Resumen

En este trabajo se utiliza el método de elemento finito para interpolación, y el método de Montecarlo para calcular integrales como introducción a este tema.

1. Introducción

Una desventaja de algunos métodos de interpolación, es que estos pasan siempre por los puntos que lo interpolan, sin embargo, algunas veces los puntos dados, pueden ser puntos atípicos, y generarían falsos patrones del conjunto. Así, en este trabajo se utiliza el método de elemento finito en una dimensión que depende fuertemente del rango donde están ubicados los puntos pero no necesariamente pasa por todos estos. Además, se utiliza el método de Montecarlo para calcular integrales en una y dos variables utilizando la idea de probabilidad geométrica, y la probabilidad de conteo.

2. Metodología

2.1. Elemento finito

El método de elemento finito, consiste en hacer particiones equiespaciadas en un intervalo [a, b], llamados elementos finitos. Dado un conjunto de m puntos $(x_i, y_i)_{i=0}^{m-1}$, se divide el intervalo en n subpuntos, de manera que el espaciado es igual a $h = (x[m-1] - x[0])/n$, y estos puntos se denotan como: $z_i = x[0] + ih$, donde $z[0] = x[0]$ y $z[m-1] = x[m-1]$. En estos subintervalos, se definirán funciones que sólo dependerán de sus vecinos, y en el subintervalo z_k y z_{k+1}, se tendrán elementos:

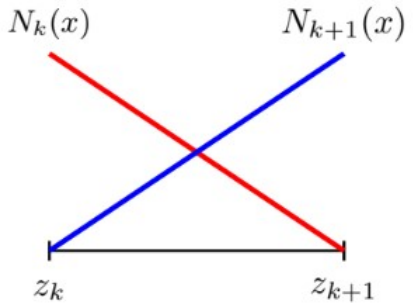

Figura 1: Nodos

$$N_k = \left[1 - \frac{1}{h}(x - z_k)\right], \tag{1}$$

$$N_{k+1} = \frac{1}{h}(x - z_k), \tag{2}$$

donde la ec. (1) y la ec.(2) se obtuvieron aplicando la ecuación de la recta, utilizando la Fig. 1. El objetivo de los métodos de interpolación, es minimizar el error entre los valores obtenidos y los reales, por ello, es que se busca minimizar el error sobre todos los puntos a interpolar, así, se definirán funciones de la forma:

$$\phi(x) = \sum_{j=0}^{n} \phi_j N_j(x), \tag{3}$$

$$\phi(x) = \phi_k N_k(x_i) + \phi_{k+1} N_{k+1}(x_i), \tag{4}$$

y se buscará minimizar:

$$\min_{\phi_0, \phi_1, \ldots, \phi_n} = \sum_{i=1}^{m} \left[\phi(x_i - y_i)\right]^2 + \lambda \int_a^b \left(\phi'(x)\right)^2 dx, \tag{5}$$

donde el parámetro λ es un valor que penaliza el error. Así, para cada valor de z, se tendrá una submatriz,

$$\begin{bmatrix} \sum_i N_k^2(x_i) + \frac{\lambda}{h} & \sum_i N_k(x_i)N_{k+1}(x_i) - \frac{\lambda}{h} \\ \sum_i N_k(x_i)N_{k+1}(x_i) - \frac{\lambda}{h} & \sum_i N_{k+1}(x_i)N_{k+1}(x_i) + \frac{\lambda}{h} \end{bmatrix}. \tag{6}$$

Así, se resolverá un sistema de ecuaciones para los elementos ϕ, los coeficientes de las rectas entre cada subintervalo, es decir:

$$\begin{bmatrix} \sum_i N_k^2(x_i) + \frac{\lambda}{h} & \sum_i N_k(x_i)N_{k+1}(x_i) - \frac{\lambda}{h} \\ \sum_i N_k(x_i)N_{k+1}(x_i) - \frac{\lambda}{h} & \sum_i N_{k+1}(x_i)N_{k+1}(x_i) + \frac{\lambda}{h} \end{bmatrix} \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix} = \begin{bmatrix} \sum_i y_i N_k(x_i) \\ \sum_i y_i N_k(x_{i+1}) \end{bmatrix} \tag{7}$$

Estas submatrices dependen de la cantidad de subintervalos en los que se resoverá el problema, un punto importante de este método es que no pasa por todos los puntos x como lo hacen la mayoría de los métodos de interpolación, este siempre pasará por el punto inicial y final del intervalo.

Algorithm 1 Elemento finito

Entrada: Puntos x, y; int cantPuntos; int cantIntervalos
Salida: Coeficientes entre subintervalos para las rectas

function NI(h, indice k, valor x, vector z)
    Esta función aplicará para $N_k(x_i)$ y $N_{k+1}(x_i)$, y se harán casos dependiendo del valor de x y de sus vecinos.
    if $k > 0$ **then**
        if $z[k] < x < z[k+1]$ **then**
            ni = $\frac{1}{h}(x - z[k-1])$
            return: ni
    if $z[k] < x < z[k+1]$ **then**
        ni = $1 - \frac{1}{h}(x - z[k])$
        return: ni
    return: 0

Crear matriz vacía.
suma00, suma01, suma11, s, rSuma.
Para la matriz A (del sistema Ax = b): suma00, suma01, suma11, s.
Para el vector b: rsuma.

for $k = 0; k < (cantIntervalos + 1); k++$ **do**
    Inicializar las variables suma en cero.
    Hay tres casos importantes, cuando es la primera fila, la última y los restantes.
    if $k == 0$ **then**
        for $i = 0; i < cantPuntos; i++$ **do**
            suma00 += $Ni(h, k, x[i], z)^2$
            suma01 += $Ni(h, k, x[i], z) * Ni(h, k+1, x[i], z)$
            rsuma += $Ni(h, k, x[i], z) * y[i]$
        Llenar la matriz.
    if $k == cantIntervalos$ **then**
        for $i = 0; i < cantPuntos; i++$ **do**
            suma11 += $Ni(h, k, x[i], z)^2$
            suma01 += $Ni(h, k, x[i], z) * Ni(h, k+1, x[i], z)$
            rsuma += $Ni(h, k, x[i], z) * y[i]$
        Llenar la matriz.
    Todos los demás casos.
    for $i = 0; i < cantPuntos; i++$ **do**
        suma00 += $Ni(h, k, x[i], z)^2$
        suma01 += $Ni(h, k, x[i], z) * Ni(h, k-1, x[i], z)$
        suma11 += $Ni(h, k, x[i], z)^2$
        rsuma += $Ni(h, k, x[i], z) * y[i]$
    Llenar la matriz
Resolver el sistema Ax = b
Se obtienen k coeficientes, y se hace una combinación lineal para cada intervalo. Para el intervalo [z[0], z[1]], se obtendrá una recta de la forma: $\phi_0 N_0(x_i) + \phi_1 N_1(x_i)$, para todos los x_i a interpolar.

2.2. Montecarlo

El método de Montecarlo parte de la idea de conteo de probabilidad geométrica. Se tiene un área y esta se llena de puntos, si los puntos caen dentro de la función a integrar, se cuentan, en otro caso no. Y de esta forma se encuentra una proporcion entre n puntos aleatorios dentro de un área y los puntos dentro de la función a integrar.

Algorithm 2 Montecarlo - una variable de integración

Entrada: Intervalo de integración a, b; cantidad de puntos (mientras más puntos, mejorará la respuesta).
Salida: Solución aproximada de la integral.

: Encontrar el máximo y mínimo de la función a integrar.
Calcular el área total: intervalo de x de integración por la diferencia entre el máximo y el mínimo: (x2-x1)(max-min).
inicializar conteo = 0,
for $i = 0; i < Puntos; i++$ **do**
    Generar punto aleatorio x: puntox.
    Generar punto aleatorio y: puntoy.
    Evaluar función en puntox: evalPuntox.
    if $puntoy <= evalPuntox$ **then** conteo +1;
Calcular la proporción de puntos con el área.
Si el punto está desplazado del origen, se debe de aumentar el valor de la integral.
valor integral = proporcion*area + min*(x2-x1).
return: valor integral.

En el caso de una integral de dos variables, nos interesa el volumen, por ello es que se seguirá el mismo algoritmo que el método con una variable, pero se hará una proporción con respecto a un volumen y habrá más variables aleatorias, punto x, punto y, punto z.

Algorithm 3 Montecarlo - dos variables de integración

Entrada: Intervalo de integración x: x2, x1; intervalo de integración en y: y2, y1; cantidad de puntos.
Salida: Solución aproximada de la integral.

Encontrar máximo y mínimo de la función(x,y).
inicializar conteo = 0
for $i = 0; i < Puntos; i++$ **do**
    Generar punto aleatorio x: puntox.
    Generar punto aleatorio y: puntoy.
    Evaluar la función: evalPunto.
    Generar punto aleatorio z: puntoz.
    if $puntoz <= evalPunto$ **then** conteo +1
Calcular el volumen: (x2-x1)(y2-y1)(max-min)
valor integral: proporción*volumen.

3. Resultados

3.1. Elemento finito

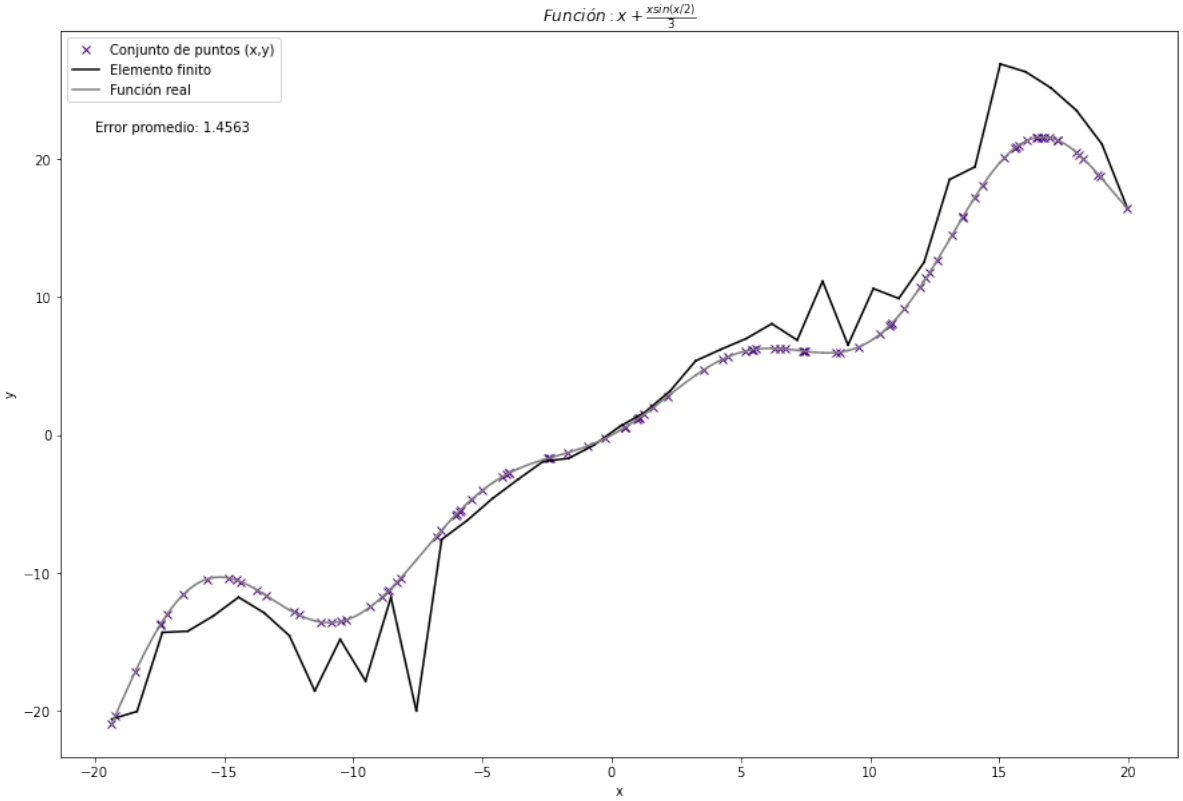

Figura 2: Interpolación utilizando 100 puntos y 40 intervalos.

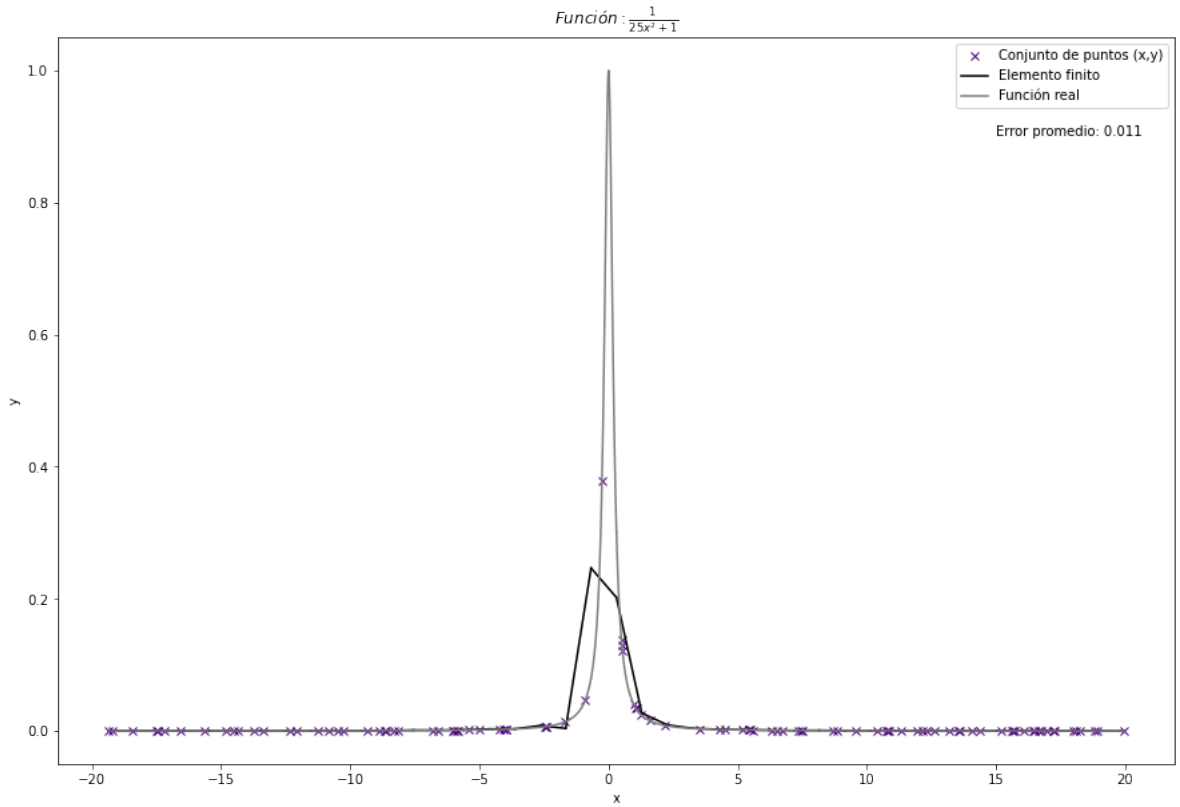

Figura 3: Interpolación utilizando 100 puntos y 40 intervalos.

Y con tiempos:

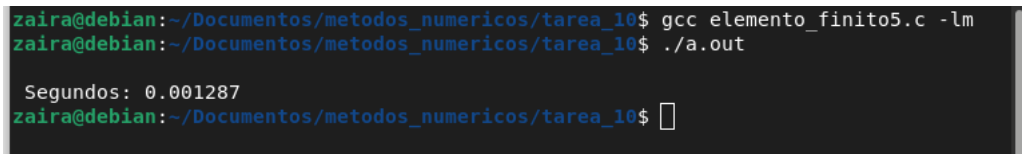

Figura 4: Tiempo de ejecución

3.2. Método de Montecarlo

Se utilizó la integral:

$$\int_0^3 x^2 + 2x + 5 = 33. \tag{8}$$

Y se obtuvo:

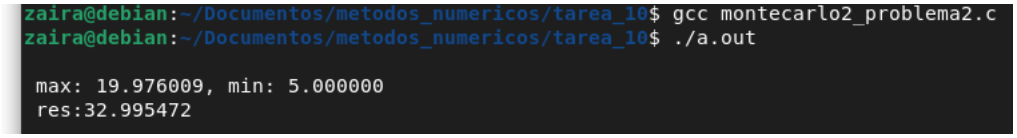

Figura 5: Resultados con montecarlo para una variable.

Para el caso de dos variables, se utilizó la integral:

$$\int_0^1 x^2 + y^2 = \frac{2}{3}. \tag{9}$$

Con resultados:

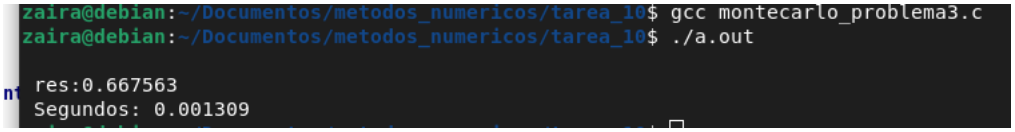

Figura 6: Resultados con montecarlo para dos variables.

4. Conclusiones

Como ya se mencionó el método de elemento finito tiene una ventaja grande sobre los métodos que pasan por todos los puntos, sin embargo, este método es más complicado de programar que los métodos de Splines, Lagrange o Newton, ya que el enfoque de la programación es con respecto a la cantidad de intervalos, no con la cantidad de puntos en este. Sin embargo, los resultados fueron muy buenos y competitivos con los otros métodos. En el caso del método de Montecarlo, mientrás más puntos se añadieran al área, el error numérico convergía a un número, pero para programar este para más variables, tendría que tener más variables aleatorias con el aumento.