Taller Parejas 1

Nicolás Duarte Ospina y Daniela Beltran Saavedra

Agosto 2018

1 Punto 1.b

El método de Horner lo desarrollamos en el lenguaje C++ en sus versiones 11 y posteriores, realizamos el análisis del método para traducirlo al lenguaje de nuestra preferencia teniendo en cuenta las reglas que este método tiene.

Desarrollado en C++ en sus versiones 11 y posteriores.

2 Punto 2

Para este caso nos apoyamos en el método de Newton y de Bisección para resolver el problema, además de generar una fórmula para la optimización del material.Para ambos casos lo resolvemos por cada método aparte.

Desarrollado en C++ en sus versiones 11 y posteriores.

(32-2*x)*(24-2*x)*x-1000;

1. La etapa mas importante es la comprensión del problema, saber que se esta solicitando y comprender los métodos matemáticos para su solución.

2. Conocimientos de programación y de aritmética, adicionalmente conocimientos de teoremas y modelos básicos matemáticos.

3. La desventaja mas grande es la perdida de tiempo y la exactitud del resultado obtenido.

4. El error de truncamiento es uno de los mas graves, desprecia gran cantidad de cifras lo cual genera una solución demasiado inestable.

5. Si el método matemático es aplicado en la máquina correcta y el lenguaje escogido por parte del programador es el adecuado se puede ahorrar mucho tiempo, también puede ser más exacta la solución, el ahorro de recursos es mayor y de dinero también en algunos casos.

6. La validación de resultados es la que nos da la certeza de que nuestro trabajo como programadores dio como resultado una solución correcta, y que el programa que generamos es de confianza.

3 Punto 4

Para dar una solución concreta de este problema aplicamos en C++ los cálculos de redondeo relativo y redondeo absoluto en este caso para un tamaño de capacidad n ingresado por el usuario.

Desarrollado en C++ en sus versiones 11 y posteriores.

$redondeo = 1 * (10^{n-m})$

$error = 10 * (10^{n-m})$

4 Punto 6

Usamos el método de divisiones aritméticas para hallar la eficiencia de un algoritmo denotado por T(n). En este proceso hacemos una división entera y un residuo entero de la división con el operador módulo (mod). la cantidad de divisiones aritmeticas T(n) es un entero (log2 n)+1 T(n), expresada en O() es de O(log2(n)).

Desarrollado en C++ en sus versiones 11 y posteriores.

$d = (algoritmomod2);$

$algoritmo = (algoritmo/2);$

5 Punto 7

Haciendo uso del método de Newton, resolveremos el problema propuesto este punto.

Desarrollado en math lab.

$R(t) = (2cos(t), sen(t), 0)$

$P = (2, 1, 0)$

Desarrollando el problema seria.

$(2cos(x) - 2)^2 + (sen(x) - 1)^2$

$4(cos(x))^2 - 8cos(x) + 4 + (sen(x))^2 - 2sen(x) + 1$

Simplificando y hallando las primera y segunda derivada obtenemos.

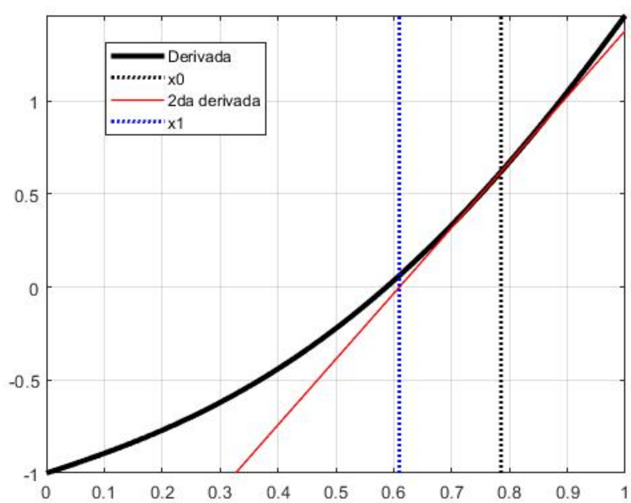
$f'(x) = -6sen(x)cos(x) + 8sen(x) - 2cos(x)$

$f''(x) = -6(cos(x))^2 + 6(sen(x))^2 - 8cos(x) + 2sen(x)$

Para las iteraciones usaremos x0 para empezar, hasta alcanzar un mínimo de tolerancia y tener una buena aproximación.

$x(n+1) = x(n)\frac{f'(Xn)}{f''(Xn)}$

Desde un punto de vista grafico los resultados arrojados serian.



6 Punto 11

El método de Muller, por medio de iteraciones para hallar los coeficientes, esto lo haremos con ayuda de las siguientes formulas y una serie de iteraciones para hallar los nuevos valores para las raíces con mayor exactitud.

Desarrollado en C++ en sus versiones 11 y posteriores.

$sigma0 = \frac{f(x1)-f(x2)}{x1-x0}$

$sigma1 = \frac{f(x2)-f(x1)}{x2-x1}$

$h0 = (x1 - x0)$

$h1 = (x2 - x1)$

Estas formulas nos ayudaran a encontrar los coeficientes a y b, de la siguiente formula, ya que la constante c se encuentra de f(x2)=c.

$(h0 + h1)b - (h0 + h1)^2 a = h0 * sigma0 + h1 * sigma1$

De acá despejamos a y b.

$a = \frac{sigma1 - sigma0}{h1 + h0}$

$b = a * h1 + sigma1$

Las constantes resultantes las podremos evaluar en la formula cuadrática, dando como resultado x3. Debido a que el error aun es muy grande, asignamos nuevos valores para x1,x2 y x3 y realizamos una nueva iteración, hasta que el error sea muy cercano a cero.

$error = \frac{x3 - x2}{x3} * 100$

7 Punto 14

Revisar archivos anexos en el repositorio “punto14.pdf”.

8 Punto 15

Revisar archivos anexos en el repositorio “punto15.pdf”.

Desarrollado en C++ en sus versiones 11 y posteriores.