Problemas de análisis numérico

Fabián Pallares
fpallares@javeriana.edu.co
GitHub: FabianPallaresJ

Santiago Jaramillo
l_jaramillo@javeriana.edu.co
GitHub: l_jara20

Agosto de 2019

1 Error de redondeo

Teniendo un dispositivo que puede almacenar unicamente los cuatro primeros digitos decimales de un número, y dado un número con determinados decimales mayores a 4, se da un error de redondeo.

Un error de rendondeo se da al usar una aproximación a un número en lugar de su valor real. Para determinar este error, decidimos realizar un algoritmo en que se íde al usuario el número a almacenar en el dispositivo. El algoritmo se encarga de determinar el número de digitos decimales del número ingresado y calcula el error de redondeo teniendo como base los números que siguen a los cuatro primeros ingresados.

Para mostrar la solución de este algoritmo, se ingresa al programa el número 536.78 y e obtiene el siguiente resultado:

```
Ingrese el numero a almacenar: 536.78
El error de redondeo corresponde a:
            E = 0.08
```

Figure 1: Error de redondeo para 536.78 almacenando 4 dígitos

2 Convergencia de raíz cuadrada

Se busca un algoritmo que permita calcular la raíz cuadrada de un número real desde un valor inicil y un valor de tolerancia (error) que servirá para determinar las iteraciones necesarias para llegar a un valor bastante aproximado.

Para nuestro ejemplo, queemos hallar la raíz cuadrada de 7. Usamos un valor inicial de 3 y un error de 0.00001.

Teniendo en cuenta que la solución a la raíz de un número es otro que, al multiplicarse por él mismo, se llega al valor dado inicialmente, podemos comprobar que nuestro resultado es coherente.

```
result = (0.5 * (vali + (dato / vali)))

while abs(vali - result) > error:
    vali = result
    result = (0.5 * (vali + (dato/vali)))
    print("resultado " + str(cont) + ": " + str(result))
    cont += 1
    print("El error es: " + str(abs(vali-result)))
```

Figure 2: Algoritmo iterativo para el cálculo de la raíz

Asimísmo, es posible determinar el valor dado por el programa en cada iteración y su error para cada una de ellas hasta el resultado que tomaremos como la aprximación más cercana al valor real.

En la siguiente imagen se ve representado el ejercicio enunciado en el programa:

```
Ingrese el dato a usar: 7
Error permitido: 0.00001
Valor inicial: 3
resultado 0: 2.645833333333333
El error es: 0.020833333333333925
resultado 1: 2.6457513123359577
El error es: 8.202099737530943e-05
resultado 2: 2.6457513110645907
El error es: 1.2713670116681897e-09

Comprobación: 2.6457513110645907 ** 2 = 7.000000000000001
El error es: 1.2713670116681897e-09
```

Figure 3: Raíz con error, comprobación, y error por cada iteración

3 Teorema de Taylor

Gracias al teorema de Taylor podemos obtener aproximaciones polinómicas de funciones gracias a sus derivadas.

Del mismo modo, gracias a las iteraciones producidas por la sumatoria del teorema, se permite reducir el error por cada una de ellas.

Para este ejercicio pretendemos hallar una aproximación con una función dada:

$$f(x) = e^{0.5}$$

Para este ejercicio usamos un grado 8. Lo que nos da 9 aproximaciones (Contando la de grado 0). Los resultados se ven representados a continuación:

```
Polinomio de taylor para e^0.5
A qué grado desea realizar el polinomio? 8
Resultado real: 1.6487
0 Aproximación: 1.0
1 Aproximación: 1.5
2 Aproximación: 1.625
3 Aproximación: 1.6458
4 Aproximación: 1.6484
5 Aproximación: 1.6487
6 Aproximación: 1.6487
7 Aproximación: 1.6487
8 Aproximación: 1.6487
Error en la aproximación de grado 8: 2.1265e-05
```

Figure 4: Teorema de Taylor evaluado grado 8

4 Tamaño del error en operaciones aritméticas

Para este ejercicio se pretende hallar el error en operaciones aritméticas. En este caso hallaremos una distancia con su respectivo error absoluto y realitivo, basados en los errores proporcionados por la velocidad y el tiempo del recorrido.

Para hallar la distancia simplemente se multiplica la velocidad del recorrido por el timepo recorrido, y se consiguen los errores con las fórmulas expresadas en el siguiente algoritmo:

```
distancia = vel * tiempo
error_abs = (vel * ev) + (tiempo * et)
error_rel = ((ev / vel) + (et / tiempo)) * 100
print("\n\nLa distancia recorrida es de: " + str(distancia))
print("Con un error absoluto de: " + str(error_abs))
print("Por lo cual la distancia tiene un rango de variación: ")
print("   " + str(distancia - error_abs) + " <= d => " + str(distancia + error_abs))
print("\nEl error relativo es de: " + str(error_rel) + "%")
```

Figure 5: Calculo de distancia, errores, e intervalo

Para este ejericio usaremos estos valores (dados por el usuario):

- Velocidad: 4.
- Tiempo: 5.
- Error de ambas magnitudes: 0.1

Con lo cual obtenemos los siguientes resultados:

```
Ingrese la velocidad del recorrido: 4
Error de la velocidad: 0.1
Ingrese el tiempo del recorrido: 5
Error del tiempo: 0.1


La distancia recorrida es de: 20.0
Con un error absoluto de: 0.9
Por lo cual la distancia tiene un rango de variación:
   19.1 <= d => 20.9

El error relativo es de: 4.5%
```

Figure 6: Resultados de distancia recorrida, errores absoluto, relativo, e intervalo de variación

5 Evaluación de un polinomio

El algoritmo de Horner es bastante útil a la hora de evaluar funciones polinómicas, ya que mediante este podemos calcular tanto la solución aproximada de este como el número de operaciones a realizar para llegar al resultado.

Para este ejercicio, evaluaremos la siguiente función:

$$f(x) = 2x^4 + 3x^2 - 3x - 4$$

Y usaremos un valor de x inicial de -2. Los resultados son presentados a continuación:

```
Ingrese un valor para x: -2
Ingrese valor en a: 2
Ingrese valor en a: 0
Ingrese valor en a: -3
Ingrese valor en a: 3
Ingrese valor en a: -4


El resultado de la evaluación del polinomio es: 10.
Con un minimo de operaciones de 10
```

Figure 7: Resultados del polinomio a evaluar con x0 = -2