Instituto Politécnico Nacional

Escuela Superior de Cómputo

Tarea #2. Sesión 3 del parcial 2 de la clase de Métodos Numéricos.

Extrapolación de Richardson.

Nombre: De Luna Ocampo Yanina

Fecha de clase: 29/09/2021

Introducción: Este método nos permite determinar mejores aproximaciones a partir de aproximaciones de un orden no tan bueno. Lo podemos aplicar siempre que sepamos que el método de aproximación tiene un término de error con una forma previsible, la cual se basa en el tamaño de paso de la aproximación.

La extrapolación de Richardson cuenta con la ventaja de servir como una forma alternativa de optimizar ciertos procesos futuros con datos ya recolectados. Permite tomar medidas preventivas, se puede usar para buscar la solución de un problema o enseñar a corregir un error cometido con anterioridad. Sin embargo, como en todos los métodos, no todo puede ser perfecto, tiene desventajas, una de ellas es que la mayoría del tiempo usa datos de situaciones que están ya interviniendo, por lo que a veces lo datos obtenidos son útiles por corto tiempo.

Descripción: Aplique el método de extrapolación de Richardson para obtener una aproximación del tipo $N_4(h)$ con las siguientes funciones, tamaños de h y punto iniciales.

Procedimiento del ejercicio 3,4, número 1):

Lo primero que tenemos que hacer es definir nuestro punto de aproximación y de paso como nos fue indicado en el enunciado del ejercicio, que son los siguientes:

```
In [3]: # Definimos tambien los tamaños de pasos iniciales, el punto de aproximación:
        x0 = 1.00000000 # Punto de aproximación
        h = 0.40000000 # Tamaño de paso
```

Declaramos nuestra función, que es: $f(x) = \ln(x)$

```
def fx(x):
    fx = log(x)
    return fx
```

En esta ocasión nuestro orden va a ser de 4, definimos n=4

```
n = 4
```

Resultado:

Procedemos a imprimir nuestras aproximaciones, obtendremos 4 aproximaciones porque pasamos el valor de n al for y obtenemos lo siguiente:

```
In [6]: [central2(fx, h / (2**i), x0) for i in range(4)]
Out[6]: [1.0591223254840045, 1.0136627702704106, 1.003353477310756, 1.000834585569826]
```

Ahora tendremos una lista de listas, en donde mi primer elemento son las diferencias iniciales, obtenemos la primera, las subsecuentes y la última.

```
In [7]: richard
Out[7]: [[1.05912233, 1.01366277, 1.00335348, 1.00083459],
         [0.99850958, 0.99991705, 0.99999496],
         [1.00001088, 1.00000015],
         [0.99999998]]
```

Procedimiento del ejercicio 3,4, número 2):

Lo primero que tenemos que hacer es definir nuestro punto de aproximación y de paso como nos fue indicado en el enunciado del ejercicio, que son los siguientes:

```
In [3]: # Definimos tambien los tamaños de pasos iniciales, el punto de aproximación:
        x0 = 0.00000000 # Punto de aproximación
        h = 0.40000000 # Tamaño de paso
```

Declaramos nuestra función, que es: $f(x) = \mathrm{x} + \mathrm{e}^{\mathrm{x}}$

```
def fx(x):
    fx = x + e**(x)
    return fx
```

En esta ocasión nuestro orden va a ser de 4, definimos n=4

```
n = 4
```

Resultado:

Procedemos a imprimir nuestras aproximaciones, obtendremos 4 aproximaciones porque pasamos el valor de n al for y obtenemos lo siguiente:

```
In [6]: [central2(fx, h / (2**i), x0) for i in range(4)]
Out[6]: [2.026880814507418, 2.006680012705828, 2.001667500198793, 2.0004167187534536]
```

Ahora tendremos una lista de listas, en donde mi primer elemento son las diferencias iniciales, obtenemos la primera, las subsecuentes y la última.

```
In [7]: richard
Out[7]: [[2.02688081, 2.00668001, 2.0016675, 2.00041672],
         [1.99994641, 1.99999666, 1.99999979],
         [2.00000001, 2.0],
         [2.0]]
```

¿Qué aprendí?

Aprendí un nuevo método que como se mencionó, sirve para generar resultados de gran exactitud cuando se usan fórmulas de bajo orden. Es posible aplicarlo en cualquier ámbito de trabajo siempre y cuando cumpla con las condiciones previamente establecidas.