Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
Profesor: Dr. Pablo Alvarado Moya
CE-3102 Análisis Numérico para Ingeniería
I Semestre 2017

Fecha de entrega: **24 de marzo, 2017**

Proyecto 2

1 Introducción

En la comunidad de C++, la biblioteca `boost` ofrece gran cantidad de estructuras de datos y algoritmos. De allí, en esta tarea partirá de la clase `boost::math::tools::polynomial`, que ofrece una forma sencilla de representar polinomios.
Dicha propuesta utiliza el método `evaluate` para evaluar el polinomio, y carece de la implementación del `operador()` para evaluar como functor. No ofrece la facilidad de realizar la división polinomial.
En este proyecto usted ofrecerá la funcionalidad adicional necesaria para realizar dicha división polinomial y además implementará dos métodos para la búsqueda de raíces de polinomios.
Usted puede utilizar el evaluador de polinomios realizado en el proyecto 1, si lo considera conveniente.

2 Objetivos

2.1 Objetivo general

Implementar y evaluar algoritmos para búsqueda de raíces de polinomios, capaces de encontrar todas las raíces reales y complejas.

2.2 Objetivos específicos

1. Implementar un algoritmo de división polinomial
2. Implementar el método de Müller para búsqueda de raíces
3. Implementar el método de Laguerre para búsqueda de raíces
4. Utilizar deflación polinomial para buscar las raíces.
5. Evaluar el desempeño de los algoritmos implementados

3 Procedimiento

1. Revise el método de evaluación de polinomios implementado en `boost`. ¿Qué tipo de algoritmo utilizan?
2. Implemente una función externa para realizar la división polinomial de dos polinomios. La función recibe dos polinomios de entrada: dividendo y divisor, y produce un polinomio con el cociende a la salida, y devuelve el residuo de la división como un tercer argumento:

```
polynomial<T> divide(const polynomial<T>& dividendo, const
    polynomial<T>& divisor, polynomial<T>& residuo)
```

3. Implemente los métodos de Müller y de Laguerre y compare sus convergencias en la búsqueda de todas las raíces de un polinomio.

   Para la presentación de resultados utilice iniciando desde 0 para el polinomio

   $$f(x) = (x-2)(x^2+4)(x+1)$$

   y además elija dos polinomios: uno con al menos tres raíces reales y otro con raíces reales y complejas, para probar su algoritmo.

   Deben encontrarse todas las raíces, por lo que se debe realizar deflación polinomial. Para ello refiérase al libro "Numérical Recipes" de Press et al.

   Asegúrese de pulir sus raíces, para reducir el error.

4. Evalúe en lo anterior la influencia de usar precisión simple (`float`) y doble (`double`), así como el efecto de pulir o no las raíces. Usted debe decidir y justificar qué tipo de error utilizar.

5. Evalúe el desempeño de su implementación en tiempo de ejecución y número de operaciónes básicas empleadas.

6. Observe que hay raíces complejas, y por tanto debe usar como tipo numérico a `std::complex<float>` y `std::complex<double>` en los casos que sea necesario. Si su algoritmo se use con tipos `float` o `double`, puede ser usado solo para encontrar raíces reales, y en caso de que el algoritmo detecte que hay raíces complejas, debe reportar una advertencia.

7. Realice un artículo formal, donde presente los resultados obtenidos. El esquema de un artículo formal lo puede encontrar aquí, en donde el énfasis debe darlo a las secciones de la propuesta para describir su implementación y los resultados con las evaluaciones correspondientes. Dicho artículo no debe tener más de 3 páginas.

4 Entregables

El código fuente y el artículo científico deben ser entregados según lo estipulado en el programa del curso.
Incluya un archivo de texto README con las instrucciones para compilación y ejecución del programa.