Notes on Number Theory and Discrete Mathematics
Print ISSN 1310–5132, Online ISSN 2367–8275
Vol. 27, 2021, No. 4, 140–148
DOI: 10.7546/nntdm.2021.27.4.140-148

Algorithms for computing sums of powers of arithmetic progressions by using Eulerian numbers

Peter J. Shiue[1], Shen C. Huang[2] and Jorge E. Reyes[3]

[1] Department of Mathematical Sciences, University of Nevada, Las Vegas
4505 S. Maryland Pkwy. Las Vegas, NV, 89154, United States of America
e-mail: shiue@unlv.nevada.edu

[2] Department of Mathematical Sciences, University of Nevada, Las Vegas
4505 S. Maryland Pkwy. Las Vegas, NV, 89154, United States of America
e-mail: huangs5@unlv.nevada.edu

[3] Department of Mathematical Sciences, University of Nevada, Las Vegas
4505 S. Maryland Pkwy. Las Vegas, NV, 89154, United States of America
e-mail: reyesj1@unlv.nevada.edu

Received: 29 May 2021 **Revised:** 14 September 2021 **Accepted:** 22 September 2021

Abstract: The sums of powers of arithmetic progressions is of the form $a^p + (a + d)^p + (a + 2d)^p + \cdots + (a + (n - 1)d)^p$, where $n \geq 1$, p is a non-negative integer, and a and d are complex numbers with $d \neq 0$. This sum can be computed using classical Eulerian numbers [11] and general Eulerian numbers [12]. This paper gives a new method using classical Eulerian numbers to compute this sum. The existing formula that uses general Eulerian numbers are more algorithmically complex due to more numbers to compute. This paper presents and focuses on two novel algorithms involving both types of Eulerian numbers. This paper gives a comparison to Xiong *et al.*'s result with general Eulerian numbers [12]. Moreover, an analysis of theoretical time complexities is presented to show our algorithm is less complex. Various values of p are analyzed in the proposed algorithms to add significance to the results. The experimental results show the proposed algorithm remains around 70% faster as p increases.

Keywords: Analysis of algorithms, Sums of powers of arithmetic progressions, Dynamic programming, Classical Eulerian numbers, General Eulerian numbers.
2020 Mathematics Subject Classification: 05A10, 11B25, 11B65, 68N15, 68Q25.

1 Introduction

This paper is a continuation of the work on sums of powers of arithmetic progressions [9] with classical Eulerian numbers. In [9], the authors used Stirling numbers of the second kind to compute the sums of powers of arithmetic progressions. Our purpose is to discover different ways to compute the sum and compare algorithmically with other formulas of similar type. The sums of powers of arithmetic progressions are of the form

$$S_{p,a,d}(n) = a^p + (a+d)^p + \cdots + (a+(n-1)d)^p = \sum_{j=0}^{n-1}(a+jd)^p, \tag{1}$$

where $n \geq 1$, p is a non-negative integer, and a and d are complex numbers with $d \neq 0$. Several formulae and algorithms using different special functions and numbers have been developed to compute this sum [1, 6, 9, 12].

Letting $a = d = 1$ in (1) gives the following famous formula:

$$S_{p,1,1}(n) = \sum_{j=1}^{n} j^p = 1^p + 2^p + 3^p + \cdots + n^p. \tag{2}$$

For example, letting $p = 1$ gives the identity

$$S_{1,1,1}(n) = \sum_{j=1}^{n} j = \frac{n(n+1)}{2}.$$

In 2013, Xiong *et. al.* [12] presented a method to compute such a sum based on general Eulerian numbers. However, generalized Eulerian numbers may not be good because it takes longer to compute. Other authors have also published results using generalized Eulerian numbers [7, 8]. This paper aims to use a new method with classical Eulerian numbers as a comparison with Xiong *et al.*'s method with general Eulerian numbers. More specifically, we algorithmically analyze their method to obtain theoretical time complexity.

Classical Eulerian numbers are defined as the coefficients of $\binom{x+p-j}{p}$, $j = 0, 1, \ldots, n$ in the factorial expansion of x^n, namely,

$$x^p = \sum_{j=0}^{p} \left\langle {p \atop j} \right\rangle \binom{x+j-1}{p}, \quad p = 0, 1, \ldots, \tag{3}$$

which is called Worpitzky's identity [2, 3, 11]. For example, the values of $\left\langle {p \atop j} \right\rangle$ for $1 \leq p \leq 5$ are given as follows [3]:

$p \backslash j$	0	1	2	3	4
1	1				
2	1	1			
3	1	4	1		
4	1	11	11	1	
5	1	26	66	26	1

Table 1. Values of Eulerian numbers for $1 \leq p \leq 5$.

Xiong *et. al.* [12] defined the general Eulerian numbers $A_{p,j}(a,d)$ with the following recurrence relation as $A_{0,-1}=1$, $A_{p,j}=0$ $(j \geq p$ or $j \leq -2)$ and

$$A_{p,j}(a,d) = (-a+(j+2)d)A_{p-1,j}(a,d) + (a+(p-j-1)d)A_{p-1,j-1}(a,d). \quad (0 \leq j \leq p-1). \tag{4}$$

Two lemmas are presented in the next section to give explicit forms of the two special numbers discussed. In addition, the new method with classical Eulerian numbers is given. In Section 3, we analyze both Xiong *et al.*'s formula and our formula algorithmically and present their corresponding theoretical time complexities. In Section 4, experiments are conducted to compare both algorithms.

2 Main results

In this section, a new method using classical Eulerian numbers, as opposed to using general Eulerian numbers, is given. As a result, a new formula to compute sums of powers of arithmetic progression using classical Eulerian numbers is presented. A detailed analysis of this algorithm is given in the next section.

2.1 Preliminaries

The classical Eulerian number $\left\langle {p \atop j} \right\rangle$ is the number of permutations of the numbers 0 to p in which exactly j elements are greater than the previous element.

The binomial expansion of $\binom{x}{p}$ can be rewritten using the Pochhammer symbol $(x)_p$, or falling factorial, as follows:

$$\binom{x}{p} = \frac{(x)_p}{p!}, \tag{5}$$

where $(x)_p = x(x-1)\cdots(x-p+1)$. Next, two lemmas are presented to give the explicit forms of Eulerian numbers and general Eulerian numbers, which we will use in the experiments.

Lemma 2.1. *The classical Eulerian numbers* $\left\langle {p \atop j} \right\rangle$ *are given explicitly by the sum*

$$\left\langle {p \atop j} \right\rangle = \sum_{i=0}^{j} (-1)^i \binom{p+1}{i} (j-i)^p. \tag{6}$$

This lemma is proved in many manuscripts [3–5].

Lemma 2.2. *The general Eulerian numbers* $A_{p,j}(a,d)$ *are given explicity by the sum*

$$A_{p,j}(a,d) = \sum_{i=0}^{j+1} (-1)^i \binom{p+1}{i} [(j+2-i)d-a]^p. \tag{7}$$

This lemma is proved by Xiong et al. [12].

2.2 New method with classical Eulerian numbers

Given

$$x^p = \sum_{j=0}^{p} \left\langle {p \atop j} \right\rangle \binom{x+j-1}{p}, \tag{8}$$

the substitution $x = (k-1) + \frac{a}{d}$ results in

$$\left[(k-1) + \frac{a}{d}\right]^p = \sum_{j=0}^{p} \left\langle {p \atop j} \right\rangle \binom{(k-1) + \frac{a}{d} + j - 1}{p}. \tag{9}$$

Multiplying both sides by d^p and summing over k leads to

$$\sum_{k=1}^{n} [a + d(k-1)]^p = d^p \sum_{j=0}^{p} \left\langle {p \atop j} \right\rangle \sum_{k=1}^{n} \binom{(k-1) + \frac{a}{d} + j - 1}{p}. \tag{10}$$

Now, consider just $\sum_{k=1}^{n} \binom{(k-1)+\frac{a}{d}+j-1}{p}$. Expanding the right-hand side of (10) results in

$$\sum_{k=1}^{n} \binom{(k-1) + \frac{a}{d} + j - 1}{p} = \binom{\frac{a}{d} + j - 1}{p} + \binom{\frac{a}{d} + j}{p} + \cdots + \binom{n - 2 + \frac{a}{d} + j}{p}.$$

Now using Pascal's identity $\binom{n}{p} = \binom{n-1}{p-1} + \binom{n-1}{p}$, the right-hand side becomes

$$\binom{\frac{a}{d} + j}{p+1} - \binom{\frac{a}{d} + j - 1}{p+1} + \binom{1 + \frac{a}{d} + j}{p+1} - \binom{\frac{a}{d} + j}{p+1} + \cdots + \binom{n - 1 + \frac{a}{d} + j}{p+1} - \binom{n - 2 + \frac{a}{d} + j}{p+1}.$$

We see that the sum telescopes, so we have the following result

$$\sum_{k=1}^{n} \binom{(k-1) + \frac{a}{d} + j}{p} = \binom{n - 1 + \frac{a}{d} + j}{p+1} - \binom{\frac{a}{d} + j - 1}{p+1}. \tag{11}$$

Utilizing the identity (5) we arrive at the following

$$\binom{n - 1 + \frac{a}{d} + j}{p+1} - \binom{\frac{a}{d} + j - 1}{p+1} = \frac{(\frac{a}{d} + j + n - 1)_{p+1} - (\frac{a}{d} + j - 1)_{p+1}}{(p+1)!}. \tag{12}$$

Plugging (12) back into (10) gives us

$$\sum_{k=1}^{n} [a + d(k-1)]^p = \frac{d^p}{(p+1)!} \sum_{j=0}^{p} \left\langle {p \atop j} \right\rangle \left[\left(\frac{a}{d} + j + n - 1\right)_{p+1} - \left(\frac{a}{d} + j - 1\right)_{p+1} \right]. \tag{13}$$

3 Analysis of algorithms

A comparison between the general Eulerian algorithm and the proposed classical Eulerian algorithm is provided.

Since it takes $n-1$ operations to compute $n!$ and $(x)_n$, both operations have a time complexity of $O(n)$. Therefore, the binomial coefficients $\binom{n}{k}$ can be computed with $n - 1 + k - 1 + (n - k) - 1 + 2 = 2n - 1$ operations and has a time complexity of $O(n)$.

3.1 Xiong *et al.*'s formula

Xiong *et al.* gave a method to compute the sums of powers of arithmetic progressions using general Eulerian numbers [12]. In this section, we will break down their method and present an analysis on the following:

$$\sum_{k=1}^{n}(a+d(k-1))^p=\sum_{j=-1}^{p-1}A_{p,j}(a,d)\binom{n+j+1}{p+1}. \tag{14}$$

To calculate the time complexity of Xiong *et al.*'s algorithm, we only need to consider the time complexity of the binomial coefficient and of the general Eulerian numbers. It is clear that from the binomial coefficient we get the time complexity in terms of n and j.

From (14), the portion of the algorithm which computes $\binom{n+j+1}{p+1}$, takes

$$O(2(n+j+1)-1)=O(2n+2j+1)=O(n+j)$$

operations to compute.

Lemma 2.2 can be used to compute the general Eulerian numbers. Examining (7), we get a time complexity of $O(i+2p+2+5+p)=O(i+p)$. Summing from $i=0$ to $j+1$ gives an upper bound for the time complexity of $O(ji+jp)=O(j^2+jp)$. Using this calculation of $A_{p,j}(a,d)$ in (14) and summing from $k=0$ to p gives an upper bound for the time complexity of the first algorithm as

$$O(pj^2+jp^2+2pn+2pj+3p)=O(2p^3+2p^2+2pn+p)=O(p^3+pn) \tag{15}$$

in terms of the input variables p, n.

3.2 Our algorithm

An analysis on the new method presented in Subsection 2.2 is provided using Lemma 2.1 for the classical Eulerian numbers.

$$\sum_{k=1}^{n}[a+d(k-1)]^p=\frac{d^p}{(p+1)!}\sum_{j=0}^{p}\left\langle {p \atop j} \right\rangle\left[\left(\frac{a}{d}+j+n-1\right)_{p+1}-\left(\frac{a}{d}+j-1\right)_{p+1}\right]. \tag{16}$$

To calculate the time complexity of our improved formula (16), we begin with the term outside the sum, $\frac{d^p}{(p+1)!}$, which takes $2p+1$ operations to compute and, consequently, has a $O(p)$ time complexity. In addition, we see that each falling factorial terms will have a time complexity of $O(p)$. Note that the complexity no longer depends directly on the value of n.

From (6), we get a time complexity of $O(i+2p+2+3+p)=O(i+p)$ for the classical Eulerian numbers. Summing from $i=0$ to $j+1$ gives an upper bound for the time complexity of $O(ji+jp)=O(j^2+jp)$. Using this calculation of $\left\langle {p \atop j} \right\rangle$ in (16) and summing from $k=0$ to p gives an upper bound for the time complexity of the first algorithm as

$$O(pj^2+jp^2+2p^2)=O(2p^3+2p^2)=O(p^3) \tag{17}$$

in terms of the input variables p. Note that the complexity no longer depends directly on the value of n; therefore, when comparing (14) and (16), this gives evidence to suggest that the improved algorithm will run faster than the original algorithm on average. This expectation is realized in the following section, where we conduct extensive experiments.

4 Experimental results

In this section, experimental results for evaluating the sums of powers of arithmetic progressions are presented. The specifications of the testing environment are given, and the details of the experiments are described. Thereafter, the results of the original method compared to the simplified method are to be discussed.

4.1 Specifications

The computer used to run the tests consists of an Intel i7-8700K running at 3.7GHz with 6 physical cores and 16 gigabytes of RAM. The operating system used is Windows 10 Pro 20H2, 64 bit version. All of the experiments are implemented using Python 3.7 with Jupyter Notebook 6.0.3. The libraries include *Sympy* and *time*. *Sympy* is a library that does symbolic mathematics. The *time* library stores start-time and end-time, which are subtracted to obtain actual computation time. *Mathematica* is used to perform data visualizations.

4.2 Description of experiments

In the first set of experiments, the value of the power p varies from $p = 10$ to $p = 100$ with $\Delta p = 10$ as the increments. This aims to give the sums of powers of arithmetic progressions in the general form. The results are simplified when comparing the results.

In the second set of experiments, the values of $a, d,$ and n are fixed to randomly generated 20-digit numbers. The value of the power p varies from $p = 200$ to $p = 2000$ with $\Delta p = 200$ as the increments. This aims to better compare the two algorithms as lower values of p may not give significant results. In both sets of experiments, the run-times are collected and averaged over 100 runs.

4.3 Results and discussion

The results are visualized using *Mathematica*. The graphs are used to compare our algorithm (16) against Xiong *et al.*'s algorithm (14). Two tables are given to see the numerical results.

The main contribution to the drastic difference in times shown in Figure (a) is that Xiong *et al.*'s algorithm gives the result in polynomials in n, whereas our algorithm gives the result in an unsimplified form. Even though the theoretical results can be simplified to the identity, a computer check was also performed to verify that the answers are the same. This gives a more comparable result since the first set of experiments does not give the results in the same way.

Values of p	10	20	30	40	50
Our algorithm	0.4747	1.4272	3.1077	5.9152	10.2915
Xiong *et al.*'s algorithm	0.5635	4.5019	27.0317	101.6553	252.6017
Values of p	60	70	80	90	100
Our algorithm	17.8694	25.0377	34.7735	46.7552	61.1408
Xiong *et al.*'s's algorithm	500.5395	807.2444	1309.9709	2003.4787	2703.5214

Experiment One: The time is measured in seconds. Algorithms generate the general form of the sums of powers of arithmetic progressions. This table shows that our algorithm is far more efficient.

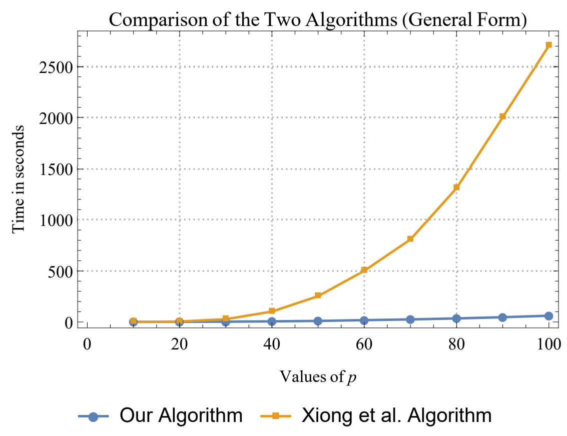

Figure (a). A comparison of the two algorithms that calculates the general form of the sums of powers of arithmetic progressions by varying p only shows that our algorithm is drastically more efficient.

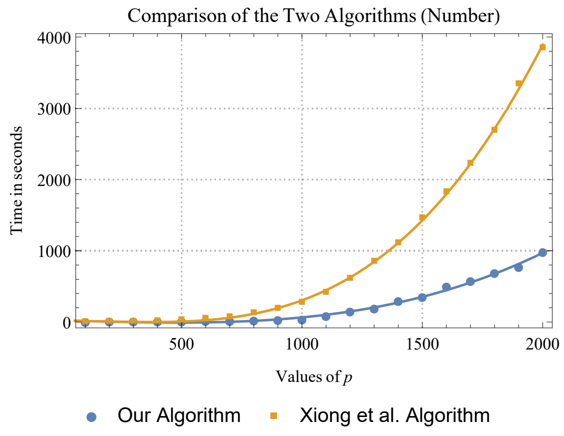

Figure (b). A comparison of the two algorithms that calculate the result of the sums of powers of arithmetic progressions by fixing values of a, d, and n and varying p. The plot includes the line of best fit for two data sets.

Values of p	100	200	300	400	500
Our algorithm	0.4497	1.0876	2.1052	3.9748	5.9838
Xiong *et al.*'s's algorithm	0.359	1.3106	4.1687	10.9749	22.7876
Values of p	600	700	800	900	1000
Our algorithm	9.2827	13.244	17.6911	23.2921	34.1791
Xiong *et al.*'s's algorithm	44.2563	70.7529	124.8384	188.3088	273.7294
Values of p	1100	1200	1300	1400	1500
Our algorithm	82.4342	145.502	188.3088	290.617	346.7947
Xiong *et al.*'s's algorithm	412.5691	604.4786	846.1521	1105.5296	1457.421
Values of p	1600	1700	1800	1900	2000
Our algorithm	492.055	570.5803	681.9292	770.6358	979.0942
Xiong *et al.*'s's algorithm	1822.8166	2221.2837	2681.4324	3336.1961	3840.6724

Experiment Two: The time is measured in seconds. The constants a, d, and n are fixed to randomly generated 20-digit numbers, where the value of p varies. This table shows that our algorithm is faster in generating the values of sums of powers of arithmetic progressions.

In the second set of experiments, the algorithms compute the value of the sums of powers of arithmetic progressions with $a, d,$ and n fixed and p varied. The results show that our algorithm performs faster than Xiong *et al.*'s method since our algorithm does not depend on the value of n. As seen in Figure (b), both algorithms have a cubic polynomial time, with the general Eulerian number method using more time.

5 Conclusion

In this paper, two novel algorithms using the Eulerian numbers are given to compute the general form of sums of powers of arithmetic progressions $S_{p,a,d}(n)$. In addition, formula (13) is used to compare with Xiong *et al.*'s method, which uses the general Eulerian numbers.

In Section 3, extensive analyses of both algorithms are presented. These analyses suggest that our algorithm using classical Eulerian numbers will on average run faster than Xiong *et al.*'s algorithm using general Eulerian algorithm. The experimental results also support the analyses.

This paper focused on using a new method with classical Eulerian numbers. A future project may involve algorithms using dynamic programming for solving the sums of powers of arithmetic progressions using other special combinatorial numbers such as Bernoulli polynomials and numbers [1, 10].

Acknowledgements

The authors would like to thank two referees and Professor Anthony Shannon for their comments, corrections, and suggestions which led to considerable improvements to this paper.

References

[1] Bounebirat, F., Laissaoui, D., & Rahmani, M. (2018). Several explicit formulae of sums and hyper-sums of powers of integers. *Online Journal of Analytic Combinatorics*, 13(4), Article 4 (9 pages).

[2] Carlitz, L. (1959). Eulerian numbers and polynomials. *Mathematics Magazine*, 32(5), 247–260.

[3] Charalambides, C. A. (2002). *Enumerative Combinatorics*. Chapman & Hall / CRC Press.

[4] Comtet, L. (1974). *Advanced Combinatorics: The Art of Finite and Infinite Expansions*. Springer Science & Business Media.

[5] Hsu, L. C., & Shiue, P. J. (1999). On certain summation problems and generalizations of Eulerian polynomials and numbers. *Discrete Mathematics*, 204(1–3), 237–247.

[6] Laissaoui, D., & Rahmani, M. (2017). An explicit formula for sums of powers of integers in terms of Stirling numbers. *Journal of Integer Sequences*, 20(4), Article 17.4.8 (6 pages).

[7] Lehmer, D. (1982). Generalized Eulerian numbers. *Journal of Combinatorial Theory, Series A*, 32(2), 195–215.

[8] Pita-Ruiz, C. (2018). On a generalization of Eulerian numbers. *Notes on Number Theory and Discrete Mathematics*, 24(1), 16–42.

[9] Shiue, P. J., Huang, S. C., & Jameson, E. (2020). On algorithms for computing the sums of powers of arithmetic progressions. *Notes on Number Theory and Discrete Mathematics*, 26(4), 113–121.

[10] Vassilev, P., & Vassilev-Missana, M. (2005). On the sum of equal powers of the first n terms of an arbitrary arithmetic progression. *Notes on Number Theory and Discrete Mathematics*, 11(3), 15–21.

[11] Worpitzky, J. (1883). Studien über die Bernoullischen und Eulerschen Zahlen. *Journal für die reine und angewandte Mathematik*, 94, 203–232.

[12] Xiong, T., Tsao, H.-P., & Hall, J. I. (2013). General Eulerian numbers and Eulerian polynomials. *Journal of Mathematics*, 2013, 1–9.