Universidad de Guadalajara

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍA



*Proyecto final*

Fundamentos Filosóficos de la Computación

Sección D07

Conrado Cruz Gómez

Edgar Manuel Cabral Serpa- 217552864

Ismael Iván López Murillo – 220975903

María José Rangel Salmerón – 212557817

27 de noviembre de 2020.

Ingeniería en Computación

**Descripción del proyecto.**

El proyecto que realizaremos es un solucionador de problemas con métodos numéricos, los temas propuestos son:

* Ecuaciones diferenciales ordinarias.
* Cálculo integral.
* Interpolación
* Ajuste de curva.

Para cada tipo de problema existirá la opción de cargar un ejemplo previamente definido o cargar información propia mediante entrada en consola.

Las opciones (por ahora) estarán definidas como:

* Bienvenida.
  + Mostrar Integrantes.
  + Mostrar datos del programa.
* Ejemplos.
  + Ejemplo ecuaciones diferenciales (información cargada de archivo).
  + Ejemplo cálculo integral (información cargada de archivo).
  + Ejemplo interpolación (información cargada de archivo).
  + Ejemplo de ajuste de curva (información cargada de archivo.)
* Solucionador:
  + Ecuaciones diferenciales ordinarias.
  + Cálculo integral y diferencial.
  + Interpolación y ajuste de curva.
  + Ajuste de curva
* Leer archivo ejercicio guardado anteriormente
  + Ecuaciones diferenciales ordinarias.
  + Cálculo integral y diferencial.
  + Interpolación y ajuste de curva.
  + Ajuste de curva

**Temas aplicados**

1. Listas
2. Cadenas y funciones de cadena
3. Ciclos for, while
4. Selectiva múltiple, simple
5. Tuplas
6. Operaciones con math y básicas
7. Funciones nativas (os).
8. Manejo de archivos
9. Validación de tipos de datos (cast)
10. Excepciones con try except
11. Funciones (con paso de parámetro, sin paso, con valores de retorno y con valores por default).
12. Entrada y salida de datos.
13. Formato de datos
14. Librería numpy
15. Separación de scripts en archivos y su importación

**CÓDIGO FUENTE**

menu.py (principal): Contiene los menús para el uso de todas las demás funciones dentro del archivo

"""

FUNDAMENTOS FILOSOFICOS DE LA COMPUTACION SECCION D07

NOMBRE DEL ALUMNO: Ismael Iván López Murillo   Código 220975903

NOMBRE DEL ALUMNO: Edgar Manuel Cabral Serpa   Código 217552864

NOMBRE DEL ALUMNO: María José Rangel Salmerón  Código 212557817

Maestro: Conrado Cruz Gomez

PROYECTO FINAL FUNDAMENTOS FILOSÓFICOS

FECHA. 27/NOVIEMBRE/2020

TEMAS APLICADOS:

            \*Listas

            \*Archivos

            \*Ciclos

            \*Selectiva

            \*Try - except

            \*Variables y tipos de dato (cast)

            \*Manejo de cadenas

            \*Funciones (con paso, sin paso, return, default...)

            \*Funciones de libreria

"""

import ajuste         #Ajute

import interpolar     #Interpolar

import integrar       #Integrales

import edo            #Ecuaciones

import os

"""

edo.cargarEjemplo()    ->Ejemplo

edo.nuevoEjercicio()   ->Solucionador

edo.leerEjercicio()    ->Leer ejercicio

"""

*def* mainMenu():

    opcion = "A"

    while opcion != "F":

        os.system("cls")

        print("Bienvenido")

        print("\tA: Integrantes")

        print("\tB: Datos del programa")

        print("\tC: Ejemplos")

        print("\tD: Solucionador")

        print("\tE: Cargar ejercicio desde archivo")

        print("\tF: Salir")

        opcion = *str*(input("Ingrese su eleccion: ")).upper()

        if opcion == "A":

            integrantes()

        elif opcion == "B":

            datos()

        elif opcion == "C":

            ejemplos()

        elif opcion == "D":

            solucionador()

        elif opcion == "E":

            ejercicio()

        elif opcion == "F":

            print("Gracias por usar este programa")

            return

        else:

            print("Seleccione una opcion valida")

        os.system("pause")

*def* ejemplos():

    while True:

        os.system("cls")

        print("EJEMPLOS")

        print("\tA: Ajuste de curva")

        print("\tB: Interpolar valor")

        print("\tC: Integrar numericamente")

        print("\tD: Ecuacion diferencial")

        print("\tE: Salir")

        opcion = *str*(input("Ingrese su eleccion: ")).upper()

        if opcion == "A":

            ajuste.cargarEjemplo()

        elif opcion == "B":

            interpolar.cargarEjemplo()

        elif opcion == "C":

             integrar.cargarEjemplo()

        elif opcion == "D":

            edo.cargarEjemplo()

        elif opcion == "E":

            return

        else:

            print("Seleccione una opcion valida")

        os.system("pause")

*def* integrantes():

    print("Los integrantes del equipo son: ")

    print("Ismael Iván López Murillo")

    print("Edgar Manuel Cabral Serpa")

    print("María José Rangel Salmerón")

*def* datos():

    print("Materia: Fundamentos Filosóficos de la Computación")

    print("Sección: D07")

    print("Profesor: Conrado Cruz Gomez")

    print("Proyecto Final")

    print("Fecha: 27/11/2020")

*def* solucionador():

    while True:

        os.system("cls")

        print("SOLUCIONADOR")

        print("\tA: Ajuste de curva")

        print("\tB: Interpolar valor")

        print("\tC: Integrar numericamente")

        print("\tD: Ecuacion diferencial")

        print("\tE: Salir")

        opcion = *str*(input("Ingrese su eleccion: ")).upper()

        try:

            if opcion == "A":

                ajuste.nuevoEjercicio()

            elif opcion == "B":

                interpolar.nuevoEjercicio()

            elif opcion == "C":

                integrar.nuevoEjercicio()

            elif opcion == "D":

                edo.nuevoEjercicio()

            elif opcion == "E":

                return

            else:

                print("Seleccione una opcion valida")

        except:

            print("La informacion proporcionada genero un error")

        os.system("pause")

*def* ejercicio():

    while True:

        os.system("cls")

        print("Cargar Ejercicios")

        print("\tA: Ajuste de curva")

        print("\tB: Interpolar valor")

        print("\tC: Integrar numericamente")

        print("\tD: Ecuacion diferencial")

        print("\tE: Salir")

        opcion = *str*(input("Ingrese su eleccion: ")).upper()

        if opcion == "A":

            ajuste.leerEjercicio()

        elif opcion == "B":

            interpolar.leerEjercicio()

        elif opcion == "C":

             integrar.leerEjercicio()

        elif opcion == "D":

            edo.leerEjercicio()

        elif opcion == "E":

            return

        else:

            print("Seleccione una opcion valida")

        os.system("pause")

if \_\_name\_\_ == "\_\_main\_\_":

    mainMenu()

validador.py

"""

Funciones para validar

tipos de entrada

"""

*def* requestInt(*message* = "Ingrese un valor entero: ",*warning*="Solo numeros enteros"):

    while True:

        n = input(message)

        try:

            n = *int*(n)

            if n > 1:

                return n

        except *ValueError*:

            print(warning)

*def* requestReal(*message* = "Ingrese el valor correspondiente", *warning*="Ingrese un numero real"):

    while True:

        n = input(message)

        try:

            n = *float*(n)

            return n

        except *ValueError*:

            print(warning)

*def* requestLimits(*message* = "Ingrese los limites de integracion", *warning* = "Ingrese un numero valido"):

    print(message)

    while True:

        a = input("Ingrese el limite inferior: ")

        try:

            a = *float*(a)

            break

        except *ValueError*:

            print(warning)

    while True:

        b = input("Ingrese el limite superior: ")

        try:

            b = *float*(b)

            if b > a:

                return [ a , b ]

        except *ValueError*:

            print(warning)

*def* requestEDO():

    while True:

        edo = input("Ingrese la ecuacion diferencial en terminos de \"x\" y \"y\": ")

        x = y = 0; x+=y   #Odio la warning xd

        try:

            eval(edo)

            return edo

        except:

            continue

archivos.py: Se encarga del manejo de los archivos, crea, solicita y devuelve archivos con un tipo de lectura específico y nombre.

"""

Clase encargada de regresar el contenido de un archivo en

especifico y de cargarlos en un archivo

"""

import os

*def* getRealPath(*filename*):

    directory = os.path.dirname(\_\_file\_\_) # directory of script

    return *r*'{}/archivos/'.format(directory) + filename + ".txt" # File to open

*def* getRealPathToSave(*filename*):

    directory = os.path.dirname(\_\_file\_\_) # directory of script

    return *r*'{}/archivos/saves/'.format(directory) + filename + ".txt" # File to open

*def* leerArchivo(*filename*):

    path = getRealPath(filename)

    file = open(path,"r")

    return file

*def* leerGuardado(*filename*):

    path = getRealPathToSave(filename)

    file = open(path,"r")

    return file

*def* guardarArchivo(*filename*):

    path = getRealPathToSave(filename)

    file = open(path,"w")

    return file

ajuste.py: Archivo encargado de manejar el ajuste de curva con el método de mínimos cuadrados

"""

Ajuste de curva por el metodo de minimos cuadrados

"""

import numpy as np

import validador

import archivos

epsilon = 4

*def* leerEjercicio():

    try:

        filename = input("Ingrese el nombre del archivo a cargar ejemplo: ")

        cargarEjemplo(filename,True)

    except:

        print("ARCHIVO INVALIDO")

*def* cargarEjemplo(*filename*="ajuste",*isSave*=False):

    if isSave:

        file = archivos.leerGuardado(filename)

    else:

        file = archivos.leerArchivo(filename)

    datos = file.read().split("\n")

    datos.pop()

    data = []

    for dato in datos:

        valores = dato.split("|")

        data.append([*float*(valores[0]) , *float*(valores[1])])

    calcular(data, len(data)-1)

    file.close()

*def* guardarEjercicio(*data*,*filename*):

    file = archivos.guardarArchivo(filename)

    for dato in data:

        file.write(*f*"{dato[0]}|{dato[1]}\n")

    file.close()

*def* nuevoEjercicio():

    print("\t\t\tMinimos cuadrados\t\t\t")

    n = validador.requestInt("Ingrese el numero de datos a ingresar: ")

    data = []

    for i in range(n):

        x = validador.requestReal(*f*"Ingrese el valor para x{i}: ")

        y = validador.requestReal(*f*"Ingrese el valor para f({x}): ")

        data.append([x,y])

    polinomios = validador.requestInt("Ingrese los polinomios a calcular: ")

    if polinomios > len(data) - 1 :

        print("Se haran los polinomios maximos si se excede")

        polinomios = len(data) - 1

    calcular(data,polinomios)

    save = input("Ingrese s o S si quiere guardar este ejercicio: ")

    if save == 'S' or save == 's':

        filename = input("Ingrese el nombre del archivo (sin extension): ")

        guardarEjercicio(data,filename)

*def* calcular(*data*,*polynomials*):

    print("\t\tMinimos cuadrados\n\n")

    print("Valores\n")

    for valor in data:

        print(*f*"x = {valor[0]} , y = {valor[1]}")

    print("")

    xSum = [0 for i in range(2\*polynomials)]

    xYSum = [0 for i in range(polynomials+1)]

    for i in range(polynomials\*2):

        xSum[i] = getSummatory(data,i+1)

    for i in range(polynomials+1):

        xYSum[i] = getSumXY(data,i)

    for i in range(len(xSum)):

        print(*f*"x^{i+1} = {xSum[i]}")

    for i in range(len(xYSum)):

        print(*f*"x^{i+1}y = {xYSum[i]}")

    print("\n\n")

    for i in range(polynomials):

        print(*f*"Solucion polinomio grado {i+1}")

        print("\*"\*10,"Sistema","\*"\*10)

        coefMatr = [0 for i in range(i+2)]

        coefSol = [0 for i in range(i+2)]

        for j in  range(i+2):

            coefEq = getEquation(i+1,j,xSum,len(data))

            coefMatr[j] = coefEq

            coefSol[j] = xYSum[j]

            print(*f*" = {xYSum[j]}")

        a = np.array(coefMatr)

        b = np.array(coefSol)

        x = np.linalg.solve(a, b)

        print("\*"\*10,"Soluciones","\*"\*10)

        for i in range(len(x)):

            print(*f*"a{i} = {round(x[i],epsilon)}")

        print("\*"\*10,"Ecuacion","\*"\*10)

        for i in range(len(x)):

            if i == 0:

                print(*f*"y = {round(x[i],epsilon)}",*end*='')

            else:

                print(*f*" + {round(x[i],epsilon)}x^{i}",*end*='')

        print("\n")

#Order -> Orden del polinomio

#n -> Numero de ecuacion

#xSum -> Coeficientes x^n

#data -> n (Si se requiere, solo para n=0)

*def* getEquation(*order*,*n*,*xSum*,*terms*):

    data = [0 for i in range(order+1)]  #Order 1 -> 2 terminos y asi

    for i in range(len(data)): #Se llena el array

        if i == 0 and n == 0:    #No podemos acceder a data[-1]

            data[0] = terms

        else:

            data[i] = round(xSum[n-1+i],epsilon)

        if i != 0:

            print(" + ",*end*='')

        print(*f*"{data[i]*:.4f*}a{i}",*end*='')

    return data

*def* getSumXY(*data*,*pow*):

    total = 0

    for i in range(len(data)):

        total += data[i][0]\*\*pow \*data[i][1]

    return round(total,epsilon)

*def* getSummatory(*data*,*pow*):

    total = 0

    for i in range(len(data)):

        total += data[i][0]\*\*pow

    return round(total,epsilon)

edo.py: Archivo encargado de manejar los problemas de valor inicial de ecuaciones diferenciales ordinarias con Runge-Kutta

"""

Solucionar de PVI usando metodo de Runge Kutta

"""

epsilon = 4

import validador

import archivos

*def* leerEjercicio():

    try:

        filename = input("Ingrese el nombre del archivo a cargar ejemplo: ")

        cargarEjemplo(filename,True)

    except:

        print("ARCHIVO INVALIDO")

*def* cargarEjemplo(*filename* = "edo",*isSave* = False):

    if isSave:

        file = archivos.leerGuardado(filename)

    else:

        file = archivos.leerArchivo(filename)

    datos = file.read().split("\n")

    edo = datos[0]

    x0 = *float*(datos[1])

    y0 = *float*(datos[2])

    xf = *float*(datos[3])

    n = *int*(datos[4])

    calcular(edo,x0,y0,xf,n)

    file.close()

*def* nuevoEjercicio():

    edo = validador.requestEDO()

    x0 = validador.requestReal("Ingrese el valor inicial de x (x0): ")

    y0 = validador.requestReal(*f*"Ingrese el valor de f({x0}): ")

    xf = validador.requestReal("Ingrese el valor de x a encontrar y (xf): ")

    n = validador.requestInt("Ingrese el numero de subintervalos a dividir: ")

    calcular(edo,x0,y0,xf,n)

    save = input("Ingrese s o S si quiere guardar este ejercicio: ")

    if save == 'S' or save == 's':

        filename = input("Ingrese el nombre del archivo (sin extension): ")

        guardarEjercicio(edo,x0,y0,xf,n,filename)

*def* guardarEjercicio(*edo*,*x0*,*y0*,*xf*,*n*,*filename*):

    file = archivos.guardarArchivo(filename)

    file.write(*f*"{edo}\n")

    file.write(*f*"{x0}\n")

    file.write(*f*"{y0}\n")

    file.write(*f*"{xf}\n")

    file.write(*f*"{n}\n")

    file.close()

*def* calcular(*edo*,*x0*,*y0*,*xf*,*n*):

    print("\t\tRunge-Kutta para E.D.O con P.V.I")

    h = round((xf-x0)/n,epsilon)

    print("EDO: ",edo)

    print(*f*"f({x0}) = {y0}")

    print(*f*"f({xf}) = ?\n")

    print(*f*"h = ({xf}-{x0})/{n}")

    print(*f*"h = {h}")

    xn = x0

    yn = y0         #Para empezar

    for i in range(n):

        print("\*"\*10,"Iteracion ",i+1,"\*"\*10)

        xi = round(xn,epsilon)

        yi = round(yn,epsilon)

        xn = xi + h

        print(*f*"x{i+1} = {xi} + {h} = {xn}")

        print(*f*"k1 = f({xi},{yi}) = ",*end*='')

        k1 = round(getFunction(edo,xi,yi),epsilon)

        print(*f*"{printFunction(edo,xi,yi)} = {k1}")

        print(*f*"k2 = f({xi} + {h}/2,{yi} + {h}\*{k1}/2) = ",*end*='')

        k2 = round(getFunction(edo,xi+h/2, yi+h\*k1/2),epsilon)

        print(*f*"{printFunction(edo,xi + h/2,yi+h\*k1/2)} = {k2}")

        print(*f*"k3 = f({xi} + {h}/2,{yi} + {h}\*{k2}/2) = ",*end*='')

        k3 = round(getFunction(edo,xi+h/2,yi + h\*k2/2),epsilon)

        print(*f*"{printFunction(edo,xi+h/2,yi + h\*k2/2)} = {k3}")

        print(*f*"k4 = f({xi} + {h},{yi} + {h}\*{k3}) = ",*end*='')

        k4 = round(getFunction(edo,xi+h, yi + h\*k3),epsilon)

        print(*f*"{printFunction(edo,xi + h, yi + h\*k3)} = {k4}")

        yn = yi + (h/6)\*(k1+2\*k2+2\*k3+k4)

        print(*f*"y{i+1} = {yi} + ({h}/6)({k1}+2({k2})+2({k3})+{k4}) = {yn*:.4f*}")

*def* printFunction(*edo*,*x*,*y*):

    nuevaFuncion = edo

    nuevaFuncion = nuevaFuncion.replace("x","{x}")

    nuevaFuncion = nuevaFuncion.replace("y","{y}")

    nuevaFuncion = *f*"f\"{nuevaFuncion}\""

    return eval(nuevaFuncion)

*def* getFunction(*edo*,*x*,*y*):

    return eval(edo)

integral.py: Archivo encargado de manejar la integral con el método de Newton Cotes

"""

Formulas de integracion con Newton Cotes

"""

epsilon = 4

import math

import archivos

import validador

*def* leerEjercicio():

    try:

        filename = input("Ingrese el nombre del archivo a cargar ejemplo: ")

        cargarEjemplo(filename,True)

    except:

        print("ARCHIVO INVALIDO")

*def* cargarEjemplo(*filename*="integrar",*isSave*=False):

    if isSave:

        file = archivos.leerGuardado(filename)

    else:

        file = archivos.leerArchivo("integrar")

    datos = file.read().split("\n")

    datos.pop()

    headers = datos[0].split("|")

    a = *float*(headers[0])

    b = *float*(headers[1])

    n = *int*(headers[2])

    data = []

    for i in range(1,len(datos)):

        data.append(*float*(datos[i]))

    calcular(a,b,n,data)

    file.close()

*def* guardarEjercicio(*a*,*b*,*h*,*data*,*filename*):

    file = archivos.guardarArchivo(filename)

    file.write(*f*"{a}|{b}|{h}\n")

    for dato in data:

        file.write(*f*"{dato}\n")

    file.close()

*def* nuevoEjercicio():

    limits = validador.requestLimits("Ingrese los limites de integracion: ")

    n = validador.requestInt("Ingrese cuantos subintervalos tiene: ")

    fun = input("Ingrese la funcion a tabular en terminos de x y sintaxis python: ")

    h = (limits[0] - limits[1])/n

    data = []

    if fun == "":

        for i in range(n+1):

            x = limits[0] + h\*i

            y = validador.requestReal(*f*"Ingrese el valor para f{x}: ")

            data.append(y)

    else:

        try:

            x = 1

            eval(fun)

            for i in range(n+1):

                x = limits[0] + h\*i

                y = eval(fun)

                data.append(y)

        except:

            print("La funcion no es valida")

            return

    calcular(limits[0],limits[1],n,data)

    save = input("Ingrese s o S si quiere guardar este ejercicio: ")

    if save == 'S' or save == 's':

        filename = input("Ingrese el nombre del archivo (sin extension): ")

        guardarEjercicio(limits[0],limits[1],n,data,filename)

*def* calcular(*a*,*b*,*n*,*data*):

    print("\t\tNewton Cotes\n\n")

    h = (b-a)/n

    for i in range(len(data)):

        x = a + h\*i

        y = data[i]

        print(*f*"x = {x} , y = {y}")

    print("\n","\*"\*10,"Trapecio compuesto","\*"\*10)

    getCompositeTrapeze(data,h)

    print("\*"\*10,"Simpson 1/3 compuesto","\*"\*10)

    getSimpson1over3(data,h)

    print("\*"\*10,"Simpson 3/8 compuesto","\*"\*10)

    getSimpson3over8(data,h)

*def* getCompositeTrapeze(*data*,*h*):

    total = data[0]

    print(*f*"{h}/2({data[0]} + 2(",*end*='')

    for i in range(1,len(data)-1):

        if i!= 1:

            print(" + ",*end*='')

        print(data[i],*end*='')

        total += 2\*data[i]

    print(*f*") + {data[len(data)-1]}) = ",*end*='')

    total += data[len(data)-1]

    total = round(total\*h/2,epsilon)

    print(total)

    return total

*def* getSimpson1over3(*data*,*h*):

    total = data[0]

    n = len(data)

    print(*f*"{h}/3({data[0]} + 4(",*end*='')

    for i in range(1,n-1,2):

        if i != 1:

            print(" + ",*end*='')

        print(data[i],*end*='')

        total += 4\*data[i]

    print(") + 2(",*end*='')

    for i in range(2,n-1,2):

        if i != 2:

            print(" + ",*end*='')

        print(data[i],*end*='')

        total += 2\*data[i]

    print(*f*") + {data[n-1]}) = ",*end*='')

    total += data[n-1]

    total = round(total\*h/3,epsilon)

    print(total)

    return total

*def* getSimpson3over8(*data*,*h*):

    n = len(data)

    total = data[0]

    print(*f*"3\*{h}/8({data[0]} + 3(",*end*='')

    for i in range(1,n-1):

        if i % 3 != 0:

            if i != 1:

                print(" + ",*end*='')

            print(data[i],*end*='')

            total += 3\*data[i]

    print(") + 2(",*end*='')

    for i in range(1,n-3):

        if i\*3 < n:

            if i != 1:

                print(" + ",*end*='')

            print(data[i\*3],*end*='')

            total += 2\*data[i\*3]

    print(*f*") + {data[n-1]}) = ",*end*='')

    total += data[n-1]

    total = round(3/8\*total\*h,epsilon)

    print(total)

    return total

interpolar.py: Archivo encargado del método de interpolación con diferencias finitas.

“””

INTERPOLACION MEDIANTE EL USO

DE DIFERENCIAS FINITAS

“””

import validador

import archivos

delant = 4

*def* leerEjercicio():

    try:

        filename = input(“Ingrese el nombre del archivo a cargar ejemplo: “)

        cargarEjemplo(filename,True)

    except:

        print(“ARCHIVO INVALIDO”)

*def* cargarEjemplo(*filename*=”interpolar”,*isSave*=False):

    if isSave:

        file = archivos.leerGuardado(filename)

    else:

        file = archivos.leerArchivo(filename)

    text = file.read().split(“\n”)

    text.pop()

    value = *float*(text[0])

    data = []

    for I in range(1,len(text)):

        vals = text[i].split(“|”)

        data.append( [*float*(vals[0]), *float*(vals[1])])

    calcular(data,value)

    file.close()

*def* nuevoEjercicio():

    n = validador.requestInt(“Ingrese el numero de datos a ingresar: “)

    data = []

    for I in range(n):

        val = []

        x = validador.requestReal(*f*”Ingrese el valor para x{i}: “)

        y = validador.requestReal(*f*”Ingrese el valor para f(x{i}): “)

        val.append(x)

        val.append(y)

        data.append(val)

    value = validador.requestReal(“Ingrese el punto a interpolar: “)

    calcular(data,value)

    del = input(“Ingrese \”S\” si quiere guardar este ejercicio: “)

    if save == ‘S’ or save == ‘s’:

        filename = input(“Ingrese el nombre del archivo (sin delante): “)

        guardarEjercicio(value,data,filename)

*def* guardarEjercicio(*value*,*data*,*filename*):

    file = archivos.guardarArchivo(filename)

    file.write(*str*(value)+”\n”)

    for dato in data:

        file.write(*f*”{dato[0]}|{dato[1]}\n”)

    file.close()

*def* calcular(*data*, *value*):

    print(“\t\tDiferencias finitas\n\n”)

    print(“Datos: “)

    for valor in data:

        print(*f*”x = {valor[0]} , y = {valor[1]}”)

    print(*f*”Valor a interpolar: {value}\n\n”)

    dif = getFiniteDifferences(data)

    n = len(data)

    offset = findOffest(data,value)

    print(offset)

    forward = findDirecion(data,offset)

    if forward:

        polynomials = n – 1 – offset

    else:

        offset += 1

        polynomials = offset

    s = getS(data,value,offset,forward)

    for I in range(polynomials):

        total = data[offset][1]

        print(*f*”{total} + “, *end* = ‘’)

        for j in range(i+1):

            if j != 0:

                print(“ + “,*end*=’’)

            total += getMultiplers(s,j,forward) \* getDif(dif,offset,j,forward) / factorial(j+1)

        total = round(total,epsilon)

        print(*f*” = {total*:.4f*}”)

*def* getDif(*dif*,*offset*,*j*,*forward*):

    if forward:

        A = dif[offset][j+1]

    else:

        A = dif[offset-j-1][j+1]

    print(“(%.4f)”%A,*end*=’’)

    return A

*def* factorial(*n*):

    t = 1

    for I in range(1,n+1):

        t \*= i

    print(*f*”/{n}!”,*end*=’’)

    return t

*def* getMultiplers(*s*,*n*,*direction*):

    t = s

    print(“%.4f”%s,*end*=’’)

    for I in range(1,n+1):

        if direction:

            t \*= (s-i)

            print(“(%.4f - %d)”%(s,i),*end*=’’)

        else:

            t \*= (s+i)

            print(“(%.4f + %d)”%(s,i),*end*=’’)

    return t

*def* getS(*data*,*value*,*off*,*direction*):

    if direction: #Dependiendo si es hacia adelante o hacia atrás

        h = abs(data[off][0] – data[off+1][0])

        print(*f*”s = ({value} – {data[off][0]})/{h} = “,*end*=’’)

        s = (value – data[off][0])/(h)

    else:

        h = abs(data[off][0] – data[off+1][0])

        print(*f*”s = ({data[off][0]} – {value})/{h} = “,*end*=’’)

        s = (data[off][0] – value)/(h)

    print(s,*end*=’\n\n’)

    return round(s,epsilon)

*def* getFiniteDifferences(*data*):

    n = len(data)

    dif = [[0 for I in range(n)] for j in range(n)]

    for i in range(len(dif)): #Llenar con ceros para valores no definidos

        for j in range(len(dif[0])):

            dif[i][j] = 0

    for I in range(n):  #Primera columna es y

        dif[i][0] = data[i][1]

    for i in range(n):   #Repetir n-1 porque ya tenemos a0

        diff = i+1 #Primera delante necesitamos n-1, delant n-2 y así y así

        for j in range(n-diff):

            print(*f*”{dif[j+1][i]*:.4f*} – {dif[j][i]*:.4f*}”)

            ak =  round(dif[j+1][i] – dif[j][i],epsilon)

            print(“Δ^%d [%d] = %.4f\n”%(i+1,j,ak))

            dif[j][i+1] = ak

    printMatrix(dif) #Imprimir la matriz resultante

    return dif

*def* findOffest(*data*, *value*):

    n = len(data)

    for I in range(n-1):

        if data[i][0] < value and data[i+1][0] > value:

            return I

    if data[0][0] > value:

        return 0

    else:

        return n-1

*def* findDirecion(*data*,*offset*):

    if offset < len(data) // 2:

        print(“\t\tHacia delante\n\n”)

        return True

    else:

        print(“\t\tHacia atras\n\n”)

        return False

*def* printMatrix(*data*):

    for I in range(len(data)):

        for j in range(len(data[0])):

            print(“%f”%data[i][j],*end*=’\t’)

        print(“\n”)

**Conclusión**

Aunque tenía experiencia con la programación en general, cuando entré al curso no tenía ni la más mínima idea sobre Python.

Lo había manejado para alguna clase anteriormente, sin embargo, hice cosas muy específicas que realmente no necesitaban de que yo pensara en lo que estaba escribiendo, solamente en hacer y repetir código, así que podía decir que mi base de Python no era sólida.

Es por ello por lo que me agradó mucho que una clase que iba a llevar se enfocara en aprender Python totalmente desde cero, porque ya más adelante ningún maestro se va a tomar el tiempo de enseñar desde cero ningún lenguaje, y Python es uno de los más utilizados actualmente, así que me encantó la idea de cursar esta materia.

Al inicio aprendí lo fundamental y esto me sirvió para empezar a comprender todo lo que no entendía del lenguaje, que, por más sencillo que sea, nunca me había dado el tiempo de aprenderlo ni investigarlo.

Con lo fundamental como entrada, salida, condicionales, sintaxis y funciones me bastó para animarme a hacer más cosas en otras materias o cuando quería hacer pruebas de alguna cosa.

Poco a poco dejé de quedarme solamente con el conocimiento en la clase y empecé a buscar por mi cuenta, usé este lenguaje para resolver problemas de mi materia de seminario de métodos matemáticos III, es por esta razón que tuvimos la idea de hacer un formulario de métodos numéricos que se encargue de resolver este tipo de problemáticas.

Y realmente Python te da todas las herramientas para este tipo de problemas, y es tan sencillo y flexible que no se nos complicó demasiado importar un algoritmo a un lenguaje.

Referente a lo aprendido fuera de clase y que se aplicó en el proyecto, como lo fueron archivos o la librería numpy, estas funciones son muy sencillas, realmente hay mucha documentación en internet y creo que todo esto lo hace más ameno al momento de programarlo.

Trabajar en equipo fue un alivio, ya que pudimos dividirnos el trabajo y aprendimos mucho unos de otros.

Una materia en la que aprendí mucho y creo que pude sacarle mucho potencial una vez teniendo una base sólida en el tema. Se complicó muy poco en general.

**Referencias**

* Libros
  + Pinales Delgado, F. & Velázquez Amador, C. (2014). *Algoritmos resueltos con diagramas de flujo y pseudocódigo.* México: Universidad Autónoma de Aguascalientes.
  + Cervantes Villagómez, O., Báez López, D., Arízaga Silvia, A. & Castillo Juárez, E., 2017. *Phyton Con Aplicación A Las Matemáticas, Ingeniería Y Finanzas*.