A New Algorithm for Multipoint Evaluation of Univariate Polynomials

WA Gunarathna[1#] and HM Nasir[2]

[1]*Department of Mathematics, Faculty of Engineering, General Sir John Kotelawala Defence University, Ratmalana, Sri Lanka*

[2]*Department of Mathematics and Statistics, College of Science, Sultan Qaboos University, Oman*

[#]gunarathnawa@yahoo.com

***Abstract*— Let $p_n(x_i)=\sum_{l=0}^{n-1}a_l x_i^{\,l}$ *be a univariate polynon mial of degree* $n-1$ *defined on* $\Re[x]$*, where* $\Re[x]$ *denotes the ring of polynomials in* x *over* $\Re$*, the field of real numbers and let* $S=\{x_0,\ldots,x_{n-1}\}$ *be any set of* m *distinct elements in* $\Re$*. The role of Multipoint Evaluation Problem (MEP) is to compute the finite sum* $p_n(x_i)=\sum_{l=0}^{n-1}a_l x_i^{\,l}$ *for all* $i=0,1,\ldots,m$*. These types of evaluations are used most abundantly in many areas such as Engineering, Physics, Medicine, and Weather forecasting. The MEP of interest in this paper is restricted to the case where* $m=n$*. The paper proposes a new algorithm with asymptotic time complexity of* $O(n^2)$ *for the MEP. For the sake of simplicity, we assume that* $n=2^k$*, where* $k=0,1,2,\ldots$*. We explore performance of the algorithm by means of numerical experiments. The numerical results confirm that the algorithm is faster than Estrin's method and that it is as accurate as Estrin's method.*

Keywords*— *Multipoint evaluation problem, Horner's rule, Estrin's method.

I. INTRODUCTION

The evaluation of a polynomial at several points, referred in the literature as the Multipoint Polynomial Evaluation problem (MPE), is one of the fundamental tasks in Computational Mathematics, which holds indispensable applications arising from many areas such as Engineering, Physics, Weather forecasting, Medicine, Signal processing, and so on. We denote a generic univariate polynomial of degree $n-1$ in $\Re[x]$ by $p_n=\sum_{i=0}^{n-1}a_i x^i$, where $\Re[x]$ denotes the ring of polynomial in x over $\Re$, the field of real numbers.

In the naive approach for solving the MPE, each term in p_n is computed independently of the other remaining terms in p_n; thereby it requires $n(n-1)/2$ multiplications and $(n-1)$ additions, in order to evaluate p_n at a single evaluation point. Therefore, the total number of operations required to evaluate p_n at n evaluation points is $(n^3+n^2-2n)/2$, which is $O(n^3)$. In the MEP literature, some attempts were also made to bring down this prohibitive cost. Horner's rule, which is a sequential algorithm, is known to be a more stable algorithm for solving any MPE, which would require $O(n^2)$ arithmetic operations (Dorn, W. S. (1961, January). Estrin's method establishes a parallel algorithm for solving the MPE which costs $O(n^2)$ arithmetic operations (Estrin, G. (1960, May)).

The standard Discrete Fourier Transform (DFT) which is a special case of the MEP where the polynomial p_n is evaluated at n uniformly located points, $e^{-i2\pi j/n}\,(j=0,1,\ldots,n-1)$, on the unit circle in the complex plane. The well-known Fast Fourier Transform (FFT) solves this kind of the MPE much more efficiently and accurately in a cost of $O(n\log_2 n)$ (Cooley JW. & Tukey JW (1965)). The materials found from the work of A Borodin and M Munro in 1975 (Borodin, A. M., & Munro, I. H. (1975) pointed out an exact algorithm of $O(n\log_2{}^2 n)$ for solving the MPE of size n. These authors used modular technique to develop this algorithm. Further, J.R.Driscoll, D.M.Healy Jr., and D. Rockmore (Driscoll JR, Healy Jr, DM. & Rockmore DN (1997), Moore SS. Healy Jr DM. & Rockmore DN. (1993)) formulated an $O(n\log_2{}^2 n)$ algorithm. The basic idea of this algorithm is that the Vandermonde matrix associated with the Vandermonde matrix- vector product equivalent to MPE is factored into a product of sparse matrices including Toeplitz blocks, so that the fast Toeplitz matrix vector product can iteratively be applied to compute the Vandermonde matrix vector product more efficiently. However, the naive implementation of this algorithm in computer arithmetic shows serious numerical stability problems.

In this paper, we propose a new sequential algorithm with time complexity of $O(n^2)$, in order to solve the

MPE. For the sake of simplicity, it is assumed that $n = 2^k$ for $k = 0,1,2,\ldots$.

The organization of this paper is set to have the following structure: In Section II, we present preliminaries. Section III describes the formulation of the new algorithm, Section IV is devoted to present the implementation of the algorithms, and Section V exhibits the numerical results. Section VI concentrates on the discussion of the numerical results. Finally, we give conclusions in Section VII.

II. PRELIMINARIES

This section presents the multipoint evolution problem and some known algorithms, namely, Horner's method and Estrin's method for solving the MPE.

A. Multipoint Evolution problem

Let $p_n(x) = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1}$ be a generic poly nominal of degree $n-1$ defined on $\Re$ and let $S = \{x_0, x_1, \ldots, x_m\}$ be any finite subset of $\Re$ including $m(\geq n-1)$ distinct evaluation points. Then the problem of evaluation of the polynomial $p_n(x)$ at every point in S is called the Multipoint Evaluation Problem. To put it another way, it is required to compute the finite sum:

$$p_n(x_i) = \sum_{l=0}^{n-1} a_l x_i^{\,l} \text{ for all } i = 0,1,\ldots,m.$$

This paper is restricted to address the case where $m = n$.

B. Horner's rule(Hor.)

Let $q_{n-k}(x)$ be a polynomial of degree $n-1-k$ defined by

$$q_{n-k}(x) = a_k + a_{k+1}x + a_{k+2}x^2 + \ldots + a_{n-1}x^{n-1-k},$$

so that

$$q_{n-k}(x) = a_k + xq_{n-k-1}(x) \text{ for all } k = 0,1,\ldots,n-1.$$

It should be noticed that

$$q_1(x) = a_{n-1} \text{ and that } p_n(x) = q_n(x).$$

Now, for a given fixed value of x, $q_n(x)$ can recursively be computed in $n-1$ additions and $n-1$ multiplications and then we get $p_n(x) = q_n(x)$.

This method is called Horner's rule and it further can be expressed by the following nested multiplication form:

$$p_n(x) = a_0 + x(a_1 + \ldots + x(a_{n-3} + x(a_{n-2} + a_{n-1}x))\ldots).$$

For example, $p_8(x)$ has the following form:

$$a_0 + x(a_1 + x(a_2 + x(a_3 + x(a_4 + x(a_5 + x(a_6 + a_7x)))))).$$

The total number of operations required to evaluate $p_n(x)$ at n evaluation points is $2n^2 - 2n$ which implies that the computational complexity of Horner's rule for solving MPE is $O(n^2)$.

C. Estrin's method(Est.)

Let $b_i = a_{2i} + xa_{2i+1}$, where $i = 0,1,\ldots,n/2-1$.

The idea of Estrin's method is that polynomial $p_n(x)$ is expressed in terms of b_i's, so that

$$p_n(x) = b_0 + b_1x^2 + b_2x^4 + \ldots + b_{(n-2)/2}x^{n-2}.$$

For example,

$$p_8(x) = b_0 + b_1x^2 + b_2x^4 + b_3x^6,$$

where $b_i = a_{2i} + a_{2i+1}x$ for $i = 0,1,2,3$.

The number of additions and the number of multiplications should be devoted to compute b_i's for every $i = 0,1,\ldots,n/2-1$ are $n/2$ and $n/2$, respectively.

It can be easily seen that computing power term x^k in the way that x is multiplied by itself $k-1$ times is rather time consuming. For example, $x^8 = x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x$ takes 7 multiplications. However, this can be reduced to 3 multiplications by pre-computing x^2, x^4 and then writing x^8 as:

$$x^8 = (x^4)(x^4)$$

(Nawaz Khan, S. (2010)).

We may exploit this idea to compute power terms in Esttrin's method, in order to save number of operations.

In Estrin's expansion, it is required to generate all the power terms in the form of

$$(x^2)^m \text{ for all } m = 1,\ldots,(n/2-1).$$

We use the relation that

$$(x^2)^m = (x^2)^{m-1}(x^2) \text{ for all } m = 1,\ldots,(n/2-1).$$

This implies that:

$$\begin{aligned} x^2 &= x \cdot x \\ (x^2)^2 &= (x^2)\cdot(x^2) \\ (x^2)^3 &= (x^2)^2(x^2) \\ (x^2)^4 &= (x^2)^3(x^2) \\ &\vdots \quad \vdots \quad \vdots \\ (x^2)^{(n/2-1)} &= (x^2)^{(n/2-2)}(x^2). \end{aligned}$$

In this way, the total number of multiplications required to generate all the power terms is $(n/2-1)$.

Therefore, the total number of operations required to evaluate $p_n(x)$ at n evaluation points is

$$[n/2+n/2+(n/2-1)+(n/2-1)]n = 2n^2-2n.$$

This shows that the computational complexity of Estrin's method for solving MPE is $O(n^2)$.

III. FORMULATION OF NEW ALGORITHM

Let $x=\alpha \in S$.

Define $a_i^{(j)} = a_i^{(j-1)} + \alpha^{n_j} a_{i+n_j}^{(j-1)}$

for all $j=1,2,\ldots k$ and $i=0,1,\ldots,n_j-1$,

where $a_i^{(0)} = a_i$ for all $i=0,1,\ldots,n-1$ and $n_j = n/2^j$ for all $j=1,2,\ldots,k$.

Now,

$$\begin{aligned} p_0(\alpha) = p(\alpha) &= \sum_{i=0}^{n-1} a_i \alpha^i \\ &= \sum_{i=0}^{n/2-1} a_i \alpha^i + \alpha^{n/2} \sum_{i=0}^{n/2-1} a_{i+n/2} \alpha^i \\ &= \sum_{i=0}^{n_1-1} (a_i + \alpha^{n_1} a_{i+n_1}) \alpha^i \\ &= \sum_{i=0}^{n_1-1} a_i^{(1)} \alpha^i \end{aligned}$$

Let $p_1(\alpha) = \sum_{i=0}^{n_1-1} a_i^{(1)} \alpha^i$.

Then $p_1(\alpha)$ can again be written as

$$\begin{aligned} p_1(\alpha) &= \sum_{i=0}^{n/4-1} a_i^{(1)} \alpha^i + \alpha^{n/4} \sum_{i=0}^{n/4-1} a_{i+n/4}^{(1)} \alpha^i \\ &= \sum_{i=0}^{n_2-1} (a_i^{(1)} + \alpha^{n_2} a_{i+n_2}^{(1)}) \alpha^i \\ &= \sum_{i=0}^{n_2-1} a_i^{(2)} \alpha^i \end{aligned}$$

Letting $p_2(\alpha) = \sum_{i=0}^{n_2-1} a_i^{(2)} \alpha^i$, we can write $p_2(\alpha)$, in terms of $p_3(\alpha)$ as

$$p_2(\alpha) = \sum_{i=0}^{n_3} a_i^{(3)} \alpha^i = p_3(\alpha).$$

Then at the j th step, we get

$$p_j(\alpha) = \sum_{i=0}^{n_j} a_i^{(j+1)} \alpha^i ,$$

where

$$a_i^{(j+1)} = a_i^{(j)} + \alpha^{n_{j+1}} a_{i+n_{j+1}}^{(j)}$$

for all $j=0,1,\ldots,k-1$.

Noticing the fact that $p(\alpha) = p_j(\alpha)$ for all $j=0,1,\ldots,k$, we get,

$$p(\alpha) = p_k(\alpha) = a_0^{(k)} = a_0^{(k-1)} + \alpha a_1^{(k-1)}.$$

To illustrate this, consider case where $n=8$.

That is

$$p_8(x) = \sum_{i=0}^{7} a_i x^i.$$

Suppose that it is required to evaluate $p_8(x)$ at $x=\alpha$.

Step 1

Compute the vector $(a_0^{(1)}, a_1^{(1)}, a_2^{(1)}, a_3^{(1)})$,

where $a_i^{(1)} = a_i + \alpha^4 a_{i+4}$ for $i=0,1,2,3$.

Step 2

Compute the vector $(a_0^{(2)}, a_1^{(2)})$,

where $a_i^{(2)} = a_i^{(1)} + \alpha^2 a_{i+2}^{(2)}$ for $i=0,1$.

Step 3

Compute the vector $(a_0^{(3)})$, where

$$a_0^{(3)} = a_0^{(2)} + \alpha a_1^{(2)}.$$

Therefore,

$$p(\alpha) = a_0^{(3)}.$$

1) Operation count

The power terms that must be computed in the new algorithm are $\alpha^2, \alpha^4, \ldots, \alpha^{n/2-1}$ only.

Using the efficient method discussed in Section II for generating power terms, one can easily show that the number of operations needed to compute all the power terms required in the new algorithm is $(k-1)$.

In the first step, it is required to compute:

$$a_i^{(1)} = a_i + \alpha^{n/2} a_{i+n/2} \text{ for all } i=0,1,\ldots,n/2-1.$$

Therefore, the number of operations taken is n.. Making the same argument, the number of operations required for Step II is $n/2$ and thus the number of operations that we may devote, in order to compute all the k steps is

$$n+\frac{n}{2}+\frac{n}{2^2}+\ldots+\frac{n}{2^{k-1}}=2(n-1).$$

Therefore, the total number of operations required to evaluate $p_n(x)$ at a single point is

$$2(n-1)+k-1.$$

Therefore, the total number of operations required to evaluate $p_n(x)$ at n such points is

$$\begin{aligned} n[2(n-1)+k-1] &= 2n^2-3n+nk \\ &= n(2n+\log_2 n-3). \end{aligned}$$

This evidently confirms that the computational cost of the new algorithm is $O(n^2)$.

IV. IMPLEMENTATION

This section focuses on implementing the previously described algorithm in Section III. First, the Horner' rule is described. Then, Estrin's method is discussed. Finally, the implementation of the new algorithm is presented.

Algorithm 1: Horner's rule

INPUT: $S=(x_0,x_1,\ldots,x_{n-1})$, $p=(a_0,a_1,\ldots,a_{n-1})$

OUPUT: $V=(p(x_0),p(x_1),\ldots,p(x_{n-1}))$

STAGES:

for $i=1$ to do n

$q_0=0$

for $j=1$ to do n

$q_j = p(n+1-j)+S(i)q_{j-1}$

end

$V(i)=q_n$

end

Algorithm 2: Estrin's method

INPUT: $S=(x_0,x_1,\ldots,x_{n-1})$, $p=(a_0,a_1,\ldots,a_{n-1})$

OUTPUT: $V=(p(x_0),p(x_1),\ldots,p(x_{n-1}))$

STAGES:

for $i=1$ to do length (S)

Polyval=0

for $j=1$ to length (p)/2

$b_j = p(2j-1)+S(i)p(2j)$

Polyval=Polyval+ $b_j[(S(i))^2]^{j-1}$

end

$V(i)=$ Polyval

end

Algorithm 3: New algorithm

INPUT: $S=(x_0,x_1,\ldots,x_{n-1})$ $p=(a_0,a_1,\ldots,a_{n-1})$,

$K=\log_2 n$

OUTPUT: $V=(p(x_0),p(x_1),\ldots,p(x_{n-1}))$

STAGES:

for $i=0$ to do length (S) -1

$\alpha \leftarrow x_i$

for $j=0$ to do length (p)-1

$a_j^{(0)}=a_j$

$n_j \leftarrow \frac{n}{2^j}$

for $k=1$ to do K

for $m=0$ to do $n/2^{k+1}-1$

$a_k^{(j+1)}=a_k^{(j)}+\alpha^{n_{j+1}}a_{k+n_{j+1}}^{(j)}$

end

end

end

$p(\alpha)=a_0^{(K)}$

end

v. NUMERICAL RESULTS

This section shows numerical results of two numerical experiments carried out to test the performance of the algorithm. All the computations were performed on a personal computer with Intel(R) Pentium 2.1 GHz processor, with 2.00 GB RAM, 64 bit windows 7 operating system using MATLAB version 12 codes.

A. Numerical Experiments

1) In this experiment, the sets of evaluations points, $S=\{x_0, x_1, \ldots, x_{n-1}\}$ are randomly chosen from the interval $[0,1]$ and so are the coefficients, a_i's of polynomial p_n. The polynomial p_n is computed on S for various values of n.

2) The experiment focuses on the evaluation of the polynomial p_n given by:

$$p_n(x)=1+2x+3x^2+\ldots+nx^{n-1}.$$

The sets of evaluations points S are randomly chosen from the interval [0, 1].

In both experiments, the accuracy of the algorithm is described by means of relative errors (RE). The relative error of the output vector $V=(p(x_0), p(x_1), \ldots, p(x_{n-1}))$ is computed with respect to the maximum relative error (MRE), the infinity norm and the 2-norm defined through the following the formulae, respectively:

$$\mathrm{MRE}=\max_{i=0,1,\ldots,n-1}\frac{\left|v_i-v_i^*\right|}{\left|v_i^*\right|} \quad (1)$$

$$RE_\infty=\frac{\max_{0\le i\le n-1}\left|v_i-v_i^*\right|}{\max_{0\le i\le n-1}\left|v_i^*\right|} \quad (2)$$

$$RE_2=\sqrt{\frac{\sum_{i=0}^{n-1}\left|v_i-v_i^*\right|^2}{\sum_{i=0}^{n-1}\left|v_i^*\right|^2}} \quad (3)$$

In the above error formulae, v and v^* stand for the result computed by either the new algorithm or Estrin's method and the corresponding result computed by the Horner's rule, respectively. MATLAB rand () function has been used to generate all sets of random evaluation points and random polynomials. The efficiency of the new algorithm and Estrin's method is demonstrated in terms of CPU times. MATLAB CPU time function has been operated to compute the CPU times.

In both experiments, the accuracy and efficiency of the new algorithm are compared with those of Estri's method. We have used Horner's rule as the reference point.

B. Numerical Results

Table 1 and Table 3 exhibit relative errors, namely, MRE RE_∞ and RE_2, computed for various values of n in Experiment 1 and Experiment 2, respectively, while Table 2 and Table 4 demonstrate the corresponding CPU times (in seconds) elapsed to evaluate polynomial p_n.

1) Numerical results of Experiment 1

Table 1. Relative errors

	New algo. $\times 10^{-15}$			Est. $\times 10^{-15}$		
n	MRE	RE_∞	RE_2	MRE	RE_∞	RE_2
64	1.08	0.248	0.208	0.671	0.186	0.167
128	21.5	0.959	0.479	21.2	1.44	0.883
256	72.9	1.49	0.879	107	4.49	2.08
512	34.6	1.42	0.965	28.45	5.34	2.97
1024	836	2.61	1.77	179	12.1	6.54
2048	743	2.73	1.17	176	29.9	14.5
4096	8.51	4.61	1.74	58.5	57.9	28.9

Table 2. CPU times

	CPU times in seconds		
n	Hor.	New algo.	Est.
64	0.3120	0.3276	0.3432
128	0.3120	0.3588	0.3900
256	0.3276	0.3744	0.6240
512	0.3900	0.6864	1.747
1024	0.6240	1.700	13.92
2048	1.529	6.365	109.3
4096	5.164	23.63	1011.6

2) Numerical results of Experiment 2

Table 3. Relative errors

	New algo. $\times 10^{-15}$			Estrin's meth $\times 10^{-15}$		
n	MRE	RE_∞	RE_2	MRE	RE_∞	RE_2
64	239.4	0.328	0.216	7.85	0.765	0.457
128	1583	0.441	0.319	43.3	2.09	1.09
256	1670	0.995	0.488	357	3.43	2.16
512	9525	0.665	0.498	1299	7.76	4.15
1024	8501	1.77	0.923	2147	18.3	9.48
2048	49434	2.22	1.40	14319	36.3	20.3
4096	323251	3.55	1.94	80634	75.0	39.7

Table 4. CPU times

	CPU times in seconds		
n	Hor.	New algo.	Est.
64	0.000	0.016	0.016
128	0.016	0.016	0.062
256	0.016	0.094	0.296
512	0.078	0.312	1.092
1024	0.234	1.217	4.98
2048	0.998	4.29	42.9
4096	3.136	11.44	386.4

V. DISCUSSION

Based on the numerical results, it can be seen from Table 1 and Table 3 that the MRE's of the new algorithm are the same as those of Estrins' method up to at least 9 significant figures , for example , in Experiment 2, MRE of the new algorithm $= 3.23 \times 10^{-10}$, and MRE of Estrin's method $= 8.06 \times 10^{-11}$ when $n = 4096$, whereas the corresponding MRE's of the new algorithm and Estrin's methode in Experiment 1 are 8.5×10^{-15} and 5.84×10^{-14}, respectively when $n = 4096$, while in both experiments, RE_∞ and RE_2 of the new algorithm are almost equal to those for Estrin's method up to 14 significant figures for all the values of n stipulated. In fact, RE_∞ and RE_2 of the new algorithm are slightly smaller than those of Estrin's method. The preceding results establish the fact that the accuracy of the new algorithm is almost the same as that of Estrin's method (in fact the accuracy of the new algorithm is slightly better than the accuracy of Estrin's method).

Table 2 and Table 4 demonstrate that in both experiments, the new algorithm is faster than Estrin's method when $n > 128$ (in Experiment 1, the new algorithm is about 43 times faster at $n = 4096$, while Horner's rule is about 196 times faster at $n = 4096$ and in Experiment 2, the new algorithm is about 34 times faster and Horner's rule is about 123 times faster at $n = 4096$). Indeed, the number of operations required to compute all power terms ($(n\log_2 n - n)$)in the new algorithm is smaller than that of Estrin's method $(n^2/2 - n)$. Further it also should be taken into account that the number of arithmetic operations taken by the new algorithm is higher than those of Estrin's method, whereas the new algorithm is faster than Estrin's method in practical computation. Perhaps, this would be due to the fact that Estrin's method is a parallel algorithm, while the Horner's rule and the new algorithm are both a sequential or serial algorithms. Although the focus of this paper was restricted to the polynomials of degree $n-1$, where $n = 2^k$ for $k = 1, 2, \ldots$, the new algorithm can be used to address the MEP with general polynomials.

VI. CONCLUSION

In this paper, we have developed a new algorithm, in order to evaluate a univariate polynomial of degree $n-1$ at n evaluation points or to solve a multipoint evaluation problem (MPE) consisting of a univariate polynomial of degree $n-1$ together with n evaluation points. The asymptotic time complexity of the algorithm is $O(n^2)$, which is the same as the that of Horner's method or Estrin's method. It is concluded that the new algorithm is more suitable for sequential implementation than Estrin's method. The future interest of this paper is subject to bring down the complexity of the new algorithm to a quasilinear time complexity, for example $O(n\log^2 n)$.

REFERENCES

Borodin, A. M., & Munro, I. H. (1975). Computational complexity of algebraic and numeric problems.

Cooley JW. & Tukey JW (1965). "An algorithm for the machine calculation of complex Fourier series," *Mathematics of computation*, vol. 19, 297- 301pp.

Driscoll JR , Healy Jr, DM. & Rockmore DN (1997). "Fast discrete polynomial transform with applications to data analysis for distance transitive graphs," *SIAM Journal on Computing*, vol.26, 1066-1099 pp.

Moore SS. Healy Jr DM. & Rockmore DN.(1993).“Symmetry stabilization for fast discrete monomial transforms and polynomial evaluation,” *Linear algebra and its applications, vol.* 192, 249-299 pp.

Dorn, W. S. (1961, January). A generalization of Horner's rule for polynomial evaluation. In *Proceedings of the 1961 16th ACM national meeting* (pp. 61-501). ACM

Estrin, G. (1960, May). Organization of computer systems: the fixed plus variable structure computer. In *Papers presented at the May 3-5, 1960, western joint IRE-AIEE-ACM computer conference* (pp. 33-40). ACM.

BIOGRAPHY OF AUTHORS

WA Gunarathna is a Mathematics lecturer at the department of Mathematics of Faculty of Engineering, General Sir John Kotelawala Defence University, Ratmalana, Sri Lanka. His research interests include design of super fast algorithms for polynomial transforms, numerical solutions of partial differential equations, Computational Number theory and Computational Algebra.

Nasir HM is a Visiting Consultant at Department of Mathematics and Statistics of College of Science of Sultan Qaboos University, Oman . He received Master of Engineering (M.Eng) and Ph.D in computational Mathematics, both from the University of Electro Communications, Tokyo, JAPAN in 1999 and 2003, respectively. His research interests include Numerical solutions of partial differential equations, and multi complex analysis.