Instituto Politécnico Nacional

Escuela Superior de Cómputo

Examen 2 del parcial 2 de Métodos Numéricos.

Nombre del alumno: De Luna Ocampo Yanina

Fecha de entrega: 10/11/2021

Ejercicio1, cuadraturas:

Aplica todas las cuadraturas de Newton – Cotes abiertas y cerradas para determinar una aproximación de la siguiente integral, asimismo compare la aproximación con el valor real y presente sus resultados en una tabla:

$$\int_{e}^{e+1} \frac{1}{xlog(x)} dx$$

Procedimiento:

Con n = 1

Puntos	Cerrada	Error	Abierta	Error
1	0.28633417	0.01382029	0.26583859	0.00667529
2	0.27267045	0.00015657	0.26800187	0.00451201
3	0.27258495	0.00007107	0.27237973	0.00013416
4	0.2725154	0.00000152	0.27241971	0.00009417

Tomamos en cuenta nuestro valor real: 0.27251388050277886

Con 8 decimales: 0.27251388

Explicación:

Lo primero que hice fue importar las funciones y las variables necesarias para poder declarar mi función correctamente. Una vez hecho esto, procedemos ahora sí a declarar nuestra integral dada anteriormente, declarando asimismo sus extremos y el número de puntos por el cual aplicaremos la cuadratura.

Obteniendo como resultado lo visto en la tabla superior:

round(cerrada1(fx,a,b),8)	round(abierta1(fx,a,b),8)
0.27267045	0.26583859
round(cerrada2(fx,a,b),8)	round(abierta2(fx,a,b),8)
0.27267045	0.26800187
round(cerrada3(fx,a,b),8)	round(abierta3(fx,a,b),8)
0.27258495	0.27237973
round(cerrada4(fx,a,b),8)	round(abierta4(fx,a,b),8)
0.2725154	0.27241971

Análisis de las respuestas obtenidas:

Observamos que cuando son cerradas, conforme el valor de n aumenta, el error comienza a disminuir, a diferencia del abierto. Requerimos un número muy grande de evaluaciones de la función, lo que hace que sea más trabajo de cálculo para alcanzar los niveles de precisión.

Nuestro valor real se mantiene muy acercado a lo que obtuvimos de nuestra función inicial, es por casi nada un poco diferente a lo obtenido en las demás.

Ejercicio2, compuesta:

Aplique los métodos de integración compuesta del trapecio y de Simpson con n = 10 para aproximar el valor de la integral siguiente:

$$\int_{1}^{11} x^2 \, log(x) \, dx$$

Se deberán comprar los resultados con el valor real de la integral y además especificar las integrales que están considerando.

Procedimiento:

Lo primero que hice fue importar las funciones y las variables necesarias para poder declarar mi función correctamente. Una vez hecho esto, procedemos ahora sí a declarar nuestra integral dada anteriormente, declarando asimismo sus extremos y el número de subdivisiones

Procedemos a definir nuestra regla del trapecio compuesto y la de Simpson. Para ambos debemos encontrar h con la fórmula que ya conocemos “h = (b – a) / n”. Creamos nuestro arreglo de puntos y lo evaluamos, procedemos a aplicar la cuadratura. Tenemos:

Regla del trapecio compuesta

$$\int_a^b f(x)dx \approx \frac{h}{2}\left(f(x_0) + 2\sum_{j=1}^{n-1} f(x_j) + f(x_n) \right)$$

Regla del trapecio compuesta en código

```
trapecioC = (h / 2) * (eval[0] + 2 * eval[1:n].sum() + eval[n])
```

Regla de Simpson

$$\int_a^b f(x)dx \approx \frac{h}{3}\left(f(x_0) + 2\sum_{j=1}^{\frac{n}{2}-1} f(x_{2j}) + 4\sum_{j=1}^{\frac{n}{2}} f(x_{2j-1}) + f(x_n) \right)$$

Regla de Simpson en código

```
simpsonC = (h / 3) * (eval[0] + 2 * eval[::2][1:int((n/2))].sum() + 4 * eval[1::2].sum() + eval[n])
```

Se pide comparar con el valor real obtenido de la integral que es equivalente a:

```
El valor real es: 0.8565204445899679
```

Las integrales que se están considerando son las siguientes:

$$\int_1^{11} x^2 \log(x)\, dx = \int_1^2 x^2 \log(x)\, dx + \int_2^3 x^2 \log(x)\, dx + \int_3^4 x^2 \log(x)\, dx$$

$$+ \int_4^5 x^2 \log(x)\, dx + + \int_5^6 x^2 \log(x)\, dx + + \int_6^7 x^2 \log(x)\, dx + + \int_7^8 x^2 \log(x)\, dx$$

$$+\int_{8}^{9} x^2 \log(x)\,dx + +\int_{9}^{10} x^2 \log(x)\,dx + +\int_{10}^{11} x^2 \log(x)\,dx$$

Explicación:

```
trapecioC(fx,a,b,10)
```

```
921.3203178403933
```

```
simpsonC(fx,a,b,10)
```

```
916.0798167997834
```

Análisis de las respuestas obtenidas:

Vemos que, con el método de trapecio compuesto, su aproximación tiende a converger mucho más, con esto me refiero a que es más exacto al momento de mostrarnos resultados, sin embargo, necesitamos de un valor en n lo suficientemente alto para que tengamos una aproximación que se considere aceptable.

Ahora con Simpson, hay que tomar en cuenta que si el límite inferior (a) y el límite superior (b) no son suficientemente pequeños el grado de error, podría ser grande.

El error de aproximación de uno a otro porque el dividir el intervalo en subintervalos de longitud más pequeña nos beneficia para obtener mejores aproximaciones.

Ejercicio3, Richardson:

Aplique el método de interpolación de Richardson para obtener una aproximación de orden $O(h^{10})$ con un tamaño de $h = 0.5$ para la función $f(x) = 2^x sin(x)$, en el punto $x = 2.00$ asimismo, compare contra el valor exacto de la derivada y presente los resultados previamente obtenidos.

Procedimiento:

Lo primero que hice fue importar las funciones y las variables necesarias para poder declarar mi función correctamente. Una vez hecho esto, procedemos ahora sí a declarar nuestra función dada anteriormente, declarando x0 que representa el punto de aproximación y h que es nuestro tamaño de paso.

Como nuestro orden esta representado por $O(h^{10})$ declaramos nuestra n como 5,determinamos las diferencias iniciales de referencia, calculamos las siguientes aproximaciones, las agregamos y aplicamos la fórmula de este método.

Explicación:

Lo explicado con anterioridad podemos verlo representado de la siguiente manera:

```
aprox = richard[j - 1][i + 1] + (1 / (4**j - 1)) * (richard[j - 1][i + 1] - richard[j - 1][i])
richard[j].append(round(aprox,8))
```

$0(h^2)$

```
[[0.56412781, 0.78287941, 0.83807659, 0.85190739, 0.85536705],
```

$0(h^4)$

```
[0.85579661, 0.85647565, 0.85651766, 0.85652027],
```

$0(h^6)$

```
[0.85652092, 0.85652046, 0.85652044],
```

$0(h^8)$

```
[0.85652045, 0.85652044],
```

$0(h^{10})$

```
[0.85652044]]
```

En comparación con la derivada:

$$\frac{d}{dx}\left(2^x \sin(x)\right) = 2^x\left(\cos(x) + \log(2)\sin(x)\right)$$

Análisis de las respuestas obtenidas:

Obtuvimos una lista de listas en donde mi primer elemento son las diferencias iniciales, obtenemos la primera, las subsecuentes y la última. Vemos que, a pesar de no ser un orden esperado, logra determinar buenas aproximaciones. Nuestro término de error se basa en el tamaño de paso de la aproximación.

Ejercicio4:

Aplique el método de integración Gaussiana múltiple con n = 3 en la variable x y m = 3 en la variable y, asimismo, comprare contra el valor real de la integral:

$$\int_0^{\pi/4} \int_{\sin(x)}^{\cos(x)} (2y\sin(x)\cos^2(x))dydx$$

def cambio_variable(u,v): return (((fxy(((b - a) * u + b + a) / 2,((d(((b - a) * u + b + a) / 2) - c(((b - a) * u + b + a) / 2)) * v + d(((b - a) * u + b + a) / 2) + c(((b - a) * u + b + a) / 2)) / 2)) * (d(((b - a) * u + b + a) / 2) - c(((b - a) * u + b + a) / 2)) * (b - a)) / 4)

$$\int_0^{\left(\frac{\pi}{4}\right)} \int_{(\sin(x))}^{(\cos(x))} \left(2y\sin(x)\cos^2(x)\right) dydx = \frac{1}{15\sqrt{2}} + \frac{1}{15} \quad (\text{Decimal: } 0.11380\ldots)$$

Ejercicio5, Euler, Runge – Kutta:

Considere el siguiente problema de valores iniciales:

$$y' = e^{t-y}, \qquad 0 \le t \le 1, \qquad y(0) = 1$$

La cual tiene como solución a la función:

$$y(t) = log(e^t + e - 1)$$

Dado lo anterior:

a) Aproxime la solución por medio del método de Euler con $h = 0.05$ para aproximar la solución.
b) Aproxime la solución por medio del método de Runge – Kutta de orden 4 con $h = 0.05$ para aproximar la solución.

Finalmente, compare los resultados obtenidos y determine los errores de aproximación con respecto a la solución real.

Procedimiento:

a) Lo primero que hice fue importar las librerías y las variables necesarias para poder declarar mi función correctamente. Una vez hecho esto, procedemos ahora sí a

declarar nuestra función dada anteriormente, declarando el punto inicial, el punto final, condición inicial y el número de pasos que en este caso debe ser 20 para poder cumplir la condición que piden de h.

Una vez hecho esto, podemos declarar nuestra fórmula de h, generamos el arreglo de puntos y aproximaciones donde trabajaremos.

Definimos nuestra solución que esta representada por:

```
def ftyR(t):
    ftyR = log(e**(t) + e - 1)
    return ftyR
```

Una vez hecho todo eso, obtenemos los siguientes resultados:

b) Lo primero que hice fue importar las librerías y las variables necesarias para poder declarar mi función correctamente. Una vez hecho esto, procedemos ahora sí a declarar nuestra función dada anteriormente, declarando el punto inicial, el punto final, condición inicial y el número de pasos que en este caso debe ser 20 para poder cumplir la condición que piden de h.

Una vez hecho esto, podemos declarar nuestra fórmula de h, generamos el arreglo de puntos y aproximaciones donde trabajaremos.

Es momento de definir nuestra función de Runge – Kutta de orden 4, dado por:

```
def rungeK4(t, w, h):
    k1 = h * fty(t, w)
    k2 = h * fty(t + h / 2, w + k1 / 2)
    k3 = h * fty(t + h / 2, w + k2 / 2)
    k4 = h * fty(t + h, w + k3)
    rungeK4 = k1 + 2 * k2 + 2 * k3 + k4
    return rungeK4
```

Definimos nuestra solución que esta representada por:

```
def ftyR(t):
    ftyR = log(e**(t) + e -1)
    return ftyR
```

Determinamos las aproximaciones, creamos nuestro arreglo, asignamos los puntos donde trabajaremos y los valores aproximados.

Asignamos los valores reales y determinamos el error de aproximación. Imprimimos nuestros resultados que podremos visualizar en la sección de abajo en el inciso b).

Explicación con código:

a) Euler

```
El valor aproximado de la función en el punto 0.0 es 1.0 el cual tiene un error de 0.0
El valor aproximado de la función en el punto 0.05 es 1.01839397 el cual tiene un error de 0.00029194
El valor aproximado de la función en el punto 0.1 es 1.03737859 el cual tiene un error de 0.00058192
El valor aproximado de la función en el punto 0.15 es 1.05696125 el cual tiene un error de 0.00086935
El valor aproximado de la función en el punto 0.2 es 1.07714871 el cual tiene un error de 0.00115358
El valor aproximado de la función en el punto 0.25 es 1.09794707 el cual tiene un error de 0.001434
El valor aproximado de la función en el punto 0.3 es 1.11936174 el cual tiene un error de 0.00170998
El valor aproximado de la función en el punto 0.35 es 1.14139738 el cual tiene un error de 0.00198091
El valor aproximado de la función en el punto 0.4 es 1.16405793 el cual tiene un error de 0.00224619
El valor aproximado de la función en el punto 0.45 es 1.18734656 el cual tiene un error de 0.00250522
El valor aproximado de la función en el punto 0.5 es 1.21126564 el cual tiene un error de 0.00275742
El valor aproximado de la función en el punto 0.55 es 1.23581676 el cual tiene un error de 0.00300226
El valor aproximado de la función en el punto 0.6 es 1.26100069 el cual tiene un error de 0.0032392
El valor aproximado de la función en el punto 0.65 es 1.28681741 el cual tiene un error de 0.00346776
El valor aproximado de la función en el punto 0.7 es 1.31326607 el cual tiene un error de 0.00368747
El valor aproximado de la función en el punto 0.75 es 1.34034503 el cual tiene un error de 0.0038979
El valor aproximado de la función en el punto 0.8 es 1.36805183 el cual tiene un error de 0.00409866
El valor aproximado de la función en el punto 0.85 es 1.39638324 el cual tiene un error de 0.00428942
El valor aproximado de la función en el punto 0.9 es 1.42533526 el cual tiene un error de 0.00446987
El valor aproximado de la función en el punto 0.95 es 1.45490311 el cual tiene un error de 0.00463973
El valor aproximado de la función en el punto 1.0 es 1.48508131 el cual tiene un error de 0.00479881
```

b) Runge – Kutta

La aproximacion obtenida se encuentra dada por:

Punto	Aproximacion	Real	Error Absoluto
0.0	1.0	1.0	0.0
0.05	1.01868591	1.01868591	0.0
0.1	1.03796051	1.03796051	0.0
0.15	1.0578306	1.0578306	0.0
0.2	1.07830229	1.07830229	0.0
0.25	1.09938108	1.09938107	0.0
0.3	1.12107172	1.12107172	0.0
0.35	1.1433783	1.1433783	0.0
0.4	1.16630412	1.16630412	0.0
0.45	1.18985178	1.18985177	0.0
0.5	1.21402306	1.21402306	0.0
0.55	1.23881902	1.23881902	0.0
0.6	1.26423989	1.26423989	0.0
0.65	1.29028517	1.29028517	0.0
0.7	1.31695354	1.31695354	0.0
0.75	1.34424293	1.34424292	0.0
0.8	1.3721505	1.3721505	0.0
0.85	1.40067267	1.40067267	0.0
0.9	1.42980513	1.42980512	0.0
0.95	1.45954285	1.45954285	0.0
1.0	1.48988013	1.48988013	0.0

<u>Análisis de las respuestas obtenidas:</u>

Para el Runge – Kutta es un método que hace que el error absoluto sea $0(h^4)$. Podemos observar que este método es mejor que el de Euler, pero aún así podemos aumentar la precisión disminuyendo los pasos entre los puntos o implementando en este método el de orden superior.