Actividad 2 (Codigos Area-Volumen y Raices)

Gustavo Alberto Medina Ferrer[1]

A219223438[1]

Numero(s):(81) 8309 6131[1]

gustavomedinaferrer@gmail.com[1]

Universidad de Sonora

Licenciatura en Fisica

Fisica Computacional

19/01/2021

Resumen

En este documento se hara una recapitulizacion sobre la actividad 2 de la materia de Fisica Copmutacional, impartida en la Universidad de Sonora por el profesor C. Lizarraga.

1. A C T I V I D A D

1.1. Ejercicio 1:

INSTRUCCIONES

Se nos proporciono con un codigo de un programa en lenguaje Python en la pagina web de la materia, el cual debiamos de introducir en un notebook de *Google Colab*. Despues de introducirlo, como estabamos en clase, el profesor nos guio un poco sobre las cosas basicas de Python, para que asi nosotros pudieramos hacer la actividad despues. El profesor edito algunas cosas del codigo original para mostrarnos diferentes funciones y demas.

Tras esto se nos pidio modificar el codigo original (que servia para calcular el area de un rectangulo) para:

- Calcular el area de un circulo.
- Calcular el area de una elipse.
- Calcular el volumen de una esfera.
- calcular el volumen de un cilindro circular.

EJERCICIO

Para hacerlo, separe cada una de las "sub-actividadesçon un renglon de texto, poniendo el fin de cada codigo como una seccion y en codigo de Latex.

Como las instrucciones nos pedian escribir la formula que usariamos como algoritmo en Latex antes de hacer cualquier calculo, debajo de cada seccion escribi la formula que describia cada codigo.

Con estas actividades no fue mayor problema, tal vez el haber olvidado algunas formulas y buscarlas, pero nada mayor. Aun asi, fue bastante util para familiarizarme con Python, ya que yo solo habia programado en FORTRAN. Tampoco fue algo muy dificil, pero me alegra poder haberlo hecho (en su mayoria) solo.

1.2. Ejercicio 2:

INSTRUCCIONES

Para este segundo ejercicio, el cual fue el que, personalmente, genero un reto, lo que debiamos hacer es escribir un programa que nos calculara las raices de una funcion cuadratica. Para esto, se baso en la *formula general*:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{1}$$

EJERCICIO

EL uso de la formula general fue lo que causo mas problema. Al principio se queria hacer un programa que solo calculara las raices **reales**, porque parecia lo mas simple. Sin embargo, resulto ser lo opuesto.

El primer problema salio cuando Python, por alguna razon extrana que desconozco, no tenia la funcion implicita de "sqrt(x)", que es la forma en la que deberia de escribirse en Python 3, pero nunca lo reconocia. Esto llevo a, inicialmente declarar una funcion "sqrt(x)"para poder sacar la raiz de la formula general. No hubo mayor problema en eso.

Una vez hechos los codigos y despues de haberlos probado, me percate de que tambien generaba raices imaginarias. Mi siguiente idea fue encontrar una manera de descartar las raices imaginarias con un IF, sin embargo eso resulto mas batalloso de lo que pense.

Por ultimo, importe una libreria de matematica compleja, la cual tenia la funcion de çmath.sqrt", la que calculaba la raiz cuadrada de un argumento y la expresaba en forma compleja, por lo que ahora podia expresar raices complejas y simples de manera facil, que fue lo que se termino haciendo.

1.3. Ejercicio 3:

INSTRUCCIONES

En este tercer ejercicio se nos pidio encontrar una forma de implementar el **Metodo Babiliano (o de Heron)**, en un codigo para aproximar la raiz cuadrada de un numero, al igual que comparar el resultado del programa con el de la funcion de la biblioteca Numpy, np.sqrt(x).

Como contexto, el Metodo Babiliano sirve para aproximar la raiz de un numero de una manera muy intuitiva. La expresion del metodo es la siguiente:

$$x_{n+1} = \frac{1}{2}(x_n + \frac{S}{x_n}) \tag{2}$$

Donde S es el numero al cual queremos aproximar su raiz y x_n es un numero arbitrario. Como podemos ver, la expresion anterior describe la media entre los numeros x_n y $\frac{S}{x_n}$. El metodo consiste en repetir el procedimiento muchas veces, hasta que el valor comience a converger a algun numero.

EJERCICIO

Para la realizacion del codigo, primero se intento comprender bien el algoritmo del metodo, lo cual tomo un poco mas de lo que esperaria. Tras esto, se realizaron varias veces calculos del metodo en una calculadora, comparando el resultado con el resultado de la funcion raiz cuadrada de la misma, para confirmar que se estaba escribiendo bien el algoritmo. Ahora ya se estaba listo para continuar con el codigo.

Se comenzo llamando a las bibliotecas que se necesitaron, la cual solo fue *Numpy*. Despues, se comenzo por explicar al usuario el uso del programa, seguido de preguntarle el numero al cual querra aproximar su raiz cuadrada (S), seguido de leerlo y tambien declarar una variable que solo guardaria la raiz del numero introducido, calculado por *Numpy*, el cual nos ayudara a encontrar el error entre el valor calculado por el metodo y el real".

Seguido de esto se realizo el algoritmo del metodo babiliano, usando un loop while, el cual compararia el valor absoluto entre S y la aproximacion, y mientras esta diferencia no sea menor a 0.01, no pararia de repetirse.

Por si nunca se llegara a alcanzar el 0.01 de diferencia, tambien se implemento un contador y un if, que alcanzados las 10,000 iteraciones, pararia el loop.

Por ultimo, se imprimieron los resultados, al igual que la diferencia entre el aproximado y el real".

1.4. Ejercicio 4:

INSTRUCCIONES

En este ultimo ejercicio, se nos pidio graficar una figura que se encontraba en un articulo de una pagina de *Wikipedia*. El articulo no incluye mucho sobre su realizacion ya que solo es usada como complemento para el tema de *Series de Taylor*, sin embargo, se podian apreciar los grados del polinomio de Taylor con los cuales se genero la grafica.

Cabe mencionar que a lo que aproximaban los polinomios es a la funcion $ln(x+1)$, la cual puede ser descrita en la siguiente serie:

$$ln(x+1) = x - \frac{x^2}{2} + \frac{x^3}{3} + \cdots \tag{3}$$

EJERCICIO

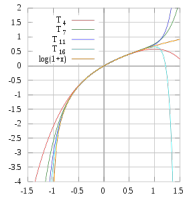


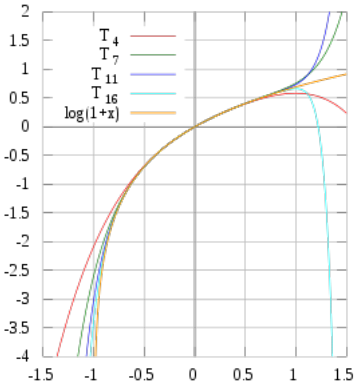
Figura 1: Grafica mostrada en el articulo de Wikipedia

Se comenzo por importar las librerias necesarias. En este caso *Numpy* y *Matplotlib*. Seguido, se pidio al usuario el grado del primer polinomio que desea calcular (n)m seguido de un if que terminaria el programa si el numero no fuera positivo.

Despues de esto, se declaro un arreglo de 100 puntos entre -1.5 y 1.5 (que es el dominio de la grafica mostrada), al igual que un valor auxiliar que seria de ayuda en el loop del algoritmo.

Lo siguiente fue hacer el algoritmo del metodo, el cual no quedo tan optimizado como se quisiera, sin embargo, a falta de conocimientos de Python, se opto por repetir el algoritmo 4 veces (el numero de polinomios en la figura 2). De no ser asi, se planeaba hacer un codigo que dejara la usuario aproximar el numero de polinomios que el quisiera (hasta un limite), y asi graficar todos con otro loop.

Tras hacer el algoritmo, simplemente se imprimio el resultado como una grafica, la cual coincide totalmente con la del articulo:



2. CONCLUSION

En general, esta resulto ser una actividad muy grata, ya que aprendi mucho mas de Python de lo que ya conocia, y siento que comienzo a familiarizarme con el tema. Por otro lado, estuvo muy bien balanceada entre tener que codificar y tener que entender conceptos matematicos, lo cual me parecio bastante bien.

3. REFERENCIAS

1. Articulo de Wikipedia sobre la Serie de Taylor mencionado: https://en.wikipedia.org/wiki/Taylor_series
2. Jan Bodnar. (2007). Python functions - . 21/01/2021, de ZETCODE Sitio web: https://zetcode.com/lang/py
3. Michele Pratusevich. (2014). 01 Character Input. 22/01/2021, de PRACTICE PYTHON Sitio web: https://www.practicepython.org/exercise/2014/01/29/01-character-input.html
4. Username: Diziet Asahi. (2016). ln (Natural Log) in Python. 21/01/2021, de STACKOVER-FLOW Sitio web: https://stackoverflow.com/questions/39235906/ln-natural-log-in-python/39235942
5. Username: naina024. (SF). Python exit commands: quit(), exit(), sys.exit() and os._exit(). 21/01/2021, de GEEKSFORGEEKS Sitio web: https://www.geeksforgeeks.org/python-exit-commands-quit-exit-sys-exit-and-os-_exit/

■