Complejidad Computacional
Análisis Comparativo de la Evaluación de Polinomios empleando el Método de Horner vs. el Tradicional Método de Potenciación

Lic. Adriana Labra Barrios.

M. en C. Eduardo René Rodríguez Ávila.

Sección de Estudios de Posgrado e Investigación
UPIICSA

Abstracto.

Este artículo presenta un análisis comparativo entre dos métodos de evaluación de expresiones polinómicas. El primero corresponde a la forma tradicional (o directa) de evaluación y que implica el cálculo de las respectivas potencias de los términos de la expresión. El segundo método considerado es conocido como el Método de Horner, en el que se reorganiza el polinomio a manera de una serie de multiplicaciones anidadas de forma que no puedan evaluarse potencias mayores a una. El objetivo del análisis es presentar el costo computacional asociado a ambos métodos para así determinar cuál es más eficiente bajo un criterio completamente cuantitativo. Una introducción al tema de la complejidad computacional es presentado y conclusiones con respecto al uso de recursos de cómputo son incluidas.

Polinomios.

Un polinomio es una expresión formada por la suma de múltiplos de potencias de una variable independiente. Por ejemplo:

$$3x^4+5x^3+7x^2-2x+5. \quad (1)$$

Cada una de los elementos unidos por operaciones de suma o resta son llamados términos y el factor multiplicativo en cada término denominado coeficiente. El último elemento del polinomio, que no incluye a una variable, recibe el nombre de término constante debido a que implica la multiplicación de la variable independiente elevada a la potencia de cero.

El *grado* del polinomio queda determinado por el término de potencia mayor. El nombre del polinomio queda determinado por la cantidad de términos que lo forman. Un polinomio con un sólo término se llama monomio, contando con dos términos es un binomio, si cuenta con tres es un trinomio y así sucesivamente, aunque en la práctica se suele denominar polinomio a toda expresión de tres o más términos. De forma general, un polinomio P de grado n es expresado:

$$P_n(x)=a_n x^n+a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\ldots+a_2x^2+a_1x+a_0 \quad (2)$$

y cuya evaluación puede ser considerada como:

$$P_n(x)=\sum_{i=0}^{n} a_i x^i \quad (3)$$

Método de evaluación directa o tradicional.

La forma de evaluación con la que seguramente estaremos más familiarizados consiste en efectuar el cálculo correspondiente a cada término y sumando resultados parciales, reproduciendo lo representado por la ecuación 3. Si hemos de hablar del cálculo de este tipo de expresiones empleando un dispositivo de cómputo, entonces debemos poder contar con la posibilidad de efectuar las operaciones aritméticas implicadas, incluyendo la potenciación.

Por supuesto, la mayoría de los microprocesadores para computadores de escritorio y superiores incluyen dentro de su repertorio de instrucciones aquellas necesarias para efectuar las operaciones aritméticas básicas incluyendo la potenciación. Todas estas debidamente soportadas por circuitería dedicada, esto es, la lógica de cada instrucción se encuentra "alambrada" permitiendo que sea ejecutada lo más rápidamente posible.

El *alambrar una instrucción* implica un determinado costo, grado de complejidad y consumo de energía. Los primeros microprocesadores incluían sólo instrucciones para las cuatro operaciones aritméticas elementales por lo que operaciones de potenciación debían realizarse mediante una serie de multiplicaciones sucesivas. Esto mismo aplica hoy en día con microprocesadores baratos y de bajo consumo de energía destinados a dispositivos portátiles o que no están pensados para la realización de complejos cálculos matemáticos. Recurriremos a esta idea para nuestro análisis y retomaremos este punto más adelante.

Método de Horner.

El método de Horner es uno de esos "trucos" aritméticos y algebraicos a los que frecuentemente recurrían los programadores en las primeras décadas de la computación, y con los que buscaban hacer más eficiente un programa reduciendo tiempo de procesamiento o espacio en memoria o disco. La utilización de este tipo de reorganización de expresiones algebraicas o manejo de operaciones aritméticas ha caído muy en desuso por las facilidades que brindan los modernos lenguajes o interfaces de programación al permitir ingresar expresiones e ideas en la misma forma como estamos acostumbrados a verlas en materiales impresos o en nuestra vida cotidiana.

El método de Horner consiste en reexpresar un polinomio en forma de multiplicaciones anidadas, buscando (idealmente) que no haya términos de potencias mayores a uno. Así, la definición dada en la ecuación 2, quedaría reorganizada mediante este método como:

$$P_n(x)=(\ldots(((a_n x+a_{n-1})\,x+a_{n-2})\,x+\ldots+a_2)\,x+a_1)\,x+a_0 \quad (4)$$

y cuya evaluación puede conceptualizarse como:

$$P_n(x) = a_{i+1}x + a_i, \quad \forall i = n-1, n-2, \ldots, 1, 0 \tag{5}$$

Modelo.

Ciertamente los modernos lenguajes de programación y microprocesadores nos permiten llevar a cabo una operación de potenciación con tan solo expresarla a sabiendas que circuitos dedicados en el interior del computador cumplirán su cometido. Recurrir a tal facilidad para este análisis nos presenta una dificultad de medición, por lo que primero hemos de homogenizar criterios de medición.

- Como se mencionó anteriormente la realización de una operación de potenciación puede verse como una serie de multiplicaciones sucesivas. Por lo que cada término puede ser considerado como un elemento que requiere de *i* multiplicaciones (si incluye un coeficiente).
- Bajo este punto de vista, el método de Horner sólo altera la cantidad de multiplicaciones a realizar. La cantidad de operaciones de suma de términos no se altera.
- Debe considerarse que, independientemente del soporte por circuitería dedicada, la realización de una operación de multiplicación es más rápida que una de potenciación.
- Para efectos del presente análisis se considerará al "peor de los casos", aquel donde se requeriría del mayor número de operaciones para la evaluación del polinomio, i.e. cada término cuenta con un coeficiente y están presentes todos los términos (con potencias desde *n* hasta 1).

Así, se plantean dos algoritmos, cada uno representando la implementación de los métodos ya descritos y que nos permitirán contabilizar las multiplicaciones involucradas en cada caso. El primer algoritmo corresponde a la evaluación de un polinomio empleando multiplicaciones sucesivas para la evaluación de las respectivas potencias; este algoritmo implica dos ciclos, uno por los términos (con *n* como el grado del polinomio) y otro por la potencia de cada término, quedando de la siguiente manera:

Algoritmo 1: Evaluación Polinomial por Multiplicaciones Sucesivas.

A0. [INICIALIZAR]	Resultado ← 0
A1. [CICLO EN]	Ejecutar los pasos A2 hasta A5 para i = *n* hasta 1 con decrementos en 1. Al finalizar el ciclo ir a A6.
A2. [INICIALIZAR]	Término ← *variable*
A3. [CICLO EN j]	Ejecutar paso A4 para j = 2 hasta *i* con incrementos en 1. Ejecutar A5 al terminar ciclo j.
A4. [CALCULAR]	Término ← *variable* * Término
A5. [ASIGNAR]	Resultado ← Resultado + (*Coeficiente*$_i$ * Término)
A6. [FINALIZAR]	Desplegar Resultado

El segundo algoritmo que presentamos corresponde a la implementación del método de Horner. Una forma sencilla de entender cómo funciona este método es considerar que las multiplicaciones son

"distribuidas" de manera que cada producto, resultado de una multiplicación previa, se aprovechará implícitamente en las siguientes multiplicaciones, por ejemplo:

$$a_2x^2 + a_1x = x\,(a_2x + a_1) \quad (6)$$

$$a_3x^3 + a_2x^2 + a_1x = x\,(a_1 + x\,(a_2 + a_3x))$$
$$a_1x + x^2\,(a_2 + a_3x)$$
$$a_1x + a_2x^2 + a_3x^3. \quad (7)$$

La implementación del método de Horner queda como sigue:

Algoritmo 2: Método de Horner.

A0. [INICIALIZAR]	Resultado ← $Coeficiente_n$
A1. [CICLO EN i]	Ejecuta paso A2 para i = n-1 hasta 0 con decrementos en 1. Al finalizar el ciclo ir a A3.
A2. [ASIGNAR]	Resultado ← Resultado * x + $Coeficiente_i$
A3. [FINALIZAR]	Desplegar Resultado

Análisis.

Se presenta ahora el análisis de los algoritmos presentados. Primero bajo un enfoque analítico y posteriormente haciendo uso de técnicas de ajuste y regresión.

Enfoque analítico.

Como podrá observarse, la estructura del primer algoritmo contiene un ciclo externo que corre de 1 a n veces. Cuenta también con un ciclo interno de 2 a i veces, que implica i-1 multiplicaciones, lo que significa que podemos considerar de 1 a i veces por la multiplicación requerida por el coeficiente; concluyendo que, por cada término, se requieren i multiplicaciones.

El ciclo externo en este algoritmo implica que se ejecuten

$$n + n\text{-}1 + n\text{-}2 + \ldots + 2 + 1 \text{ multiplicaciones} \quad (8)$$

lo que es equivalente a:

$$n(n+1)/2. \quad (9)$$

Por su parte, el segundo algoritmo contiene un ciclo de 0 a n-1 veces implicando:

$$n \text{ multiplicaciones.} \quad (10)$$

Enfoque estadístico.

El análisis de algoritmos tiene mucho de arte. Requiere un conocimiento profundo del problema, dominio de herramientas analíticas, habilidad y experiencia para identificar lo que realmente se debe medir. El empleo de las matemáticas resulta indispensable, sobre todo cuando buscamos una expresión

que explique el comportamiento del algoritmo en estudio. El enfoque anterior es un ejemplo de esto, aunque claro, bastante sencillo. La mayoría de las ocasiones el problema requerirá un mayor grado de análisis y manipulaciones algebraicas. Afortunadamente, es posible recurrir a técnicas estadísticas que sólo requieren de una muestra de datos para derivar de ésta una ecuación que explique en forma aproximada la relación entre éstos. Complementariamente, el enfoque estadístico puede considerarse una forma de validación de los resultados del método analítico.

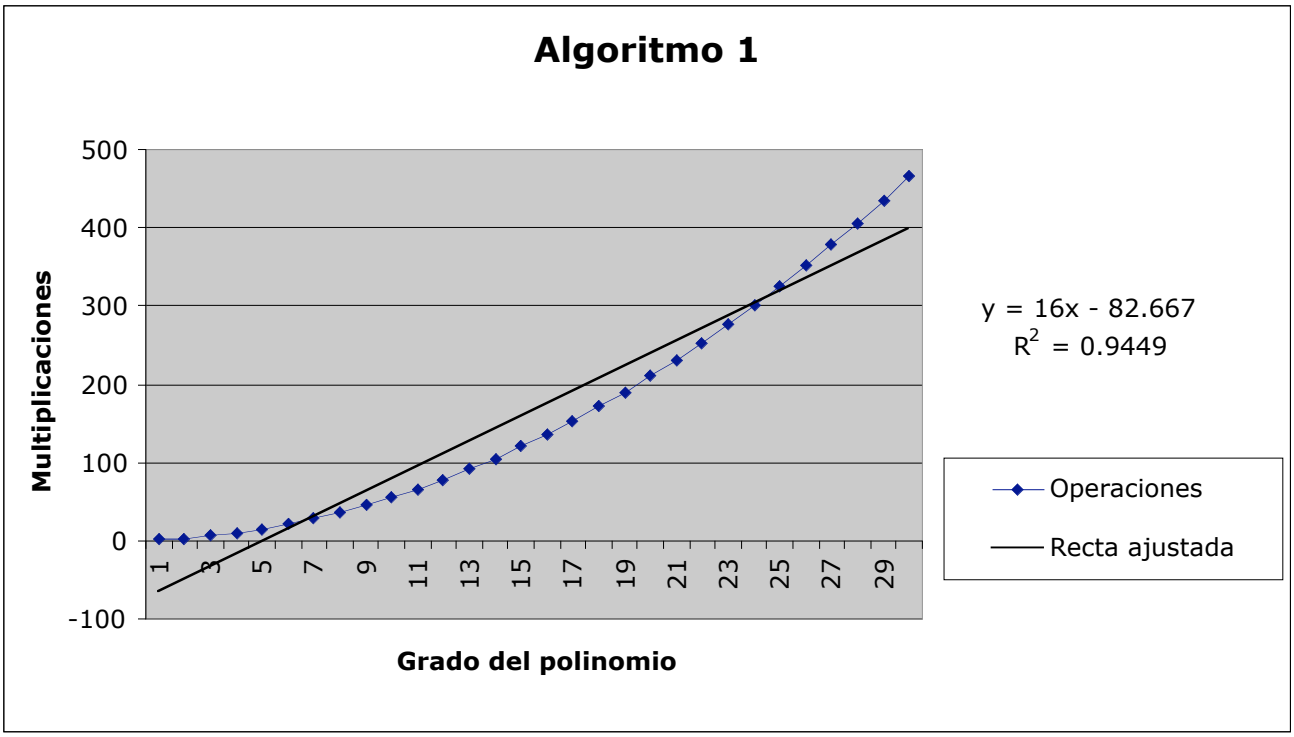
La implementación de los algoritmos descritos a través de un pequeño programa permitió generar el siguiente conjunto de datos:

Grado del polinomio	Algoritmo 1	Algoritmo 2
1	1	1
2	3	2
3	6	3
4	10	4
5	15	5
6	21	6
7	28	7
8	36	8
9	45	9
10	55	10
11	66	11
12	78	12
13	91	13
14	105	14
15	120	15
16	136	16
17	153	17
18	171	18
19	190	19
20	210	20
21	231	21
22	253	22
23	276	23
24	300	24
25	325	25
26	351	26
27	378	27
28	406	28
29	435	29
30	465	30

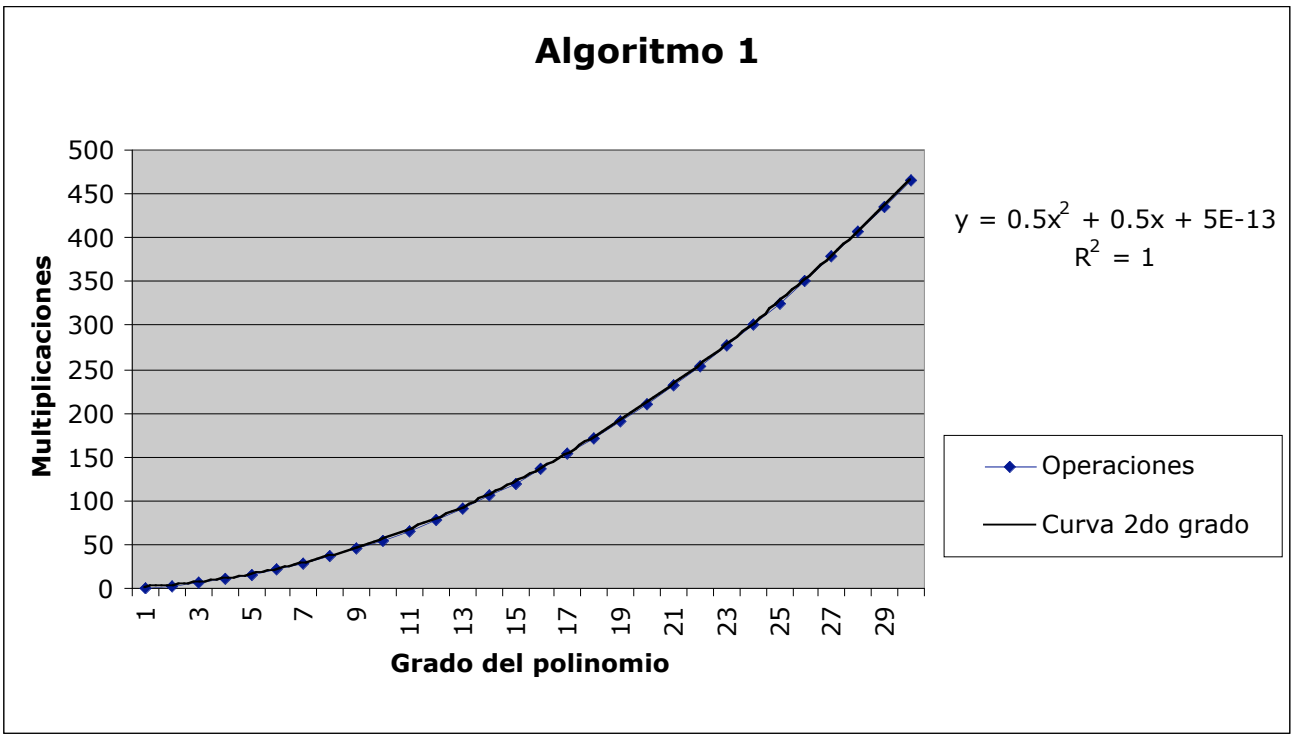
Tabla 1.- Número de multiplicaciones requeridas por los algoritmos.

La realización de los cálculos necesarios en la aplicación del método de mínimos cuadrados para el cálculo de las diferentes curvas de ajuste que podrían probarse están fuera del alcance del presente

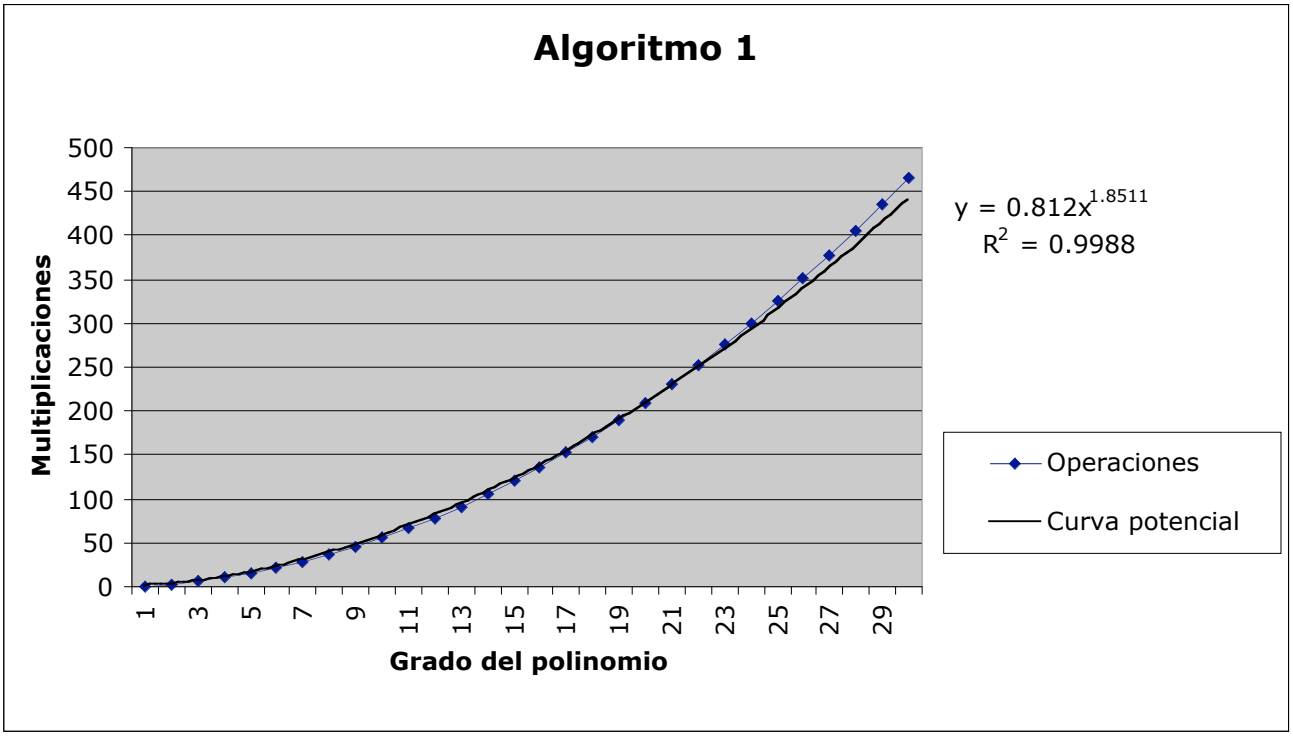
artículo. Por otra parte, existen muchísimas herramientas para realizar dichos cálculos, una de ellas es la hoja de cálculo Microsoft Excel. Presentamos las gráficas correspondientes al conjunto de datos junto con varias curvas de ajuste. Cada gráfica muestra la ecuación correspondiente a la curva calculada y su factor de correlación.



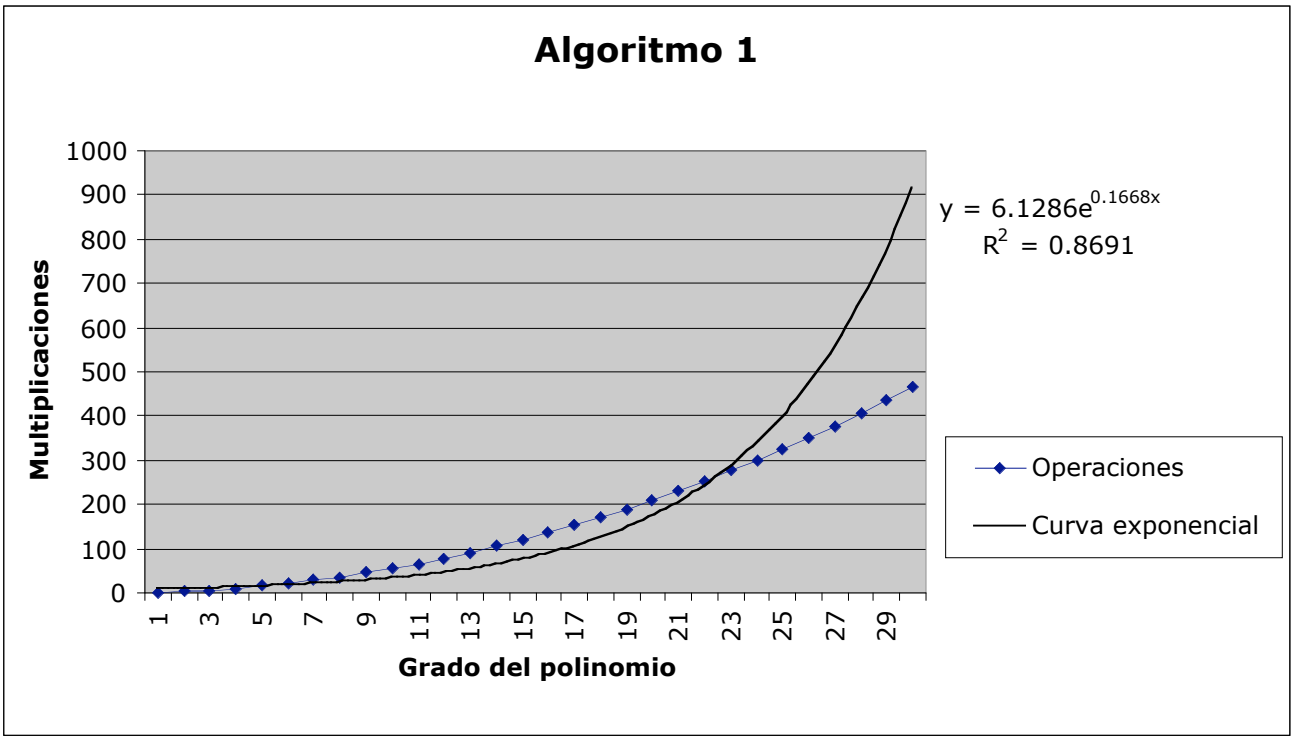
(a)



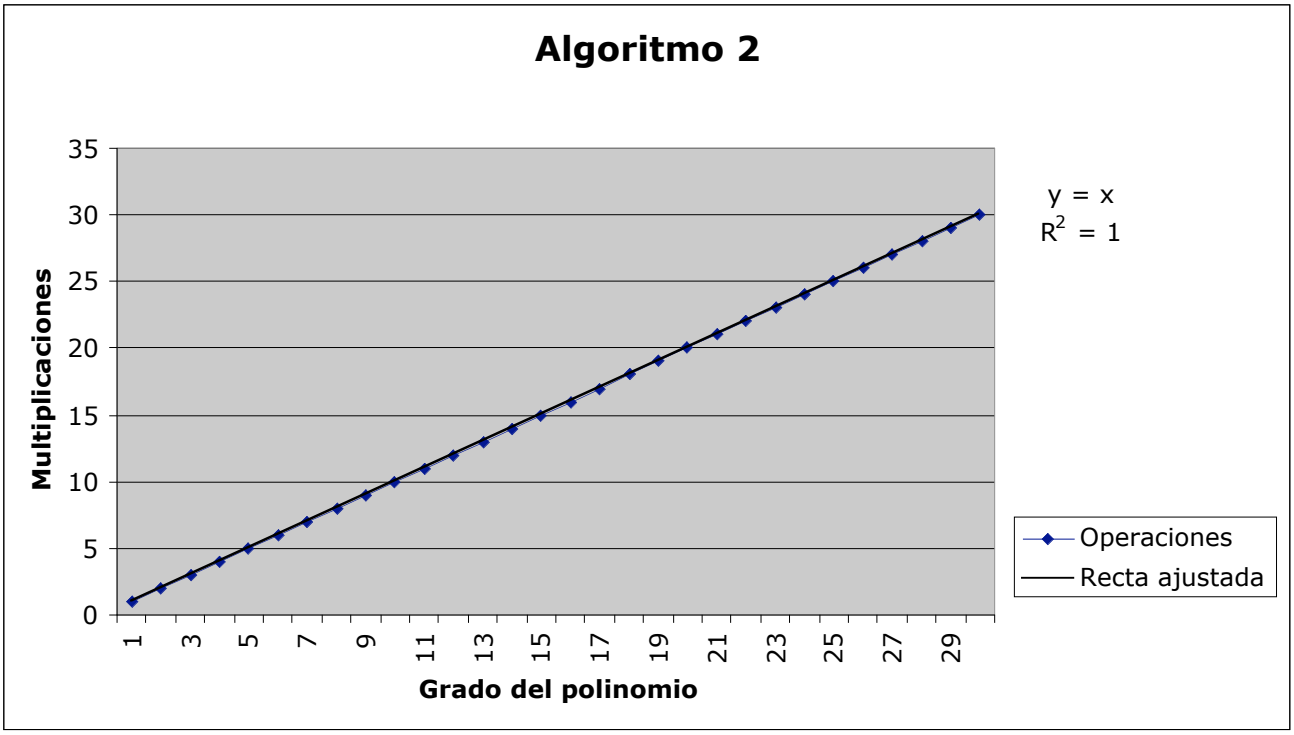
(b)


Algoritmo 1
Multiplicaciones
500
450
400
350
300
250
200
150
100
50
0
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
Grado del polinomio
y = 0.812x^1.8511
R^2 = 0.9988
Operaciones
Curva potencial

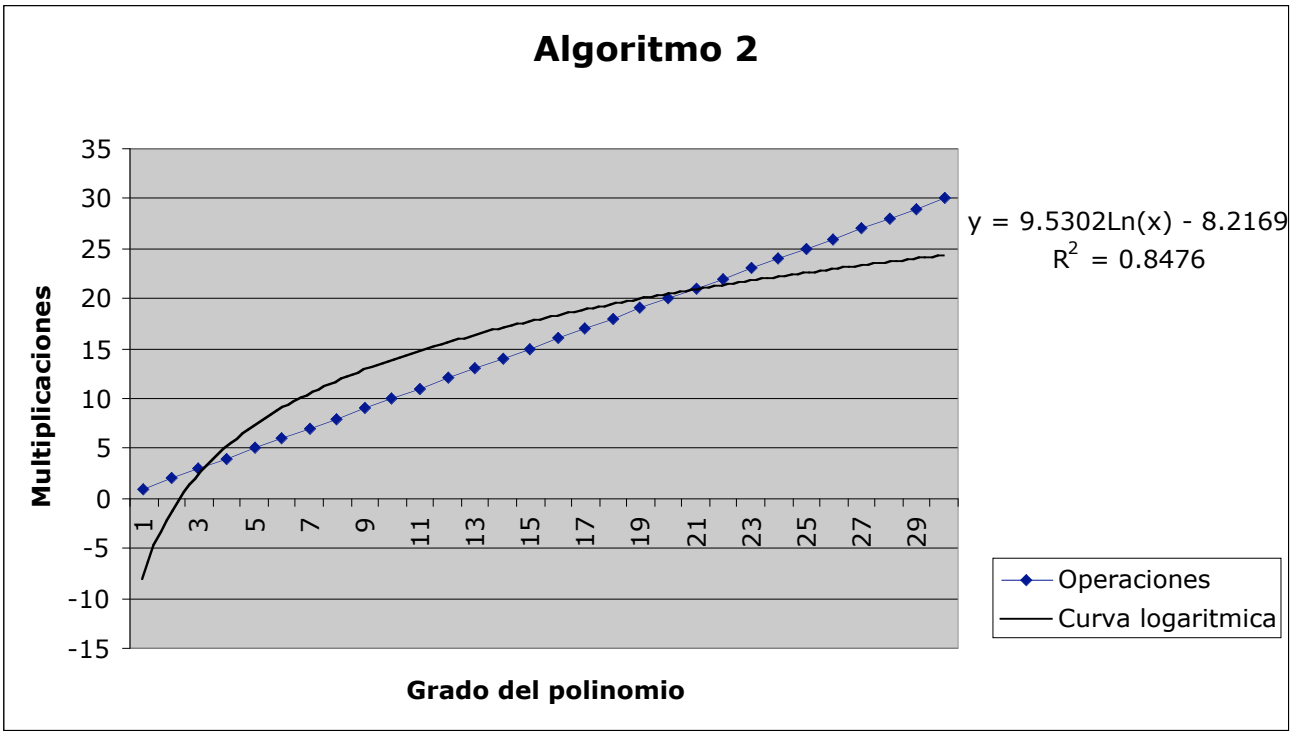
(c)


Algoritmo 1
Multiplicaciones
1000
900
800
700
600
500
400
300
200
100
0
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
Grado del polinomio
y = 6.1286e^0.1668x
R^2 = 0.8691
Operaciones
Curva exponencial

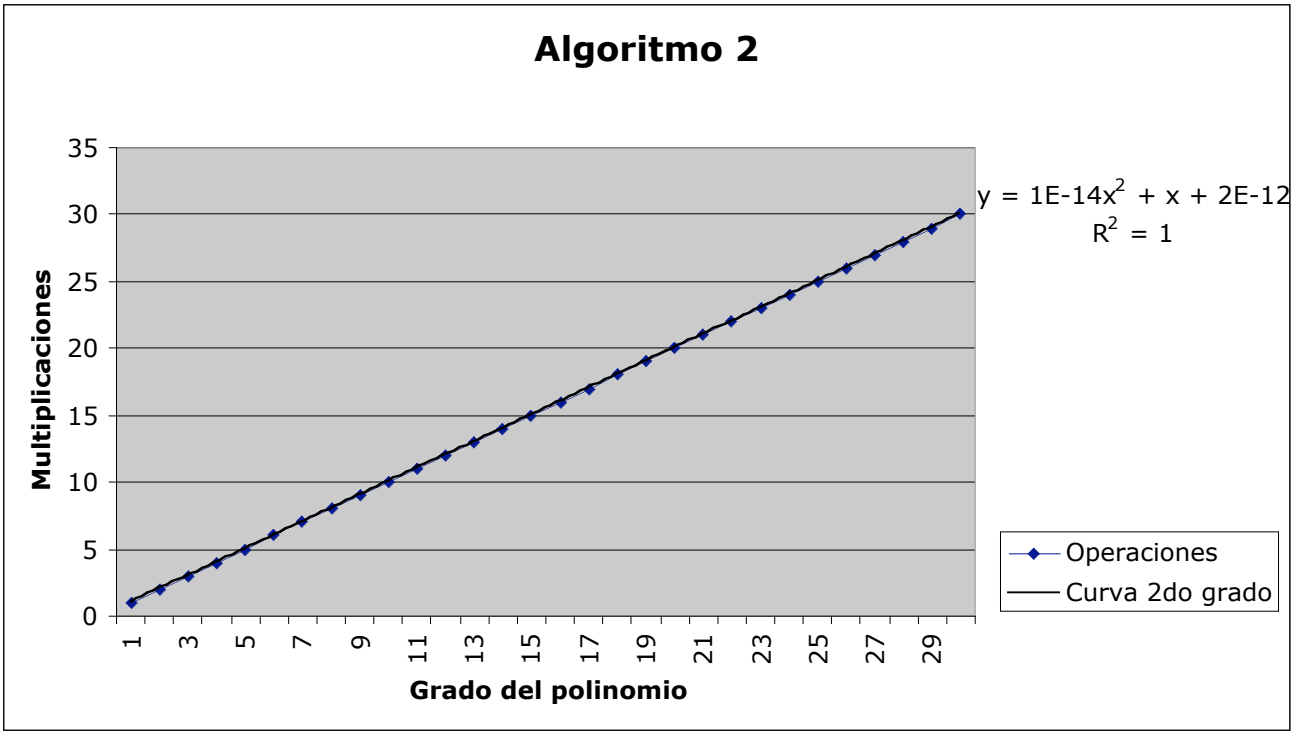
(d)


Algoritmo 2
Multiplicaciones
Grado del polinomio
y = x
R2 = 1
Operaciones
Recta ajustada

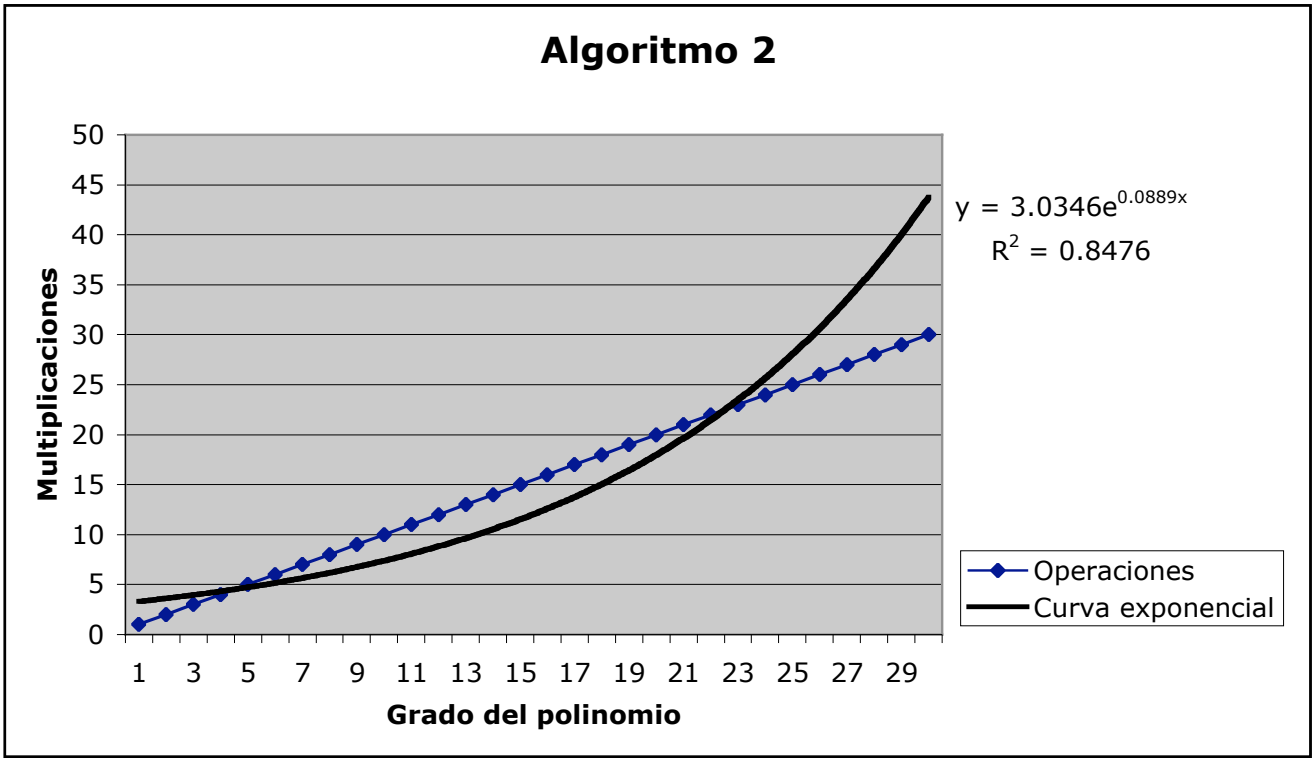
(e)


Algoritmo 2
Multiplicaciones
Grado del polinomio
y = 9.5302Ln(x) - 8.2169
R2 = 0.8476
Operaciones
Curva logaritmica

(f)



(g)



(h)

Figura 1.- Gráficas de datos y curvas de ajuste calculadas para el Algoritmo 1 (a, b, c, d) y el Algoritmo 2 (e, f, g, h).

Resultados.

Como puede verse, las curvas obtenidas por regresión confirman los resultados obtenidos analíticamente. Para el conjunto de datos correspondiente al Algoritmo 2 dos curvas presentan un grado de correlación igual a uno: la recta y la cuadrática. Examinando de cerca la ecuación correspondiente al polinomio de segundo grado puede verse que el coeficiente del término cuadrático es muy pequeño, tendiente a cero, con lo que el término se anula y la ecuación termina siendo lineal. Un resultado debido a errores de redondeo.

Complejidad computacional es un área de la ciencia de la computación que abarca el estudio del comportamiento de los algoritmos. La forma como el comportamiento de un algoritmo se describe es a través de una *función de trabajo*, una ecuación que describe cuanto tiempo de procesamiento o espacio en disco o memoria utiliza el procedimiento para llegar a una solución en función de los parámetros de entrada. Precisamente, el cálculo de las funciones de trabajo es lo que acabamos de hacer. Tenemos que la función de trabajo del Algoritmo 1 es

$$y=n(n+1)/2 \tag{11}$$

que podemos escribir como

$$y = \frac{n^2 + n}{2} \tag{12}$$

y así ver que efectivamente se trata de una función cuadrática, donde el parámetro de entrada es n, que es el grado del polinomio y *en función* del cual está la cantidad de términos a calcular.

Asociado al concepto de la función de trabajo está su *orden*, la forma en la que la función varía en magnitud a medida que sus parámetros tienden a un límite. Este se denota por una O mayúscula y es comúnmente referida como *O grande*. Una forma rápida y práctica de identificación del orden de una función es a través del elemento de mayor exponente o que presente un crecimiento mayor, que finalmente será el que fije la mayor magnitud en dicha función. Para el primer algoritmo tenemos $O(n^2)$

Para el Algoritmo 2 vemos que su función de trabajo es

$$y=n \tag{13}$$

y así su orden es $O(n)$.

Conclusiones.

Asociada a la ejecución de cada instrucción en un computador está un tiempo de realización. Un menor número de operaciones a realizar implica un menor tiempo del algoritmo para entregar resultados. Aquí se ha presentado, a través de un razonamiento metódico, cómo puede cuantificarse este esfuerzo asociado a un procedimiento determinado. La idea puede ser usada en el análisis de cualquier otro algoritmo.

Incidentalmente se presenta también un criterio que puede ser de mucha ayuda en la labor de ciertos programas de cómputo. En la resolución de muchos problemas de ingeniería y otras áreas científicas es muy común recurrir al cálculo de polinomios. Algunos de estos problemas requieren tiempo para ser resueltos. Con base en los resultados del presente artículo, podemos ver que una mejora sustancial en el tiempo de cálculo puede ser obtenida mediante la aplicación del método de Horner.

Los modernos lenguajes de programación nos permiten expresar ideas como estamos acostumbrados a hacerlo. Esto no necesariamente implica que sea la mejor forma de presentarlos para obtener la mejor utilización de los recursos de un computador, además de que no puede esperarse siempre que en todo compilador, intérprete o programa de aplicación se implementen técnicas de optimización como la aquí descrita. Es responsabilidad de todo programador profesional escribir programas de forma que garanticen el más eficiente uso de los recursos del computador.

Referencias.

Valerie Illingworth, Gen. Ed.; **Diccionario de Informática**; Ediciones Díaz de Santos S.A.; Madrid, España, 1993. ISBN 84-7978-068-1.

Apuntes de curso; *Arquitectura de Máquinas*. Semestre de primavera, 2001; UPIICSA.

Deitel y Deitel; **C++. Cómo Programar**; Prentice-Hall, 2da edición, 1999.

S. E. Goodman & S. T. Hedetniemi; **Introduction to the Design and Analysis of Algorithm**; 5th Ed., McGraw-Hill International Editions; Singapore, 1988. ISBN 0-07-Y66300-9.