Tarea 1

Algoritmos y Complejidad

"El título queda de ejercicio"

1. Introducción

Puntos de Chebyshev

Recordemos que el error de interpolación de un polinomio $Q_n(x)$ de grado $n-1$ (determinado por n puntos: $x_1, x_2, \ldots, x_n$) en un punto x está dado por:

$$f(x) - Q_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\zeta) \prod_{1 \le i \le n} (x - x_i)$$

Donde ζ pertenece al intervalo que encierra el x y los x_i.
Los puntos de Chebyshev son los puntos $x_1, x_2, \ldots, x_n$ que logran reducir lo más posible el valor (absoluto) máximo de la parte

$$\prod_{1 \le i \le n} (x - x_i)$$

en el intervalo $[-1, 1]$, y valen:

$$x_{\text{chev}\,i} = \cos\left(\frac{2i-1}{2n}\pi\right) \qquad \text{para } i \in \{1..n\}$$

Sin embargo, es posible realizar una transformación para calcular estos puntos para cualquier intervalo $[a, b]$.

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} x_{\text{chev}\,i}$$

Ahora, cuando se utilizan los puntos de Chebyshev, el valor máximo que puede adquirir la productoria es:

$$\left| \prod_{1 \le i \le n} (x - x_i) \right| \le \frac{\left(\frac{b-a}{2}\right)^n}{2^{n-1}}$$

2. Problemas

Problema 1

Encontrar una cantidad n (de puntos de Chebyshev) que permita construir un polinomio interpolador $Q_n(x)$ para la función $f(x) = \ln\left(\frac{x}{2}\right)$ de tal manera que el error de interpolación para cualquier $x \in [\frac{1}{2}, 2]$ sea:

$$|f(x) - Q_n(x)| \leq K$$

Donde K es una constante positiva cualquiera.
Nota: No es necesario que encuentre una expresión explícita para n.

(30 puntos)

Problema 2

Implemente un programa en *Python* con la siguiente funcionalidad:

1. Recibe una función $f(x)$ del usuario y una serie de puntos x_j en los que evaluar esa función.
2. Construye un polinomio interpolador $p(x)$ que pase por esos puntos.

   No importa cómo el programa represente el polinomio internamente, sólo que pueda evaluarlo en cualquier punto x.
3. Recibe los extremos a y b de un intervalo; y un valor h.
4. Utiliza estos valores para entregar aproximaciones de la integral del error de interpolación en el intervalo $[a, b]$, mediante:

   a) Sumar rectángulos de longitud h, utilizando el punto medio de cada uno.

   b) Utilizar la cuadratura gaussiana con 3 puntos.

   Nota: Tendrá que utilizar un cambio de variable para que pueda llevar la integración de $[a, b]$ a $[-1, 1]$.

Responda: ¿Cómo podría utilizar su programa para calcular aproximaciones a la integral de $f(x)$ y no a la integral del error de interpolación $f(x) - Q_n(x)$?

(40 puntos)

Problema 3

Para la función $f(x) = x^5 + x^4 + x^3 + x^2 + x + 1$ en el intervalo $[1, 3]$, y considerando un polinomio interpolador para dicha función con 4 puntos de Chebyshev.

1. Calcule, matemáticamente, el valor numérico de la integral del error de interpolación.

2. Utilice su programa para obtener las dos aproximaciones de dicha integral, para la de rectángulos utilice $h = 0,5$.

3. ¿El error de su programa al integrar la función de error con el método de los rectángulos concuerda con la aproximación del error vista en clases?

4. ¿El error de su programa al integrar la función de error con la cuadratura gaussiana concuerda con la teoría?

(30 puntos)

Ayudita

Puede utilizar la siguiente línea de código para recibir una función dependiente de x y almacenarla.

```
exec "funcion= lambda x: "+raw_input("inserte f(x)= ")
```

Posteriormente la puede evaluar en x usando:

```
y = funcion(x)
```

Esto funciona porque exec es un comando que toma un string y lo ejecuta como código, es una opción extremadamente insegura ya que el usuario del programa puede poner cualquier cosa desastrosa, pero nos permitirá probar su código con cualquier función matemática que insertemos por la entrada estándar.

Nota: Recuerde, al principio del programa, colocar

```
from math import *
```

3. Condiciones de entrega

- La tarea se realizará *individualmente* (esto es grupos de una persona), sin excepciones.
- La entrega debe realizarse vía Moodle en un *tarball* en el área designada al efecto, bajo el formato `tarea-1-rol.tar.gz` (`rol` con dígito verificador y sin guión).
  Dicho *tarball* debe contener dos directorios:
  - Un directorio `tarea`, que contenga las fuentes en LaTeX (al menos `tarea.tex`) de la parte escrita de su entrega, además de un archivo `tarea-1.pdf`, correspondiente a la compilación de esas fuentes.
  - Un directorio `programa`, que contenga su código *Python* con al menos un archivo `main.py` como punto de entrada y un `README` indicando la versión de *Python* utilizada y, opcionalmente, información adicional que deba saber el evaluador.

- Además de esto, la parte escrita de la tarea debe en hojas de tamaño carta en Secretaría Docente de Informática (Piso 1, edificio F3).
- Tanto el *tarball* como la entrega física deben realizarse el día indicado en Moodle.
  Por cada día de atraso se descontarán 20 puntos y a partir del tercer día de atraso no se reciben más tareas y la nota es automáticamente cero.

4. Revisión del código

Para evaluar el código se considerarán los siguientes aspectos:
Programa

- Uso adecuado de funciones 10 %
- Uso de estructuras de control 15 %
- Uso de estructuras de datos 10 %
- Código claro y simple 15 %
  - Nombres adecuados.
  - Indentación correcta.
  - Comentarios suficientes.
  - Ausencia de código comentado.
- Ejecución correcta 50 %
  - Ausencia de errores.
  - Resultados correctos en casos de prueba.