Reporte de resultados Proyecto 1

Arturo Chinchilla Sánchez, Jeremy Rodríguez Solórzano, Santiago Gamboa Raírez
mchinchilla11@gmail.com jrodriguezs0292@gmail.com santigr17@gmail.com
Área Académica de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

***Resumen*—El propósito de este proyecto es la implementación en C++ de métodos numéricos para la Deflación y la búsqueda de raíces reales y complejas de un polinomio, como el método de Müller y Laguerre. Para ello se hace uso de la biblioteca boost que ofrece una gran cantidad de estructuras de datos y algoritmos, como la clase boost::math::tools::polynomial que ofrece una manera sencilla de representar y manipular polinomios. Además se realiza un análisis de desempeño en tiempo de ejecución y número de operaciones básicas empleadas utilizando el perfilador ——- para los algoritmos de Müller y Lagarre.**

***Palabras clave*—Deflación, Muller, Laguerre**

I. Introducción

En el ambiente del análisis numérico la presentación de problemas sobre este ámbito lleva consigo factores que incluyen, no solo la resolución exitosa de los mismos, sino también el uso de técnicas apropiadas y eficientes para lograr respuestas, además de deseadas, satisfactorias. Es por ello, que dentro de una amplia gama de maneras de solucionar un problema, es de suma importancia dedicar tiempo al análisis de conceptos que permitan entender el porqué de un método específico usado. Siguiendo este aspecto, la solución de raíces para polinomios destaca diversas maneras de manejarla, algunas de ellas son los métodos de: Müller y Laguerre; ambos acompañados de la deflación polinomial. La deflación polinomial se considera un método de suma importancia para el trabajo con polinomios, cuando se requiere encontrar soluciones. Según Chapra et al (2007), esta trata de eliminar raíces ya encontradas para continuar con el cálculo de las mismas de forma iterativa, de manera que se evite volver a toparse con la misma raíz cada vez que se realiza el cálculo [1]. Esta se usa en conjunto con los métodos de, Müller, el cual consiste en obtener una raíz, a partir del cálculo de coeficientes de una parábola que pasa por tres puntos [1], y Laguerre, que aproxima raíces reales y complejas [1]. De esta forma, Se logra una complementación entre métodos que permita comparar resultados de acuerdo a la eficiencia de cada uno (Laguerre y Müler). Para este proyecto, se requiere, precisamente, implementar los métodos (ya descritos), con sus respectivos algoritmos desarrollados en C++. Para que de esta manera el usuario del programa pueda visualizar la diferencia en cuanto a tiempo de resolución y complejidad de cada método, y ejecutar comparaciones que permitan explicar la eficiencia de cada uno. Se pretende que al finalizar esta práctica, se logre una comprensión óptima de los métodos numéricos usados para la solución de raíces de polinomios, y que se evalúe el desempeño de los mismos a partir de los resultados obtenidos con cada prueba.

II. Estado del Arte

Según [1] los polinomios tienen muchas aplicaciones en la ciencia y en la ingeniería. Por ejemplo, se usan mucho en el ajuste de curvas. Aunque se considera que una de las aplicaciones más interesantes y potentes es la caracterización de sistemas dinámicos y, en particular, de sistemas lineales. Algunos ejemplos son los dispositivos mecánicos, las estructuras y los circuitos eléctricos. Se analizarán ejemplos específicos en el resto del texto. Éstos, en particular, se enfocarán a varias aplicaciones en la ingeniería.

III. Propuesta

Para este proyecto, el uso de la biblioteca *boost* para el manejo de algoritmos matemáticos es de suma importancia. Más específicamente, se usa *boost/math/tools/polynomial.hpp* para la manipulación de polinomios. Según [2], Esta biblioteca, junto con el uso del espacio de nombre *boost::math::tools*, permiten convertir, almacenar y evaluar (mediante el método de Horner) un arreglo común de números, en un polinomio de grado *n* ordenado de forma ascendente, tal y como se muestra en la ecuación (1):

$$P(x) = a + bx + cx^2 + ... + dx^n \tag{1}$$

Estos polinomios se admiten con los coeficientes (*a, b, c, d, ...*) en formato tanto *double* como *float*; gracias a que los métodos han sido implementados con ayuda de plantillas de C++. El uso de los métodos de Müller, Laguerre y las deflaciones tanto para raíces reales como reales y complejas, se implementan de la siguiente manera:

III-A. Laguerre

Se usó como ayuda el material "Numerical Recipes in C"de Flannery et. al (1992) [3]. Como parámetros usa:

- Un polinomio de tipo arreglo con elementos de números complejos de tipo T (a[]).
- El orden del polinomio usado, de tipo entero (m).
- El puntero a una raíz conocida (*x).
- Un número de iteraciones máximo (*its).

El método retorna void.

III-B. Müller

Se usó como ayuda el material "Numerical Recipes in C"de Flannery et. al (1992) [3]. Como parámetros usa:

- Un vector iniciales para ejecutar el método (T x0, T x1. T x2).

- El puntero hacia un vector de elementos complejos de tipo T (& coefs).
- Un número de iteraciones máximo de tipo entero sin signo (max).
- Una tolerancia para llevar el método a la convergencia, de tipo double (pTolerance).

La función se encarga de retornar un vector con elementos complejos de tipo T, que posee las raíces del polinomio.

III-C. Deflación para raíces reales

Como parámetros usa:

- El puntero a un polinomio de tipo polynomial con coeficientes de tipo T (& poly).
- Un puntero tipo T hacia la raíz conocida usada para deflacionar (& root).
- Un puntero hacia un polinomio tipo polynomial de coeficientes tipo T, que representa el residuo que se va a ir almacenando en las iteraciones de la deflación (& residuo).

La función retorna el polinomio ya deflacionado, de tipo polynomial con coeficientes tipo T.

III-D. Deflación para raíces complejas y reales

Para esta función, se usa como parámetros:

- El puntero a un polinomio de tipo polynomial con coeficientes de tipo T (& poly).
- Un puntero tipo complex con elementos tipo T hacia la raíz conocida usada para deflacionar (& root).
- Un puntero hacia un número complejo de coeficientes tipo T, que representa el residuo que se va a ir almacenando en las iteraciones de la deflación (& residuo).

La función retorna el polinomio ya deflacionado, de tipo polynomial con coeficientes tipo T.

IV. Resultados

Por medio de la herramienta Valgrind se obtuvo los siguientes resultados:

Cuadro I
RESULTADOS POR MEDIO DE LA HERRAMIENTA VALGRIND

Función	Llamadas	Costo interno	Costo incluyendo todas las llamadas de funciones
Müller	3	0.01	9.10
Laguerre	10	1.59	13.43

Cuadro II
RAICES ENCONTRADAS CON LOS POLINOMIOS DE PRUEBA

Función	Polinomio	Raíz x1	Raíz x2	Raíz x3	Raíz x4
Müller	$x^4 + x^3 + 7x^2 + 9x - 18$	1,0	-2,0	0,3	-0,-3
Laguerre	$x^4 + x^3 + 7x^2 + 9x - 18$	1,0	-3.1368e-09,3	-2.42374e-07,-3	-2,0
Müller	$x^3 + 2x^2 + 9x + 18$	-2,0	0,3	-0,-3	na
Laguerre	$x^3 + 2x^2 + 9x + 18$	-2,0	-1.64443e-09,3	-2.40892e-07,3	na
Müller	$x^3 + 6x^2 + 3x - 10$	-5,0	-2,0	1,0	na
Laguerre	$x^3 + 6x^2 + 3x - 10$	-5,0	-2,0	1,0	na

V. Conclusiones

Los algoritmos implementados y evaluados en el proyecto permiten operaciones en polinomios tanto con raíces reales como complejas. Además, describen resultados esperados e iguales o aproximados a los evaluados de forma teórica, con un porcentaje de error no mayor a 1. Como se puede observar en el Cuadro II el método de Laguerre mostró resultados diferentes pero aproximados al valor real, mientras que el método de Müller brindó soluciones a las raíces iguales a los valores reales (0 por ciento de error). La deflación polinomial se implementó de manera exitosa, y muestra resultados totalmente satisfactorios en los polinomios evaluados. Las pruebas unitarias, a pesar de saber la importancia que representan para los métodos numéricos, no se lograron implementar, debido al factor del tiempo. El desempeño de los algoritmos se evaluaron con pruebas manuales mediante la entrada de diferentes polinomios de grados distintos. Se evaluó el desempeño de los métodos de Müller y Laguerre en tiempo de ejecución y número de operaciones básicas empleadas (gprof en este caso), se implementó pero no dentro del proyecto en general por razones de errores de compatibilidad con la biblioteca *boost*. En lugar de ello se usó con los métodos de Laguerre y Müller de forma separada con casos de prueba seleccionados. Además, Se obtuvieron valores en el área del tiempo de ejecución no deseados (cero segundos en todos los casos). Por esta razón, se cambió y prefirió el uso de la herramienta Valgrind, con ella los resultados esperados fueron evaluados de una manera precisa y sin errores. Se evaluó el número de llamadas para cada función y con ellas el costo interno e incluyendo todas las llamadas de funciones en total, para mostrar los resultados en tablas.

Referencias

[1] Chapra, S. C, Canale, R. P, Métodos numéricos para ingenieros. (5th. ed.). Mexico: McGraw-Hill Interamericana, 2007

[2] boost c++ libraries, "Polynomial and rational function evaluation", *boost.org*, 2010. [online]. Available: http://www.boost.org/doc/libs/1_46_0/libs/math/doc/sf_and_dist/html/math_toolkit/ toolkit/internals1/rational.html. [Accessed March 14, 2018].

[3] Flannery, B. P, Press, W. H, Teukolsky, S. A and Vetterling, W. T, Numerical Recipes in C. (2nd. ed.). British: Cambridge University Press, 1992