Primer examen parcial del curso de Métodos Numéricos

Fecha del examen: Octubre 22, 2020

Hora de inicio: 16:30
Hora de entrega: 19:30

Indicaciones para el envío del examen:

1. Crea un archivo en el que puedas agregar texto e imágenes, y que después lo puedas exportar a PDF.
2. Comienza escribiendo tu nombre.
3. Antes de poner la respuesta, escribe a qué ejercicio corresponde.
4. Escribe el nombre del archivo que se tiene que pasar al interprete de Python, o en el caso de C/C++, el archivo que se tiene que compilar. Si requiere alguna instrucción especial para compilarlo, agrega el comando.
5. Si el programa recibe parámetros, indica en que orden se tienen que proporcionar.
6. Da al menos un ejemplo de la manera en que se tiene que ejecutar tu programa y agrega la salida del programa, poniendo directamente el texto que imprime tu programa o poniendo una imagen.
7. Si se te pide generar una gráfica de algún resultado, agrégala al archivo de respuestas.
8. Si tiene algun comentario, agrégalo también al documento de respuestas.
9. Trata de dejar algún espacio en blanco entre los ejercicios, por si el revisor quiere poner comentarios.
10. Sube los códigos en un archivo ZIP para que no tengas que subir archivo por archivo. Recuerda no adjuntar los executables y los códigos de cada ejercicio deben estar en carpetas diferentes.

11. Convierte el archivo de respuestas a un PDF y súbelo al classroom como archivo independiente.

12. Revisa que estén los archivos que quieres enviar antes de presiona el botón para enviar tu trabajo.

Ejercicio 1. (5 puntos)
Usando los códigos de la Tarea 2, escriba un programa que trate de obtener todas las raíces reales del polinomio mediante lo que se conoce como el método de Newton-Horner.

En la Tarea 2 se menciona que el Algoritmo 1 permite evaluar el siguiente polinomio $p_n(x)$ de grado n en un punto x_0 proporcionando sus coeficientes a_i:

$$p_n(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \cdots + a_nx^n$$

Además el Algoritmo 1 proporciona los coeficientes b_i del polinomio $q_{n-1}(x;\, x_0) = b_1 + b_2x + b_3x^2 + ... + b_nx^{n-1}$ de grado $n-1$, que resulta al dividir el polinomio $p_n(x)$ entre $x - x_0$:

$$p_n(x) = q_{n-1}(x;\, x_0)(x - x_0) + b_0 \tag{1}$$

Esto permite calcular el valor de la derivada de $p_n(x)$ en el punto x_0, que es igual a evaluar al polinomio $q_{n-1}(x;\, x_0)$ en $x = x_0$:

$$p_n'(x_0) = q_{n-1}(x_0;\, x_0). \tag{2}$$

De este modo, podemos aplicar el método de Newton-Raphson para generar el siguiente punto de la sucesión para encontrar una raíz de $p_n(x)$, mediante la expresión

$$x_1 = x_0 - \frac{p_n(x_0)}{p_n'(x_0)} = x_0 - \frac{p_n(x_0)}{q_{n-1}(x_0;\, x_0)}. \tag{3}$$

De esta forma, podemos generar una secuencia de valores $x_1, x_2, ...$ hasta que se cumpla que $|p_n(x_k)|$ sea menor que una tolerancia dado o se realicen el número máximo de iteraciones.

Por otra parte, si r_0 es una raíz del polinomio $p_n(x)$, entonces $b_0 = p_n(r_0) = 0$, de modo que $p_n(x) = (x - r_0)q_{n-1}(x;\, r_0)$. Esto permite diseñar una estrategia para tratar de encontrar todas las raíces reales del polinomio $p_n(x)$, y este procedimiento se describe en el Algoritmo 2, y en resumen hace lo siguiente:

1. Aplica el método de Newton-Raphson para tratar de encontrar una raíz r_0 del polinomio $p_n(x)$.

2. Si se obtiene la raíz r_0, utiliza el método de Horner para obtener los coeficientes del polinomio $q_{n-1}(x;\, r_0)$ de grado $n - 1$.

3. Como $q_{n-1}(x;\, r_0)$ y $p_n(x)$ tienen las mismas raíces, aplica el algoritmo de Newton-Raphson para tratar de obtener una raíz r_1 de $q_{n-1}(x;\, r_0)$.

4. Si el algoritmo encuentra a la raíz r_1, ya tenemos dos raíces de $p_n(x)$, y nuevamente, al aplicar el algoritmo de Horner al polinomio $q_{n-1}(x;\, r_0)$ permite obtener los coeficientes del polinomio $q_{n-2}(x;\, r_1)$ de grado $n-2$ tal que $q_{n-1}(x;\, r_0) = (x - r_1)q_{n-2}(x;\, r_1)$.

5. El algoritmo continúa de esta manera, aplicando el método de Newton-Raphson para obtener una raíz de polinomios cuyo grado va decreciendo conforme se van encontrando las raíces del polinomio original $p_n(x)$, y esto puede permiter hallar encontrar gradualmente todas sus raíces reales.

Algoritmo 1: Método de Horner

Entrada: El grado n del polinomio, el arreglo `a` con los coeficientes del polinomio $p_n(x) = a[0]x^n + a[1]x^{n-1} + ... + a[n-1]x + a[n]$ y el punto x_0.

Resultado: El valor de la evalución $y = p_n(x_0)$ y el arreglo `b` con los coeficientes del polinomio asociado $q_{n-1}(x;\, x_0)$.

Reservar memoria para el arreglo `b` con n elementos;
`b`$[0] =$ `a`$[0]$;
for $k = 1, 2, ..., n-1$ **do**
  `b`$[k] =$ `a`$[k] -$ `b`$[k-1]x_0$;
end
$y =$ `a`$[n] -$ `b`$[n-1]x_0$;

Entonces el ejercicio consiste en programar el Algoritmo 2 usando el código de la Tarea 2 y probarlo.

1. Escriba la función que corresponde al Algoritmo 2.

2. Escriba un programa que trate de calcular las raíces del polinomio

$$p_5(x) = \frac{x^5}{5} + 3x^4 + \frac{101x^3}{20} - \frac{129x^2}{2} - \frac{483x}{4} + \frac{585}{2}.$$

   Use $x_0 = 0$, $\tau = \sqrt{\epsilon_m}$ y $N_{\max} = 500$ para la función del inciso anterior.

3. Si la función del Algoritmo 2 devuelve $m > 0$, es decir, al menos encontró una raíz, entonces haga que el programa imprima el numero de raíces m y los valores $r_0, r_1, ..., r_{m-1}$.

4. Obtenga la raíz más pequeña y la más grande:

$$r_{\min} = \min_{0 \le i < m} r_i, \qquad r_{\max} = \max_{0 \le i < m} r_i.$$

Genere 50 puntos equidistantes del intervalo, $r_{\min} - 0.5 = x_0 < x_1 < ... < x_{49} = r_{\max} + 0.5$, y haga que el programa escriba en un archivo de texto una tabla en donde cada fila tiene los valores

$$x_i \qquad p_n(x_i)$$

5. Use los datos de la tabla para generar la gráfica de $p_5(x)$ en el intervalo $[r_{\min} - 0.5,\ r_{\max} + 0.5]$ y agregue esta gráfica al reporte.

Algoritmo 2: Método de Newton-Horner

Entrada: El grado n del polinomio, el arreglo `a` con los coeficientes del polinomio $p_n(x) = a[0]x^n + a[1]x^{n-1} + ... + a[n-1]x + a[n]$, el punto inicial x_0, una tolerancia τ y el número máximo de iteraciones $N_{\max}$.

Resultado: Un arreglo `r` con las raíces y el número m raíces calculadas.

Crear una copia `c` del arreglo `a`;
Crear el arreglo `r` de tamaño n;
for $m = 1, 2, ..., n$ **do**
    $x = x_0$;
    $res = False$;
    for $k = 1, 2, ..., N_{\max}$ **do**
        $px, \texttt{b} \leftarrow \text{horner}(n - m - 1, \texttt{c}, x)$;
        if $|px| < \tau$ **then**
            $\texttt{r}[m-1] = x$;
            Copiar los elementos del arreglo `b` al arreglo `c`;
            $res = True$;
            Terminar el ciclo;
        else
            $dpx, \texttt{b} \leftarrow \text{horner}(n - m, \texttt{b}, x)$;
            $x = x - px/dpx$;
        end
    end
    if $res == False$ **then**
        Hacer m igual a $m - 1$;
        Terminar el ciclo;
    end
end

Ejercicio 2. (5 puntos)
Además de usar polinomios para aproximar funciones se pueden usar otras funciones, por ejemplo, funciones trigonométricas. De hecho hay un resultado que dice que cualquier función periódica se puede aproximar por una combinación lineal de funciones trigonométricas, y es lo que queremos mostrar en este ejercicio.

Suponga que tenemos un conjunto de puntos $\{(x_1, y_1), ..., (x_N, y_N)\}$, donde $x_i \in [0, T]$, con T un entero positivo. Definimos $n = T - 1$ y $f_k = k/T$ para $k = 1, 2, ..., T - 1$. Se quiere ajustar a los datos la función

$$\begin{aligned} f(x) &= c_0 + c_1 \cos(2\pi f_1 x) + c_2 \sin(2\pi f_1 x) + c_3 \cos(2\pi f_2 x) + c_4 \sin(2\pi f_2 x) \\ &\quad + ... + c_{2n-1} \cos(2\pi f_n x) + c_{2n} \sin(2\pi f_n x) \end{aligned}$$

resolviendo el problema de mínimos cuadrados para encontrar los coeficientes $c_0, c_1, ..., c_{2n}$ que minimizar el error

$$E = \sum_{i=1}^{N} [y_i - f(x_i)]^2$$

1. Escriba el desarrollo para minimizar el error E para indicar cual es el sistema que se tiene que resolver para encontrar $c_0, c_1, ..., c_{2n}$.

2. Escriba la función que calcula la solución de mínimos cuadrados.

3. Escriba un programa que reciba desde la línea de comandos el nombre de un archivo que contiene una tabla de valores, donde cada fila es de la forma

   $$x_i \qquad y_i$$

4. Calcular $T = ceil(\max x_i)$ y defina $n_f = T - 1$. Defina las frecuencias $f_k = k/T$ para $k = 1, 2, ..., T - 1$ y calcule la solución de mínimos cuadrados.

5. Haga que el programa imprima el valor de E y los valores $c_0, c_1, ..., c_{2n}$.

6. Haga que el programa genere un archivo que contenga una matriz en la que cada fila es

   $$x_i \qquad y_i \qquad f(x_i)$$

   El propósito es usar esos datos para generar la gráfica de la función $f(x)$ y los puntos (x_i, y_i) para ver el ajuste de la función $f(x)$.

   Una observación. En este caso podemos usar las abscisas x_i para generar la gráfica porque en el conjunto de datos proporcionado las abscisas x_i están ordenadas de menor a mayor.

7. Use los datos del archivo `datosEjercicio2.zip` para probar el algoritmo, en el formato que guste, genere la gráfica correspondiente y agregue cada imagen al reporte.