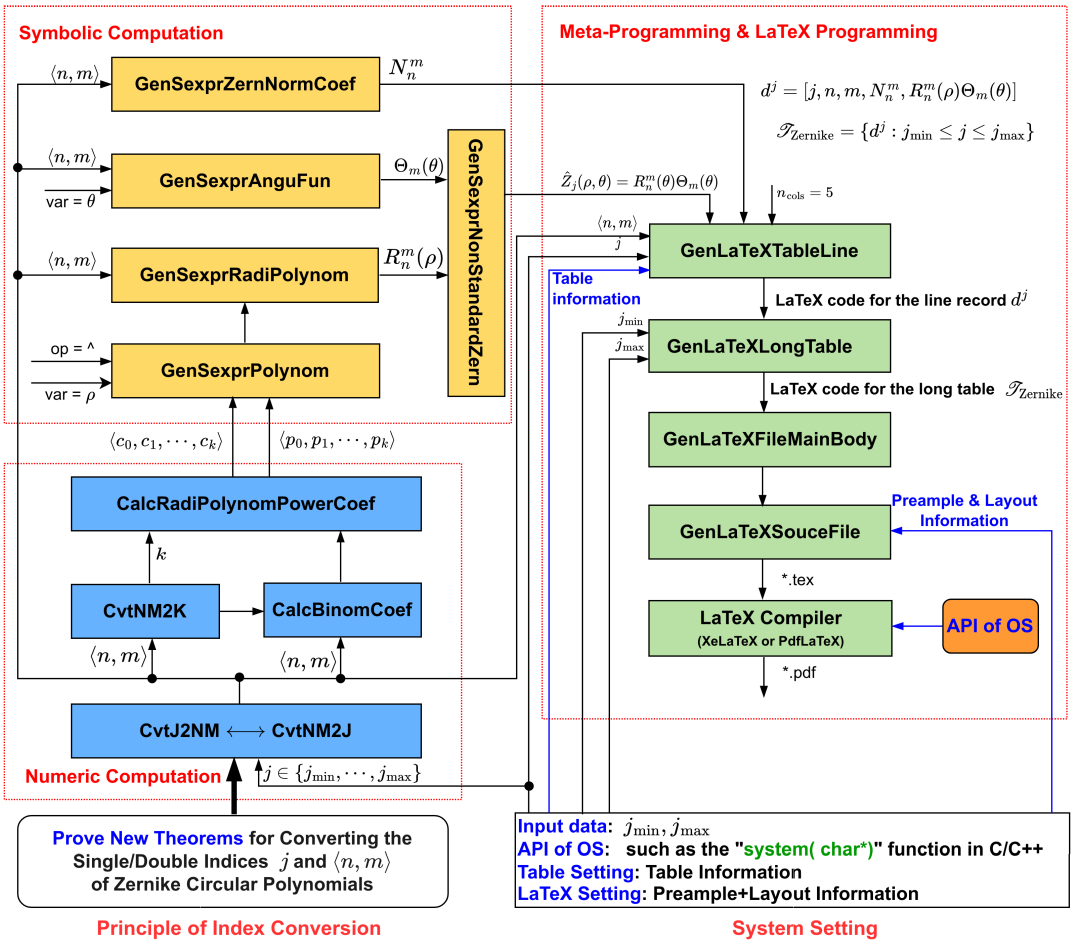

FIGURE 1. System architecture for automatically generating a long table of the mathematical expressions of ZCP $\left\{\mathrm{Z}_j(\rho,\theta) : j_{\min} \le j \le j_{\max}\right\}$.

5) DISCRETE UNIT STEP FUNCTION

The discrete unit step function is defined by

$$u(m) = \begin{cases} 1, & m \in \mathbb{Z}^+ = \{0, 1, 2, 3, \cdots\} \\ 0, & m \in \mathbb{Z}^- = \{-1, -2, -3, \cdots\} \end{cases} \tag{10}$$

with the counterpart of Heaviside function in the continuous case.

6) CARDINALITY

For a finite set $A = \{x_1, \cdots, x_r\}$, its cardinality is denoted by $|A|$, i.e.

$$|A| = |\{x_1, \cdots, x_r\}| = r. \tag{11}$$

B. ZERNIKE CIRCULAR POLYNOMIALS

1) DEFINITION OF ZERNIKE CIRCULAR POLYNOMIALS

Generally, for $n \in \mathbb{Z}^+$ and $m \in \mathbb{Z}$ such that $|m| \le n$ and $2 \mid (n-m)$, the ZCP can be denoted by [1], [3], [4]

$$\mathrm{Z}_n^m(\rho,\theta) = N_n^m \, \mathrm{R}_n^m(\rho)\Theta_m(\theta), \quad \rho \in [0,1], \theta \in [0, 2\pi] \tag{12}$$

where

$$N_n^m = \sqrt{\frac{2(n+1)}{1+\delta_{m0}}} = \begin{cases} \sqrt{2(n+1)}, & m \neq 0 \\ \sqrt{n+1}, & m = 0 \end{cases} \tag{13}$$

is the coefficients for normalization,

$$\Theta_m(\theta) = \begin{cases} 1, & m = 0, j \in \mathbb{Z}_{\text{odd}}^+ \\ \cos(|m|\,\theta), & m \neq 0, j \in \mathbb{Z}_{\text{even}}^+ \\ \sin(|m|\,\theta), & m \neq 0, j \in \mathbb{Z}_{\text{odd}}^+ \end{cases} \tag{14}$$

is the angular function with equivalent complex form $\mathrm{e}^{\mathrm{i}|m|\theta}$,

$$\begin{aligned} \mathrm{R}_n^m(\rho) &= \frac{1}{(\frac{n-|m|}{2})!\rho^{|m|}} \left[\frac{\mathrm{d}}{\mathrm{d}(\rho^2)}\right]^{\frac{n-|m|}{2}} \left[(\rho^2)^{\frac{n+|m|}{2}}(\rho^2-1)^{\frac{n-|m|}{2}}\right] \\ &= \sum_{s=0}^{\frac{n-|m|}{2}} (-1)^s \cdot \frac{(n-s)!}{s!\left(\frac{n+|m|}{2}-s\right)!\left(\frac{n-|m|}{2}-s\right)!} \cdot \rho^{n-2s} \\ &= \sum_{s=0}^{k} (-1)^s \cdot \binom{n-s}{s,\, k-s,\, n-k-s} \cdot \rho^{n-2s} \end{aligned}$$

$$= \sum_{s=0}^{k} c_s \rho^{p_s} = c_0\rho^n + c_1\rho^{n-2} + \cdots + c_k\rho^{n-2k} \tag{15}$$

is the radial Zernike polynomials in which

$$k = \frac{1}{2}(n - |m|) \tag{16}$$

and

$$\begin{cases} c_s = (-1)^s \binom{n-s}{s,\, k-s,\, n-k-s}, & 0 \le s \le k; \\ p_s = n - 2s, \quad 0 \le s \le k. \end{cases} \tag{17}$$

Substituting (7) into (17), we obtain the following efficient expression for computing c_s

$$c_s = (-1)^s \binom{k}{s}\binom{n-s}{k}, \quad 0 \le s \le k. \tag{18}$$

2) SINGLE AND DOUBLE INDICES FOR ZERNIKE CIRCULAR POLYNOMIALS

There are different schemes for representing the order of ZCP in theory analysis and applications. The first scheme is the *Born-Wolf notation* (BWN), which uses the double indices $\langle n, m\rangle$ in the expression $\mathrm{Z}_n^m(\rho, \theta)$. This kind of notation is well known in the famous book by Born-Wolf [1]. The BWN is also named with OSA notation.

The second scheme is characterized by a single index j. However, there are three choices for the single index:

- ANSI "Standard Zernike Polynomials". In this case $j_{\text{ANSI}} = \frac{n(n+2)+m}{2}$.
- ZEMAX "Standard Zernike Coefficients". This is introduced by Noll in studying atmospheric turbulence [11], in which j is used to represent the order instead of n and m. In the software ZEMAX for optics design, the Noll's notation j is taken.
- ZEMAX "Zernike Fringe Polynomials",[1] which is introduced by James C. Wyant [18].

In this paper, we just concern the Noll's Z_j and the Born-Wolf's Z_n^m due to their wide applications in optics design where the relation between $\langle n, m\rangle$ and j is well known for optics engineers. In this sense, we have [10], [11]

$$\mathrm{Z}_j(\rho, \theta) = \mathrm{Z}_n^m(\rho, \theta), \quad j = j(n, m) \in \mathbb{N} \tag{19}$$

such that

$$\langle \mathrm{Z}_j \,|\, \mathrm{Z}_{j'} \rangle = \pi \delta_{jj'}. \tag{20}$$

3) PROBLEM OF THE INTER-CONVERSION OF SINGLE/DOUBLE INDICES

For the given single index j, we have [9], [19]

$$n = n(j) = \left\lfloor \sqrt{2j-1} + \frac{1}{2} \right\rfloor - 1 \tag{21}$$

and

$$m = m(j) = \begin{cases} 2\left\lfloor \frac{2j+1-n(n+1)}{4} \right\rfloor, & n \in \mathbb{Z}^+_{\text{even}}; \\ 2\left\lfloor \frac{2j+1-n(n+1)}{4} \right\rfloor - 1, & n \in \mathbb{Z}^+_{\text{odd}}. \end{cases} \tag{22}$$

[1]It is also named polynomials of University of Arizona.

However, there is a lack of simple expression to calculate j when both n and m are given. Actually, what we can find is a description of j in a range $n(n+1)/2 + 1 \le j \le n(n+1)/2 + n + 1$ (as explained in [9], [19]). Here both the lower and the upper bound for j is not tight, so it can not be used to determine j uniquely. Consequently, it is worth exploring new results for the inter-conversion of j and $\langle n, m\rangle$.

C. ELEMENTS OF A TABLE

1) GENERAL DESCRIPTION OF A TABLE

For a general table with abstract data set

$$\mathscr{T} = \left\{ d^\alpha \in S_1 \times S_2 \times \cdots \times S_{n_{\text{cols}}} : i_{\text{begin}} \le \alpha \le i_{\text{end}} \right\}$$

where $d^\alpha = (d_1^\alpha, \cdots, d_\beta^\alpha, \cdots, d_{n_{\text{cols}}}^\alpha)$ is the α-th record as shown in TABLE 1, there are some fundamental parameters for specifications: the caption of the table, the number of columns denoted by n_{cols}, the number of rows denoted by $n_{\text{rows}} = i_{\text{end}} - i_{\text{begin}} + 1$, the list of attribute names with the size n_{cols}, the elements of attribute data which have n_{rows} records and each record is an array of abstract data type (ADT) with n_{cols} members.

2) TABLE OF MATHEMATICAL EXPRESSIONS FOR ZERNIKE CIRCULAR POLYNOMIALS

For our objective of automatically generating the long table

$$\mathscr{T}_{\text{Zernike}} = \left\{ d^j = [j, n, m, N_n^m, \mathrm{R}_n^m(\rho)\Theta_m(\theta)] : j_{\min} \le j \le j_{\max} \right\}, \tag{23}$$

for the ZCP with algorithms and computer programs, all of the data terms in the table are strings specified with LaTeX grammar. It is easy for us to set the following specifications:

- the number of columns is $n_{\text{cols}} = 5$;
- the list of attribute names are:
  - attribname-#1 — j, the symbolic expression for the single index j could be the string `"$j$"`;
  - attribname-#2 — n, the symbolic expression for the n-component of double indices $\langle n, m\rangle$ could be the string `"$n$"`;
  - attribname-#3 — m, the symbolic expression for the m-component of double indices $\langle n, m\rangle$ could be the string `"$m$"`;
  - attribname-#4 — N_n^m, the symbolic expression for the normalization coefficient could be the string `"$N^m_n$"`;
  - attribname-#5 — $\mathrm{R}_n^m(\rho)\Theta_m(\theta)$, the symbolic expression for the Zernike circular polynomial such that $\mathrm{Z}_j(\rho, \theta) = \mathrm{R}_n^m(\rho)\Theta_m(\theta)$ could be the string `"$\\Zern_j(\\rho,\\theta)= \\Radipoly{n}{m}(\\rho)\\Theta_m(\\theta)$"`;
- the caption of the table could be
  - caption — Zernike Polynomials $\mathrm{Z}_j(\rho, \theta) = N_n^m\, \mathrm{R}_n^m(\rho)\Theta_m(\theta)$.

TABLE 1. General form of caption for a table with n_{cols} attributes and $n_{\text{row}} = i_{end} - i_{begin} + 1$ records.

attribname-#1	attribname-#2	$\cdots$	attribname-#β	$\cdots$	attribname-#n_{cols}
$d_1^{i_{\text{begin}}}$	$d_2^{i_{\text{begin}}}$	$\cdots$	$d_\beta^{i_{\text{begin}}}$	$\cdots$	$d_{n_{\text{cols}}}^{i_{\text{begin}}}$
$d_1^{i_{\text{begin}}+1}$	$d_2^{i_{\text{begin}}+1}$	$\cdots$	$d_\beta^{i_{\text{begin}}+1}$	$\cdots$	$d_{n_{\text{cols}}}^{i_{\text{begin}}+1}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$	$\vdots$
d_1^{α}	d_2^{α}	$\cdots$	d_β^{α}	$\cdots$	$d_{n_{\text{cols}}}^{\alpha}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$	$\vdots$
$d_1^{i_{\text{end}}}$	$d_2^{i_{\text{end}}}$	$\cdots$	$d_\beta^{i_{\text{end}}}$	$\cdots$	$d_{n_{\text{cols}}}^{i_{\text{end}}}$

As an illustration, for $1 \le j \le 465$, our table (obtained by compiling the LaTeX source file) will have the form like TABLE 2.

III. INTER-CONVERSION OF SINGLE/DOUBLE INDICES OF ZERNIKE CIRCULAR POLYNOMIALS

The inter-conversion of the double indices $\langle n, m\rangle$ and single index j is significant for generating the table of ZCP and looking up the table for the polynomials of interests automatically. In this section, we will establish new formula and theorems to specify the implementation of the inter-conversion for Noll's single index j and Born-Wolf's double indices $\langle n, m\rangle$.

A. CONVERSION OF DOUBLE/SINGLE INDICES

The Noll's index j and the Born-Wolf's indices $\langle n, m\rangle$ can be converted to each other and the conversions can be uniquely determined.

Theorem 1: The Noll's index j can be computed with Born-Wolf's indices $\langle n, m\rangle$ by

$$j(n, m) = \frac{n(n+1)}{2} + |m| + u(m) \tag{24}$$

for $|m| \le n$ and $2 \mid (n - m)$.

Proof: Let

$$\begin{aligned} V_n &= \langle v_1, \cdots, v_r, \cdots, v_{n+1}\rangle \\ &= \begin{cases} \langle 0, -2, 2, -4, 4, \cdots, -n, n\rangle, & n \in \mathbb{Z}^+_{\text{even}} \\ \langle -1, 1, -3, 3, \cdots, -n, n\rangle, & n \in \mathbb{Z}^+_{\text{odd}} \end{cases} \end{aligned} \tag{25}$$

be the sequence of possible integers m such that

$$m = v_r = \begin{cases} r-1, & \text{for } 2 \nmid r \text{ and } 2 \mid n; \\ -r, & \text{for } 2 \mid r \text{ and } 2 \mid n; \\ -r, & \text{for } 2 \nmid r \text{ and } 2 \nmid n; \\ r-1, & \text{for } 2 \mid r \text{ and } 2 \nmid n. \end{cases} \tag{26}$$

In other words, the integer

$$r = |m| + u(m) = \begin{cases} m+1, & m \ge 0 \\ -m, & m < 0 \end{cases} \tag{27}$$

is the subscript r or the position in the sequence V_n for $m = v_r \in V_n$. It is easy to find that the cardinality of V_n is

$$|V_n| = n + 1, \quad n \in \mathbb{Z}^+ \tag{28}$$

Therefore, for $i \in \{0, 1, \cdots, n-1\}$, the total number of Zernike polynomial $\mathrm{Z}_i^m(\rho, \theta)$ is

$$\sum_{i=0}^{n-1} |V_i| = \sum_{i=0}^{n-1} (i+1) = 1 + 2 + \cdots + n = \frac{n(n+1)}{2}.$$

Thus for the given n and m, let r be position of $m \in V_{n \bmod 2}$, then the corresponding index j for the $\mathrm{Z}_j(\rho, \theta) = \mathrm{Z}_n^m(\rho, \theta)$ must be the sum of base position $n(n+1)/2$ and relative position r, i.e.,

$$j = \frac{n(n+1)}{2} + r \tag{29}$$

This implies that (24) holds since we have (27). Q.E.D.

B. CONVERSION OF SINGLE/DOUBLE INDICES

As mentioned above, there is a lack of simple formulae for converting Noll's single index j to Born-Wolf's double indices $\langle n, m\rangle$. The following theorem remedies the defects.

Theorem 2: Given the Noll index j, the Born-Wolf's indices $\langle n, m\rangle$ can be computed with the following expressions:

$$n(j) = \left\lceil \frac{-3 + \sqrt{8j+1}}{2} \right\rceil, \tag{30}$$

$$r(j) = j - \frac{n(j)[n(j)+1]}{2}, \tag{31}$$

$$m(j) = v_{r(j)} = \begin{cases} r(j) - 1, & \text{for } 2 \nmid r(j) \text{ and } 2 \mid n(j); \\ -r(j), & \text{for } 2 \mid r(j) \text{ and } 2 \mid n(j); \\ -r(j), & \text{for } 2 \nmid r(j) \text{ and } 2 \nmid n(j); \\ r(j) - 1, & \text{for } 2 \mid r(j) \text{ and } 2 \nmid n(j). \end{cases} \tag{32}$$

Proof: By (24), we have

$$\frac{n(n+1)}{2} < j \le \frac{n(n+1)}{2} + n + 1. \tag{33}$$

Consequently,

$$\frac{-3 + \sqrt{8j+1}}{2} \le n < \frac{-1 + \sqrt{8j+1}}{2} \tag{34}$$

This implies (30) by the definition of floor of real number. (31) is obvious by (29). With the help of (26) we immediately have (32). Q.E.D.

TABLE 2. Zernike circular polynomials $\mathrm{Z}_j(\rho,\theta) = N_n^m \mathrm{R}_n^m(\rho)\Theta_m(\theta)$.

j	n	m	N_n^m	$\mathrm{R}_n^m(\rho)\Theta_m(\theta)$
1	0	0	$\sqrt{1}$	1
2	1	−1	$\sqrt{4}$	$\rho\cos(\theta)$
3	1	1	$\sqrt{4}$	$\rho\sin(\theta)$
4	2	0	$\sqrt{3}$	$2\rho^2-1$
5	2	−2	$\sqrt{6}$	$\rho^2\sin(2\theta)$
6	2	2	$\sqrt{6}$	$\rho^2\cos(2\theta)$
7	3	−1	$\sqrt{8}$	$(3\rho^3-2\rho)\sin(\theta)$
8	3	1	$\sqrt{8}$	$(3\rho^3-2\rho)\cos(\theta)$
9	3	−3	$\sqrt{8}$	$\rho^3\sin(3\theta)$
10	3	3	$\sqrt{8}$	$\rho^3\cos(3\theta)$
11	4	0	$\sqrt{5}$	$6\rho^4-6\rho^2+1$
12	4	−2	$\sqrt{10}$	$(4\rho^4-3\rho^2)\cos(2\theta)$
13	4	2	$\sqrt{10}$	$(4\rho^4-3\rho^2)\sin(2\theta)$
14	4	−4	$\sqrt{10}$	$\rho^4\cos(4\theta)$
15	4	4	$\sqrt{10}$	$\rho^4\sin(4\theta)$
⋮	⋮	⋮	⋮	⋮
46	9	−1	$\sqrt{20}$	$(126\rho^9-280\rho^7+210\rho^5-60\rho^3+5\rho)\cos(\theta)$
47	9	1	$\sqrt{20}$	$(126\rho^9-280\rho^7+210\rho^5-60\rho^3+5\rho)\sin(\theta)$
⋮	⋮	⋮	⋮	⋮
464	29	−29	$\sqrt{60}$	$\rho^{29}\cos(29\theta)$
465	29	29	$\sqrt{60}$	$\rho^{29}\sin(29\theta)$

IV. GENERATING TABLE OF ZERNIKE CIRCULAR POLYNOMIALS

A. NUMERIC COMPUTATION FOR GENERATING ZERNIKE CIRCULAR POLYNOMIALS

The purpose of numeric computation for generating ZCP is to determine the key parameters involved, which includes single/double indices j, n, m, binomial coefficients $\binom{\alpha}{i}$ and the powers p_i in radial polynomial function $\mathrm{R}_n^m(\rho)$.

1) CALCUALTE THE PARAMETER k FOR THE RADIAL ZERNIKE POLYNOMIAL

The parameter k in (16) determines the number of terms of the radial Zernike polynomial $\mathrm{R}_n^m(\rho)$. The k can be directly obtained by the double indices $\langle n, m\rangle$, see **Algorithm** 1.

Algorithm 1 Calculate the parameter k for the number of terms in the radial Zernike polynomial $\mathrm{R}_n^m(\rho)$

Input: Noll's double indices $\langle n, m\rangle$
Output: the parameter k for the radial polynomial $\mathrm{R}_n^m(\rho)$
1: **function** CvtNM2K($\langle n, m\rangle$)
2: $k \leftarrow (n-|m|)/2$;
3: **return** k;
4: **end function**

2) INTERCONVERSION OF SINGLE/DOUBLE INDICES

Given the BWN pair $\langle n, m\rangle$, the Noll's single index j can be determined by (24), please see the procedure CvtNM2J in **Algorithm** 2 for the details. On the other hand, suppose the single index j is known, we can compute the double indices $\langle n, m\rangle$ according to (30), (31) and (32). The procedure CvtJ2NM in **Algorithm** 3 illustrates the steps and details completely.

Algorithm 2 Convert the BWN pair $\langle n, m\rangle$ to the Noll's single index j

Input: Double indices $\langle n, m\rangle$ of BWN where $n \in \mathbb{Z}^+$ and $m \in \mathbb{Z}$ such that $-n \le m \le n$ and $2 \mid (n-m)$.
Output: Single index $j \in \mathbb{N}$ of Noll notation.
1: **function** CvtNM2J($\langle n, m\rangle$)
2: Check the input n and m: $n \ge 0$, $-n \le m \le n$, $2 \mid (n-m)$.
3: **if** $(m \ge 0)$ **then**
4: $j \leftarrow n(n+1)/2 + m + 1$;
5: **else**
6: $j \leftarrow n(n+1)/2 - m$;
7: **end if**
8: **return** j;
9: **end function**

Algorithm 3 Convert the single index j of Noll notation to double indices $\langle n, m\rangle$ of BWN

Input: Single index j of Noll notation.
Output: BWN pair $\langle n, m\rangle$ where $n \in \mathbb{Z}^+$ and $m \in \mathbb{Z}$ such that $|m| \le n$ and $2 \mid (n-m)$.

function CvtJ2NM(j)
  Check the value of the integer j: $j \ge 1$
  $n \leftarrow \lceil(-3+\sqrt{8j+1})/2\rceil$;
  $r \leftarrow j - n(n+1)/2$;
  if $(2 \mid n)$ **then**
    if $(2 \mid r)$ **then**
      $m \leftarrow -r$; // For $2 \mid n$ and $2 \mid r$, $m = -r$.
    else
      $m \leftarrow r-1$; // For $2 \mid n$ and $2 \nmid r$, $m = r-1$.
    end if
  else
    if $(2 \mid r)$ **then**
      $m \leftarrow r-1$; // For $2 \nmid n$ and $2 \mid r$, $m = r-1$.
    else
      $m \leftarrow -r$; // For $2 \nmid n$ and $2 \nmid r$, $m = -r$.
    end if
  end if
  return $\langle n, m\rangle$;
end function

3) COMPUTATION OF BINOMIAL COEFFICIENTS

The binomial coefficients $\binom{\alpha}{i}$ is important for computing the coefficients c_s in (18). The robust and fast procedure CalcBinomCoef for computing the $\binom{\alpha}{i}$ is given in **Algorithm** 4.

Algorithm 4 Calcluating the binomial coefficient $\binom{\alpha}{i}$

Input: Real number $\alpha \in \mathbb{R}$ and non-negative integer $i \in \mathbb{Z}^+$;
Output: The real number $\binom{\alpha}{i}$.

function CalcBinomCoef(α, i)
  Check the value of integer i: $i \ge 0$
  `prod` $\leftarrow 1$;
  if $(i \ge 1)$ **then**
    for $t \in \langle 0, 1, \cdots, i-1\rangle$ **do**
      `prod` $\leftarrow$ `prod` $\cdot\, (\alpha - t)/(i-t)$;
    end for
  end if
  return `prod`;
end function

4) COMPUTATION OF COEFFICIENTS AND POWER INDICES OF RADIAL ZERNIKE POLYNOMIALS

The coefficients $\{c_s : 0 \le s \le k\}$ and power indices $\{p_s : 0 \le s \le k\}$ in the radial Zernike polynomials can be computed by the procedure CalcRadialPolyPowerCoef in **Algorithm** 5 according to (17) and (18).

Algorithm 5 Calculate the sequence of coefficients $\mathtt{c} = \langle c_0, c_1, \cdots, c_k\rangle$ and sequence of power indices $\mathtt{p} = \langle p_0, p_1, \cdots, p_k\rangle$ for the radial polynomial function $\mathrm{R}_n^m(\rho)$

Input: BWN pair $\langle n, m\rangle$ where $n \in \mathbb{Z}^+$ and $m \in \mathbb{Z}$ such that $|m| \le n$ and $n-m$ is even.
Output: The sequence of coefficients $\mathtt{c} = \langle c_0, c_1, \cdots, c_k\rangle$ and the sequence of powers $\mathtt{p} = \langle p_0, p_1, \cdots, p_k\rangle$ of radial polynomial function $\mathrm{R}_n^m(\rho)$ where $k = (n-|m|)/2$.

function CalcRadialPolyPowerCoef($\langle n, m\rangle$)
  $k \leftarrow$ CvtNM2K($\langle n, m\rangle$);
  `sign` $\leftarrow 1$;
  for $s \in \langle 0, 1, \cdots, k\rangle$ **do**
    $c_s \leftarrow$ `sign` $\cdot$ CalcBinomCoef(k, s) $\cdot$ CalcBinomCoef$(n-s, k)$;
    $p_s \leftarrow n - 2s$;
    `sign` $\leftarrow -$`sign`;
  end for
  return $\langle$`c`, `p`$\rangle$;
end function

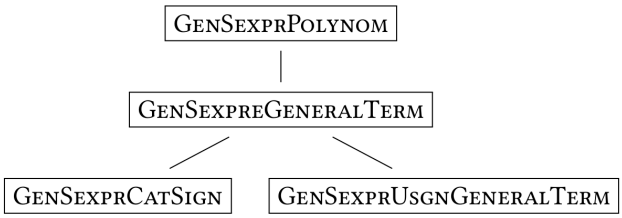

FIGURE 2. Procedures involved in generating symbolic expression for a polynomial.

B. SYMBOLIC COMPUTATION FOR GENERATING ZERNIKE CIRCULAR POLYNOMIALS

1) GENERATING SYMBOLIC EXPRESSION FOR A POLYNOMIAL

Formally, there are two steps for generating the symbolic expression for a polynomial $g(x) = \sum_{s=0}^{k} c_s x^{p_s}$:

- generate the symbolic expression for the general term $c_s x^{p_s}$
  - -- generate the unsigned general term $|c_s|\, x^{p_s}$;
  - -- determine the sign of c_s since we may get `"+"`, `"-"` or `"0"`;
- generate all of the $k+1$ terms one by one with an iterative operation via loop construction in high level computer programming language.

FIGURE 2 illustrates these steps with the help of nesting procedures for sub-tasks.

When the sequence of coefficients $\mathtt{c} = \langle c_0, c_1, \cdots, c_k\rangle$ and the sequence of power indices $\mathtt{p} = \langle p_0, p_1, \cdots, p_k\rangle$ of the polynomial $f(x) = c_0 x^{p_0} + \cdots c_s x^{p_s} + \cdots + c_k x^{p_k}$ are given, the symbolic expression of $g(x)$ can be generated with an iterative process:

- determining the generating method for the symbolic expression of general term $c_s x^{p_s}$;

- generating all of the terms iteratively for $s = 0, 1, 2, \cdots, k$.

However, there are three issues to be settled:

- specifying the formal variable x for the polynomial since it may be x, t, ρ or other possible symbol;
- specifying the operator for representing the power since x^r could be implemented by `x^r` in LaTeX and MATLAB/Octave, or by `x***r` for Python, and so on;
- specifying the special cases for the general term:
  - if $p_s = 1$, then $c_s x^1$ should be replaced by $c_s x$;
  - if $p_s = 0$, then $c_s x^0$ should be replaced by c_s;
  - if $c_s < 0$, then the string $+c_s x^{p_s}$ should be replaced by $-|c_s| x^{p_s}$;
  - if $c_s = 0$, then the term $c_s x^{p_s}$ should disappear and be ignored.

For the mathematical expression of polynomials, we should consider the coefficients and their signs carefully. The procedure GenSexprCatSign in **Algorithm** 6 is used to determine the symbol "+" or "−" for concatenation. The steps for generating a general polynomial are built on the following operations:

- determining the symbol "+" or "−" for the i-th term $c_i x^{p_i}$ according to the procedure GenSexprCatSign;
- generating symbolic expressions for the term $|c_i| x^{p_i}$ without the sign of c_i via the procedure GenSexprUsgnGeneralTerm in **Algorithm** 7;
- generating the symbolic expression for the general term $c_i x^{p_i}$ in the polynomial according to the procedure GenSexprGeneralTerm in **Algorithm** 8;
- generating the polynomial $f(x) = \sum_i c_i x^{p_i}$ with the procedure GenSexprPolynom in **Algorithm** 9.

It should be noted that if $c_i = 0$, the term $c_i x^{p_i}$ will be ignored for generating symbolic expression.

Algorithm 6 Generate the symbol "+" or "−" for concatenation

Input: Argument `fp` for text file, Coefficient c_i and position label i
Output: The sign of c_i

```
function GenSexprCatSign(fp, c_i, i)
    if (c_i < 0) then
        Fprintf(fp, "−");
    else   // for c_i ≥ 0
        if (i ≠ 0) then
            Fprintf(fp, "+");
        end if
    end if
end function
```

2) GENERATING SYMBOLIC EXPRESSIONS FOR RADIAL ZERNIKE POLYNOMIALS

Once the algorithm for generating a general polynomial is designed, we can use it to generate the Zernike radial polynomials $R_n^m(\rho)$ where the coefficients and power indices should be computed properly. The procedure GenSexprRadialPolynom in **Algorithm** 10 is designed for this purpose.

Algorithm 7 Generate symbolic expression for the term $|c_i| x^{p_i}$ without the sign of c

Input: Argument c_i for the i-th coefficient , argument p_i for the power index;
Output: The string of characters for $|c_i| x^{p_i}$;

```
function GenSexprUsgnGeneralTerm(var, op, c_i, p_i)
    ucoef ← |c_i|;
    switch p_i do
        case 1
            if (ucoef ≠ 1) then
                Fprintf(fp, "%d%s",ucoef,var);
            else
                Fprintf(fp, "%s",var);
            end if
        end case
        case 0
            Fprintf(fp, "%d",ucoef);
        end case
        default
            if (ucoef ≠ 1) then
                Fprintf(fp, "%d%s%s{%d}", ucoef,
var, op, p_i);
            else
                Fprintf(fp, "%s%s{%d}",var,op,p_i);
            end if
        end default
    end switch
end function
```

Algorithm 8 Generate the symbolic expression for the general term $c_i x^{p_i}$ in the polynomial

Input: Argument `fp` for text file, integer i, string argument `var` for x, string argument `op`, argument c_i for the i-th coefficient, integer argument for p_i for the power index of single term
Output: Symbolic expression of $c_i x^{p_i}$ stored in the text file accessed by `fp`.

```
function GenSexprGeneralTerm(fp, i, var, op, c_i, p_i)
    GenSexprCatSign(fp, c_i, i);
    GenSexprUsgnGeneralTerm(fp, var, op, c_i, p_i);
end function
```

3) GENERATING SYMBOLIC EXPRESSIONS FOR ANGULAR FUNCTION

The symbolic computation for the angular function $\Theta_m(\theta)$ is straight ford according to (14), please see the procedure GenSexprAnguFun in **Algorithm** 11.

Algorithm 9 Generate Symbolic Expression for a polynomial $f(x) = \sum_{s=0}^{k} c_s x^{p_s}$

Input: Argument `fp` for text file, symbolic argument `var` for x, symbolic argument `op` for the power operator, sequence of coefficients $\mathtt{c} = \langle c_0, c_1, \cdots, c_k \rangle$, sequence of power indices $\mathtt{p} = \langle p_0, p_1, \cdots, p_k \rangle$, integer `size` for the counting number of single terms in $f(x)$.

Output: Symbolic expression for the polynomial $f(x)$ stored in the text file accessed by `fp`.

```
function GenSexprPolynom(fp, var, op, c, p, size)
    for s ∈ ⟨0, 1, ··· , size − 1⟩ do
        if (c_s = 0) then // if c_s is equal to 0
            continue; // just ignore the term c_s x^{p_s} since it is zero, increase s by 1
        end if
        GenSexprGeneralTerm(fp, s, var, op, c_s, p_s);
        Fprintf(fp, "  "); // is is optional , print empty space for better visualization.
    end for
end function
```

Algorithm 10 Generating symbolic expression for the Zernike radial polynomial $\mathrm{R}_n^m(\rho)$

Input: Argument `fp` for text file, double indices $\langle n, m \rangle$, string `var` for ρ, string `op` for `"^"`.

Output: Print symbolic expression for the Zernike Radial polynomial $\mathrm{R}_n^m(\rho) = \sum_{s=0}^{k} c_s \rho^{p_s}$.

```
function GenSexprRadialPolynom(fp, ⟨n, m⟩, var, op)
    size ← 1 + CvtNM2K(⟨n, m⟩);   // size = 1 + k;
    ⟨c, p⟩ ← CalcRadiPolynomPowerCoef(⟨n, m⟩); // Algorithm 5
    GenSexprPolynom(fp, var, op, c, p, size);   // Algorithm 9
end function
```

Algorithm 11 Generate Symbolic Expression for Angular Function $\Theta_m(\theta)$

Input: Argument `fp` for text file, double indices $\langle n, m \rangle$

Output: Symbolic expression for angular function $\theta_m(\theta)$

```
function GenSexprAnguFun(fp, ⟨n, m⟩)
    j ← CvtNM2J(⟨n, m⟩);
    m ← |m|;
    if (j ∈ Z_even) then
        switch m do
            case 0
                if (j = 0) then
                    Fprintf(fp, "1");
                end if
            end case
            case 1
                Fprintf(fp, "\\cos(\\theta)"); // print cos(θ)
            end case
            default
                Fprintf(fp, "\\cos(%d\\theta)", m);  // print cos(mθ)
            end default
        end switch
    else
        switch m do
            case 0
                // do nothing
            end case
            case 1
                Fprintf(fp, "\\sin(\\theta)"); // print sin(θ)
            end case
            default
                Fprintf(fp, "\\sin(%d\\theta)", m);  // print sin(mθ)
            end default
        end switch
    end if
end function
```

4) GENERATING SYMBOLIC EXPRESSIONS FOR NORMALIZATION COEFFICIENT

The symbolic computation of the normalization coefficient N_n^m can be based on (13) and a simple if-else statement is enough, please see the procedure GenSexprRadiNormCoef in **Algorithm** 12.

5) GENERATING NON-STANDARD (UNNORMALIZED) ZERNIKE CIRCULAR POLYNOMIAL

For any single index $j \in \mathbb{Z}^+$ or the corresponding double indices $\langle n, m \rangle$, the ZCP $\mathrm{Z}_j(\rho, \theta) = \mathrm{Z}_n^m(\rho, \theta) = N_n^m \, \mathrm{R}_n^m(\rho)\Theta_m(\theta)$ consists of three parts: the normalization coefficient N_n^m determined by (13), the radial polynomial $\mathrm{R}_n^m(\rho)$ specified by (15), and the angular function $\Theta_m(\theta)$ given by (14). The procedure GenSexprZernikePolynomNM in **Algorithm** 13 generates the un-normalized ZCP, which separates the normalization coefficient N_n^m clearly.

C. META-PROGRAMMING AND LaTeX PROGRAMMING

For the purpose of generating mathematical formula for ZCP, we can use meta-programming and LaTeX programming. Essentially, the key idea of meta-programming is generating destination code with algorithms and computer programs implemented by source code. There are two fundamental steps to do so:

- generating LaTeX code (destination code) for creating symbolic expressions for Zernike circular polynomials with some high level programming languages such as C/C++, Octave/MATLAB, Python, Java and so on (source code);

Algorithm 12 Generating symbolic expression for the normalization coefficient N_n^m

Input: Argument `fp` for text file, double indices $\langle n, m\rangle$
Output: Print the symbolic expression normalization of coefficient $N_n^m = \sqrt{2(n+1)/(1+\delta_{0m})}$

```
function GenSexprRadiNormCoef(fp, ⟨n, m⟩)
    if m = 0 then
        Fprintf(fp, "\\sqrt{%d}", n + 1);    // print √(n+1)
    else
        Fprintf(fp, "\\sqrt{%d}", 2 ∗ (n + 1));    // print √(2(n+1))
    end if
end function
```

Algorithm 13 Generate the LaTeX code for the non-standard (un-normalized) Zernike circular function $\hat{Z}_j(\rho,\theta) = R_n^m(\rho)\Theta_m(\theta)$

Input: Argument `fp` for text file, double indices $\langle n, m\rangle$
Output: Symbolic expression for $R_n^m(\rho)\Theta_m(\theta)$ denoted by the grammar of LaTeX

```
function GenSexprZernikePolynomNM(fp, ⟨n, m⟩)
    size ← 1 + CvtNM2K(⟨n, m⟩);
    var ← "\\rho";    // for the symbol ρ in LaTeX
    op ← "^";    // for the power operator in LaTeX
    if (m = 0 or size = 1) then
        GenSexprRadiPolynom(fp, ⟨n, m⟩, var, op);    // print R_n^m(ρ)
    else
        Fprintf(fp,"(");    // print left bracket
        GenSexprRadiPolynom(fp, ⟨n, m⟩, var, op);
        Fprintf(fp,") ");    // print right braket
    end if
    GenSexprAnguFun(fp, ⟨n, m⟩);    // print Θ_m(θ)
end function
```

- compiling the LaTeX code to generate the symbolic expressions and output a file with the format *.dvi or *.pdf.

In this work, we use the C programming language to generate the LaTeX source code for representing the symbolic expressions for ZCP. Our emphasis is put on the algorithms instead of concrete C code since the algorithms can be implemented with various programming languages.

1) GENERATING LaTeX CODE FOR A LINE OF A TABLE

The procedure GenLaTeXTableLine is used to generate a line d^i of a long table

$$\mathscr{T} = \left\{ d^i \in S_1 \times S_2 \times \cdots \times S_{n_{\text{cols}}} : i_{\text{begin}} \le i \in i_{\text{end}} \right\}.$$

However, it depends on the concrete problem and we have to give details in the program. In object-oriented programming, it is a good idea to implement it with virtual function. For the purpose of generating ZCP, we can set $i \leftarrow j$ (Noll's single index j), $n_{\text{cols}} \leftarrow 5$, $i_{\text{begin}} \leftarrow j_{\min}$ and $i_{\text{end}} \leftarrow j_{\max}$. **Algorithm** 14 gives the method of generating the $Z_j(\rho,\theta)$.

Algorithm 14 Generate LaTeX code for the line of a table of Zernike Circular Polynomials

Input: Argument `fp` for text file, single index $j \in \mathbb{N}$
Output: LaTeX code for the j-th line d^j, viz.,

$$d^j = [j, n, m, N_n^m, R_n^m(\rho)\Theta_m(\theta)] \in \mathscr{T}_{\text{Zernike}}$$

```
function GenLaTeXTableLine(fp, j)
    ⟨n, m⟩ ← CvtJ2NM(j);
    Fprintf(fp, " $%d$ & $%d$ & $%d$", j, n, m);    // print the indices j, n, m
    Fprintf(fp, " &$");
    GenSexprRadiNormCoef(fp, ⟨n, m⟩);    // print the normalization coefficient N_n^m
    Fprintf(fp, "$");
    Fprintf(fp, " &$");
    GenSexprNonStandardZern(fp, ⟨n, m⟩);    // print Ẑ_j(ρ,θ) = R_n^m(ρ)Θ_m(θ)
    Fprintf(fp, "$");
end function
```

2) GENERATE LaTeX CODE FOR LONG TABLE

We can set the format of the long table based on the `\usepackage{longtable}` in the preamble of the LaTeX source file. The procedure GenLaTeXLongTable in **Algorithm** 15 is used for automatically generating a long table which may span multiple pages.

3) GENERATE MAIN BODY OF LaTeX SOURCE FILE

The main body of the LaTeX source file is a necessary part of the LaTeX source file, which has simple specification. The procedure GenLaTeXFileMainBody in **Algorithm** 16 is used for generating the main body of LaTeX source file. For controlling the the format of the pdf file to be created, it is necessary for us to set the document class of the LaTeX source file. We just set it with the mode "article" and select the 11pt fonts and A4 paper. Please see the procedure SetLaTeXDocCalss in **Algorithm** 17.

4) GENERATE LaTeX SOURCE FILE

In the LaTeX programming, it is important for us to set the preamble so as to import the macros or definitions for special symbols, mathematical environments, format specifications and so on. For our purpose, we need to import packages for mathematics, long table and paper size. Particularly, we should define new commands for printing the mathematical notations Z_n^m and R_n^m. The procedure SetLaTeXDocPreamble in **Algorithm** 18 describes the details about setting the preamble.

The key contents of LaTeX source file is the code in the LaTeX environment `\begin{document} \ldots \end{document}`, which includes the long table

Algorithm 15 Generate LaTeX code for a long table with the data sheet of size $n_{\text{rows}} \times n_{\text{cols}}$ where $n_{\text{rows}} = i_{\text{end}} - i_{\text{begin}} + 1$.

Input: Argument `fp` for LaTeX file, integer i_{begin}, integer i_{end}, variable `tab` for table information with the four members:

- n_{cols}: number of attribute
- `attribname`: list of attribute names
- `alignctrl`: alignment information
- `caption`: name of the table title

Output: LaTeX code for the long table

$$\mathscr{T} = \left\{ d^{\alpha} \in S_1 \times S_2 \times \cdots \times S_{n_{\text{cols}}} : i_{\text{begin}} \le \alpha \le i_{\text{end}} \right\}$$

function GenLaTeXLongTable(`fp`, `tab`, i_{begin}, i_{end})
  SetLtabBeginCenter(`fp`);
  SetLtabBeginLtable(`fp`, `tab`);
  SetLtabCaption(`fp`, `tab`);
  SetLtabHline(`fp`);
  SetLtabTableHead(`fp`, `tab`);
  SetLtabHline(`fp`);
  SetLtabEndFirstHead(`fp`);
  SetLtabCaptionContinue(`fp`, `tab`);
  SetLtabHline(`fp`);
  SetLtabTableHead(`fp`, `tab`);
  SetLtabHline(`fp`);
  SetLtabEndHead(`fp`);
  SetLtabHline(`fp`);
  SetLtabEndFoot(`fp`);
  SetLtabHline(`fp`);
  SetLtabEndLastFoot(`fp`);
  for $i \in \langle i_{\text{begin}}, i_{\text{begin}} + 1, \cdots, i_{\text{end}} \rangle$ **do**
    GenLaTeXTableLine(`fp`, i); // depends on concrete problem
  end for
  SetLtabHline(`fp`);
  SetLtabEndLtable(`fp`);
  SetLtabEndCenter(`fp`);
end function

Algorithm 16 Generate LaTeX code for the main body of the LaTeX source file for generating a long table

Input: Argument `fp` for LaTeX file, integer i_{begin}, integer i_{end}
Output: Main body of the LaTeX source file for generating a long table

function GenLaTeXFileMainBody(`fp`, `tab`, i_{begin}, i_{end})
  Fprintf(`fp`, `"\\begin{document}\n\n"`);
  Fprintf(`fp`, `"\\maketitle\n\n"`);
  Fprintf(`fp`, `"\n"`);
  GenLaTeXLongTable(`fp`, `tab`, i_{begin}, i_{end});
  Fprintf(`fp`, `"\n"`);
  Fprintf(`fp`, `"\\end{document}\n"`);
end function

generated by the procedure GenLaTeXLtable in **Algorithm** 15. The procedure GenKeyContentsInTeXFile in

Algorithm 17 Set the document class of the LaTeX source file

Input: Argument `fp` for LaTeX source file
Output: Text line for the document class

function SetLaTeXDocCalss(`fp`)
  Fprintf(`fp`, `"\\documentclass[11pt, a4paper,nolineno]{article}\n\n"`);
end function

Algorithm 18 Set the preamble of the LaTeX source file

Input: Argument `fp` for LaTeX source file, struct variable `layout` for the layout information which includes the following members:

- `orient`: orientation, it could be `portrait` or `landscape`
- `left`: distance for the left margin, say `"2.0cm"`
- `right`: distance for the right margin, say `"2.0cm"`
- `top`: distance for the top margin, say `"2.0cm"`
- `bottom`: distance for the bottom margin, say `"2.0 cm"`

Output: Text lines for the preamble of the LaTeX source file

function SetLaTeXDocPreamble(`fp`,`layout`)
  //Import LaTeX packages required for mathematical formula, paper size and long table
  Fprintf(`fp`, `"\\usepackage{amsmath, amsfonts,amssymb}\n"`);
  Fprintf(`fp`, `"\\usepackage[%s,left=%s, right=%s,top=%s,bottom=%s]{geometry }\n"`, `layout.orient`, `layout.left`, `layout.right`, `layout.top`, `layout.bottom`);
  Fprintf(`fp`, `"\\usepackage{longtable}\n\n"`);
  //Set the macros for generating the symbolic expresstions for Zernike functions:
  Fprintf(`fp`, `"\\DeclareMathOperator{\\Zern}{Z}\n"`);
  Fprintf(`fp`, `"\\DeclareMathOperator{\\Radi}{R}\n"`);
  Fprintf(`fp`,`"\\newcommand{\\Zernpoly}[2]{\\Zern_{#1}^{#2}}\n"`);
  Fprintf(`fp`,`"\\newcommand{\\Radipoly}[2]{\\Radi_{#1}^{#2}}\n"`);
  //Set the title and author for the pdf document to be generated
  Fprintf(`fp`, `"\\title{Table of Zernike Circular Polynoms}\n"`);
  Fprintf(`fp`, `"\\author{\ldots}\n"`);
end function

Algorithm 19 shows the mechanicsm of generating the very LaTeX code of interest.

The ultimate goal of automatically generating mathematical formula of ZCP is to create a LaTeX source file which consists of the following steps:

Algorithm 19 Generate the key contents, i.e. the LaTeX long table, for the LaTeX source file

Input: Argument `fp` for LaTeXsource file, struct variable `tab` with four members (viz. n_{cols}, `atribname`, `aligntrl` and `caption`), integer i_{begin}, and integer i_{end}

Output: The complete LaTeX code for creating long table

```
function GenKeyContentsInTeXFile(fp, tab, i_begin, i_end)
    Fprintf(fp, "\\begin{document}\n\n");
    Fprintf(fp, "\\maketitle\n\n");
    Fprintf(fp, "\n");
    GenLaTeXLtable(fp, tab, i_begin, i_end);
    Fprintf(fp, "\n");
    Fprintf(fp, "\\end{document}\n");
end function
```

- creating an empty LaTeX *.tex file with the operation mode "write";
- setting the document class;
- setting the preamble;
- generating the key contents in of the LaTeX file, and
- closing the *.tex properly.

The procedure GenLaTeXFile in **Algorithm** 20 demonstrates the above steps clearly.

Algorithm 20 Generate LaTeX source file *.tex for producing the long table of Zernike circular polynomials, viz. $\mathscr{T}_{\text{Zernike}} = \left\{d^j = [j, m, n, N_n^m, \mathrm{R}_n^m(\rho)\Theta_m(\theta)] : j_{\min} \le j \le j_{\max}\right\}$

Input: String `filename` with the suffix `.tex` for LaTeX source file, struct variable `tab` with four members (viz. n_{cols}, `atribname`, `aligntrl` and `caption`), minimum single index $j_{\min}$, maximum single index $j_{\max}$

Output: LaTeX source file with the name `filename`

```
function GenLaTeXFile(filename, tab, j_min, j_max)
    fp ← Fopen(filename, "w"); // Create the LaTeX source file with the suffix *.tex
    SetDocumentCalss(fp);
    SetDocumentPreamble(fp);
    GenKeyContentsInTeXFile(fp, tab, j_min, j_max);
    Fclose(fp);    // Close the LaTeX source file
end function
```

5) LaTeX COMPILING

Generally, the LaTeX source code for editing should be compiled with the terminal of Unix/Linux/Windows operating system or with an IDE such as TeXMaker or TeXStudio. Fortunately, there are some application program interface (API) for the operating system in high level programming language. For example, in C/C++, we can use the built-in function `system(command_expr)` to compile the source file. Here the `command_expr` stands for the commands of LaTeX compiling with the type `const char*`. Typical implementations for `command_expr` could be the string `"xelatex filename.tex"` or `"pdflatex filename.tex"`.

V. CONCLUSION

For the ZCP $\mathrm{Z}_j(\rho,\theta) = N_n^m \mathrm{R}_n^m(\rho)\Theta_m(\theta)$, our new theorems show that the conversion of Noll's single index j and Born-Wolf's double indices $\langle n, m\rangle$ can be implemented via (24), (30), (31) and (32). The inter-conversion is simple, complete and satisfactory. The symbolic expression of ZCP can be automatically generated with the GenLaTeXFile in **Algorithm** 20. For the convenience of theoretic analysis and engineering design, a system architecture of generating long table of mathematical expressions of ZCP is proposed with the help of meta-programming & LaTeX programming for computer-output typesetting. The value of the new paradigm for generating ZCP lies in three merits: editing mathematical expressions automatically instead of manually, avoiding potential errors by algorithms and programs that are verified, and saving the time overhead needed in manual operations.

As a useful reference, the mathematical expressions for $\left\{\mathrm{Z}_j(\rho,\theta) : 1 \le j \le 465\right\}$ are provided on the GitHub site, which would be sufficient for the purpose of R & D. For the users without interest of the underlying principles and implementation details, they can just download the mathematical table of ZCP released on the GitHub web site.

The implementation of our algorithms is based on the C programming language and the API of OS (the standard library function `system` in the standard C). It is easy to implement the algorithms with other programming languages which can deal with strings more conveniently such as C++, Octave, MATLAB, Python and so on. The method for automatically generating long table of mathematical expressions for ZCP can be modified slightly so as to create various long tables in optics engineering as well as in other science and technology fields, which helps to make different kinds of handbook involving massive tables.

In the sense of project-driven science-tchnology-engineering-mathematics (STEM) education, the automatic method for generating symbolic expressions of ZCP can be used to create a comprehensive project for training college students' ability of solving complex problem by combining multi-disciplinary knowledge and methods.

DATA AVAILABILITY STATEMENT

The code for automatically generating the table of Zernike circular polynomials can be downloaded from the GitHub site https://github.com/GrAbsRD/ZernikeSymbolicExpression. For the readers who has no interest in the principles and implementation of the algorithms developed in this paper, they can just download the pdf file `TableZernikePolynom-1-465.pdf` or the LaTeX source file `TableZernikePolynom-1-465.tex` to generate the long table of 465 Zernike circular polynomials $\mathrm{Z}_j(\rho,\theta)$ such that $1 = j_{\min} \le j \le j_{\max} = 465$ and $|m| \le n_{\max} = 29$. We believe that this table will satisfies the requirements of optics design.

REFERENCES

[1] M. Born and E. Wolf, *Principles of Optics*, 7th ed. London, U.K.: Cambridge Univ. Press, 1999.

[2] V. N. Mahajan, "Zernike annular polynomials for imaging systems with annular pupils," *J. Opt. Soc. Amer.*, vol. 71, no. 1, pp. 75–85, Jan. 1981.

[3] A. J. E. M. Janssen, "Zernike circle polynomials and infinite integrals involving the product of Bessel functions," 2010, *arXiv:1007.0667*.

[4] B. H. Shakibaei and R. Paramesran, "Recursive formula to compute Zernike radial polynomials," *Opt. Lett.*, vol. 38, no. 14, pp. 2487–2489, Jul. 2013.

[5] J. A. Díaz and N. Virendra Mahajan, "Orthonormal aberration polynomials for optical systems with circular and annular sector pupils," *Appl. Opt.*, vol. 52, no. 6, pp. 1136–1147, Feb. 2013.

[6] R. J. Mathar, "Zernike basis to Cartesian transformations," 2008, *arXiv:0809.2368*.

[7] J. Bühren, *Zernike Coefficients*. Berlin, Germany: Springer, 2018, pp. 1945–1946.

[8] C. G. Berger. *Zernike Aberrations*. Accessed: Aug. 18, 2022. [Online]. Available: https://opticsthewebsite.com/Zernike

[9] D. Malacara, *Optical Shop Testing*, 3rd ed. New York, NY, USA: Wiley, 2007.

[10] ZEMAX Development Corporation. (2008). *EMAX: Optical Design Program User's Guide*. [Online]. Available: www.zemax.com

[11] R. J. Noll, "Zernike polynomials and atmospheric turbulence," *J. Opt. Soc. Amer.*, vol. 66, no. 3, pp. 207–211, 1976.

[12] C.-W. Chong, P. Raveendran, and R. Mukundan, "A comparative analysis of algorithms for fast computation of Zernike moments," *Pattern Recognit.*, vol. 36, no. 3, pp. 731–742, Mar. 2003.

[13] E. C. Kintner, "On the mathematical properties of the Zernike polynomials," *Optica Acta, Int. J. Opt.*, vol. 23, no. 8, pp. 679–680, Aug. 1976.

[14] H.-Y. Zhang, Y. Zhou, and Z.-Q. Feng, "Balanced binary tree schemes for computing Zernike radial polynomials," 2022, *arXiv:2212.02495*.

[15] E. D. Knuth, *The Art of Computer Programming, volume 1: Fundamental Algorithms*, 3rd ed. New York, NY, USA: Addison-Wesley, 1997.

[16] R. Graham, E. D. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, 2nd ed. New York, NY, USA: Addison-Wesley Professional, 1994.

[17] S.-J. Zhang and J.-M. Jin, *Computation of Special Functions*. New York, NY, USA: Wiley-Interscience, 1996.

[18] J. C. Wyant. *Zernike Polynomials*. Accessed: Aug. 10, 2022. [Online]. Available: http://wyant.optics.arizona.edu/zernikes/zernikes.htm

[19] N. V. Mahajan, *Optical Imaging and Aberrations, Part III: Wavefront Analysis*. Washington, DC, USA: SPIE, 2013.

HONG-YAN ZHANG received the B.S. and M.S. degrees in applied physics and telecommunication engineering from Xidian University, China, in 2000 and 2003, respectively, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, in 2011. Currently, he is with the School of Information Science and Technology, Hainan Normal University, China. His research interests include computer vision, data analysis, STEM education, and mathematics for engineering.

YU ZHOU (Member, IEEE) received the Ph.D. degree in electronic engineering from The University of Newcastle, Newcastle upon Tyne, in 2008. Currently, he is a Research Associate in electronic engineering with Hainan Normal University, China. His research interests include algorithms analysis and design, circuit design, logic synthesis and CAD tools for asynchronous circuits and systems, and STEM education.

FU-YUN LI received the B.S. degree in computer science from Hunan University, in 1997, and the M.S. degree in telecommunication engineering from Hainan University, in 2009. She is currently an Associate Professor in computer science. Currently, she is with the School of Information Science and Technology, Hainan Normal University, China. Her research interests include applications of computer, engineering education, and speech signals processing.