

“ANÁLISIS NUMÉRICO I”
“MÉTODOS MATEMÁTICOS Y NUMÉRICOS”
<75.12> <95.04> <95.13>

DATOS DEL TRABAJO PRÁCTICO

| | | |
|---------------|-------------|--|
| 2 | AÑO | 2023 |
| | 1 | Balance de Oxígeno Disuelto en la laguna Mar Chiquita |
| TP NRO | CUAT | TEMA |

INTEGRANTES DEL GRUPO

| | | |
|----------|------------------------|---------------|
| - | Lopez francisco | 107614 |
| - | Corn Franco | 109025 |

Introducción

En este segundo trabajo práctico se pretende evaluar el balance de OD que se registra en la laguna para un año de operación normal. Para ello tendremos como datos los hidrogramas de entrada y salida, en donde el agua se almacena durante las épocas de sequía y se descarga en épocas de alta.

Objetivos

- Plantear el problema y discretizar las ecuaciones de OD y DBO , con esquemas de orden 1 y 2.
- Realizar un ajuste con los datos cota-volumen de la laguna con polinomios de grado 2 y 3
- Implementar la solución numérica en un código computacional utilizando un paso de discretización temporal diario utilizando la curva que mejor represente los datos de cota-volumen obtenidos anteriormente
- Graficar OD y DBO resultantes
- Estimar el error de truncamiento de OD y DBO con los 2 métodos y graficarlos
- Determinar el porcentaje en que se debe reducir la concentración de DBO del río para que en todo el año se garantice OD mínimo en la laguna de 4 g/m^3 . Graficar OD y DBO resultante en la laguna.
- Comprobar qué sucede si usamos un paso de discretización semanal en lugar de diario

Desarrollo

Para el desarrollo de nuestro tp utilizaremos python como lenguaje de programación, utilizaremos las bibliotecas: math, numpy, pandas . Además, utilizaremos excel para calcular los polinomios de grado 2 y 3 como nos piden en el punto B.

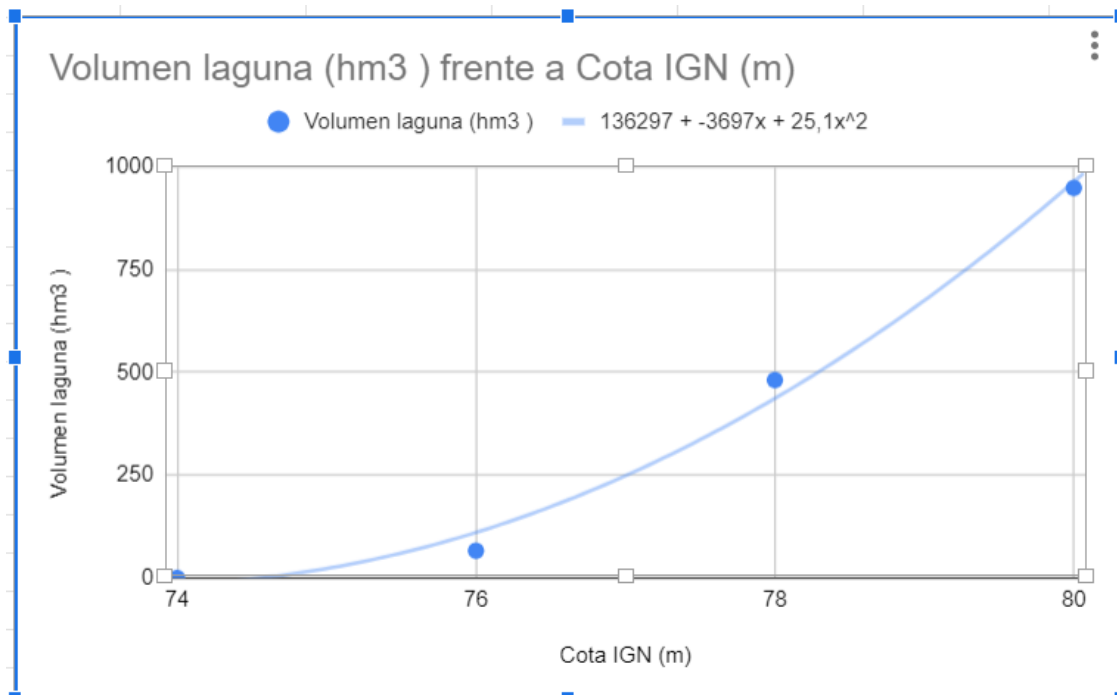
Importante: Toda la parte que tenga que ver con mostrar el código, la dejamos para el índice de “Anexo”

Ajustar los datos cota-volumen de la laguna con polinomios de grados 2 y 3

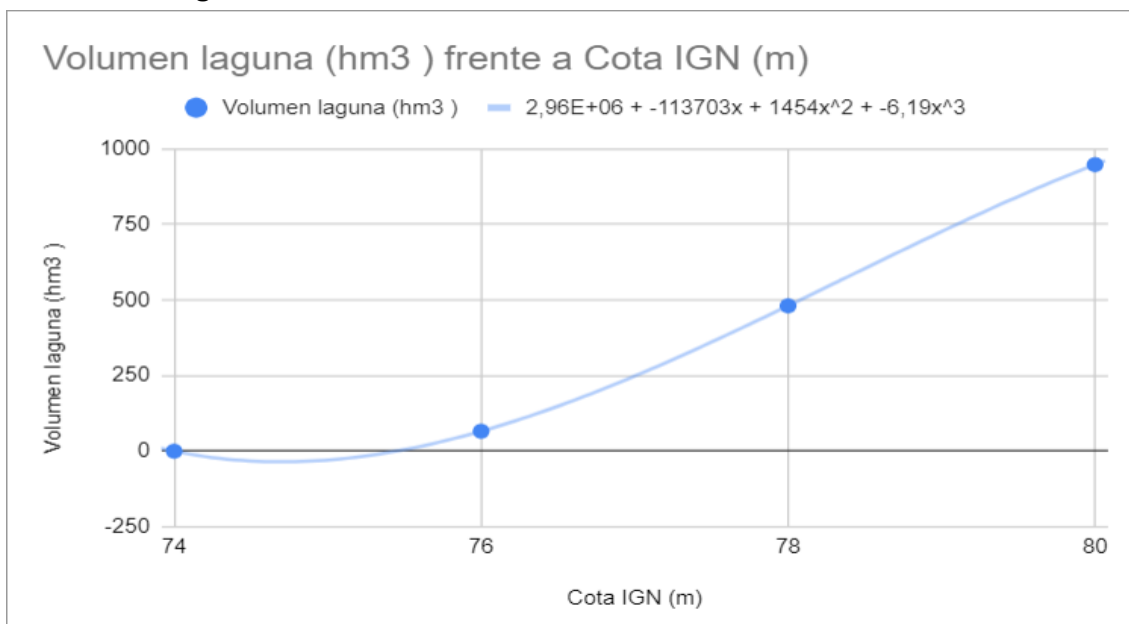
Para la búsqueda de los polinomios utilizamos el legajo(109025) y utilizando excel y su función “línea de tendencia” hallaremos los polinomios pedidos .

Utilizando la tabla dada en el tp, llegamos a que los gráficos para los polinomios de grado 2 y 3 son:

Polinomio de grado 2:



Polinomio de grado 3:



La curva que mejor aproxima los puntos es la del polinomio de grado 3. Ya que se ajusta mejor a los datos que tenemos (es más exacta al pasar por los puntos). Como se puede ver en los gráficos la curva de grado 2 no llega a pasar exactamente por los primeros dos puntos, en cambio la curva de grado 3 si lo hace.

implementar la solución numérica

Utilizando nuestro polinomio de grado 3 obtenemos que para una $Cota_0 = 77m$ (utilizando el padrón 109025) utilizando la ecuación del polinomio llegamos a que el volumen inicial resulta: $V_i = 235 \text{ h m}^3$.

Para implementar nuestra solución numérica tenemos que discretizar nuestras ecuaciones, busquemos implementar dos algoritmos para analizar el OD, en primer lugar usaremos un método de orden 1 (Euler explícito) y luego uno de orden 2 (Punto medio). Empezamos discretizando nuestras ecuaciones:

$$\frac{dV}{dt} = Q_e - Q_s \quad (1)$$

La ecuación (1) representa el balance de agua en nuestra laguna, siendo Q_e nuestro caudal de entrada y Q_s nuestro caudal de salida.

Discretizando:

- $t^{(n)} = nh$
- $V(t^{(n)}) = V^{(n)} \rightarrow W^{(n)}$
- $V'(t^{(n)}) = V'^{(n)} \rightarrow \frac{W^{(n+1)} - W^{(n)}}{h}$

Una vez discretizado el volumen necesitamos discretizar la ec de OD y DBO, planteamos nuestras ecuaciones:

$$V \frac{dc}{dt} = Q_e c_e - Q_s c + G - P \quad (2)$$

$$P = k_{bd} V DBO \quad (3)$$

$$k_{bd} = k_{bd0} \frac{OD^2}{OD^2 + K_{O2}^2} \quad (4)$$

$$G = k_a V (OD_s - OD) \quad (5)$$

Antes de discretizar reemplazamos en nuestra ecuación (2) con el resto de nuestras ecuaciones y llegamos a:

$$\text{OD: } V \frac{dOD}{dt} = Q_e \cdot OD_e - Q_s \cdot OD + K_a \cdot V (OD_s - OD) - k_{bd0} \left(\frac{OD^2 DBO}{OD^2 + K_{O2}^2} \right)$$

$$\text{DBO: } V \frac{dDBO}{dt} = Q_e \cdot DBO_e - Q_s \cdot DBO + 0 - k_{bd0} \left(\frac{OD^2 DBO}{OD^2 + K_{O2}^2} \right) \quad (G = 0)$$

Una vez obtenidas las ecs de OD y DBO discretizamos:

OD

- $t^{(n)} = nh$
- $OD(t^{(n)}) = OD^{(n)} \rightarrow OD_1^{(n)}$
- $OD'(t^{(n)}) = OD'^{(n)} \rightarrow \frac{OD_1^{(n+1)} - OD_1'^{(n)}}{h}$

DBO

- $t^{(n)} = nh$
- $DBO(t^{(n)}) = DBO^{(n)} \rightarrow DBO_1^{(n)}$

- $DBO'(t^{(n)}) = DBO'^{(n)} \rightarrow \frac{DBO_1^{(n+1)} - DBO_1'^{(n)}}{h}$

Ahora que tenemos nuestras tres ecuaciones discretizadas procedemos a reemplazar en los métodos de orden 1 y 2 mencionados anteriormente:

Para euler explícito sabemos :

$$u_{n+1} = u_n + k f(u_n, t_n)$$

Reemplazando en nuestras ecuaciones de volumen, OD y DBO:

(Volumen)

$$W^{(n+1)} = W^{(n)} + h(Qe - Qs)$$

(OD)

$$OD_1^{(n+1)} = OD_1^{(n)} + h\left(\frac{Qe \cdot ODe}{W^{(n)}} - \frac{Qs \cdot OD_1^{(n)}}{W^{(n)}} + ka(ODs - OD_1^{(n)}) - kbdo\left(\frac{DBO(n)(OD_1^{(n)})^2}{(k02)^2 + (OD_1^{(n)})^2}\right)\right)$$

(DBO)

$$DBO_1^{(n+1)} = DBO_1^{(n)} + h\left(\frac{Qe \cdot DBOe}{W^{(n)}} - \frac{Qs \cdot DBO_1^{(n)}}{W^{(n)}} - kbdo\left(\frac{DBO(n)(OD_1^{(n)})^2}{(k02)^2 + (OD_1^{(n)})^2}\right)\right)$$

Para el método de punto medio tenemos:

$$u_{n+1/2} = u_n + k/2 f(u_n, t_n)$$

$$u_{n+1} = u_n + k f(u_{n+1/2}, t_{n+1/2})$$

Reemplazando en nuestras ecuaciones de volumen, OD y DBO:

(Volumen)

$$W^{(n+1/2)} = W^{(n)} + \frac{h}{2} (Qe - Qs)$$

$$W^{(n+1)} = W^{(n)} + h(Qe - Qs)$$

(OD)

$$OD_1^{(n+1/2)} = OD_1^{(n)} + \frac{h}{2} \left(\frac{Qe \cdot ODe}{W^{(n)}} - \frac{Qs \cdot OD_1^{(n)}}{W^{(n)}} + ka(ODs - OD_1^{(n)}) - kbdo \left(\frac{DBO(n)(OD_1^{(n)})^2}{(k02)^2 + (OD_1^{(n)})^2} \right) \right)$$

$$OD_1^{(n+1)} = OD_1^{(n)} + h \left(\frac{Qe \cdot ODe}{W^{(n)}} - \frac{Qs \cdot OD_1^{(n+1/2)}}{W^{(n)}} + ka(ODs - OD_1^{(n)}) - kbdo \left(\frac{DBO(n)(OD_1^{(n+1/2)})^2}{(k02)^2 + (OD_1^{(n+1/2)})^2} \right) \right)$$

(DBO)

$$DBO_1^{(n+1/2)} = DBO_1^{(n)} + \frac{h}{2} \left(\frac{Qe \cdot DBOe}{W^{(n)}} - \frac{Qs \cdot DBO_1^{(n)}}{W^{(n+1)}} - kbdo \left(\frac{DBO(n)(OD_1^{(n)})^2}{(k02)^2 + (OD_1^{(n)})^2} \right) \right)$$

$$DBO_1^{(n+1)} = DBO_1^{(n)} + \frac{h}{2} \left(\frac{Qe \cdot DBOe}{W^{(n)}} - \frac{Qs \cdot DBO_1^{(n+1/2)}}{W^{(n)}} - kbdo \left(\frac{DBO_1^{(n+1/2)}(OD_1^{(n)})^2}{(k02)^2 + (OD_1^{(n)})^2} \right) \right)$$

Con estos dos métodos vamos a aproximar como varía el DBO y el OD en la laguna. Recordamos que el volumen cambia día a día, mientras que los caudales son mensuales, es importante aclarar que el volumen se encuentra en $h m^3$ y nuestros caudales en m^3/s entonces vamos a pasar nuestro volumen a m^3 y nuestros caudales a $m^3/día$:

$$V = h m^3 \rightarrow V = (m^3) 10^6$$

$$Q = m^3/s \rightarrow Q = (m^3/día) 86400$$

Una vez pasados V y Q nos restaría conocer DBO(inicial) y OD(inicial) lo que haremos será simular un año y tomar dichos valores como iniciales, tomamos entonces como valores iniciales:

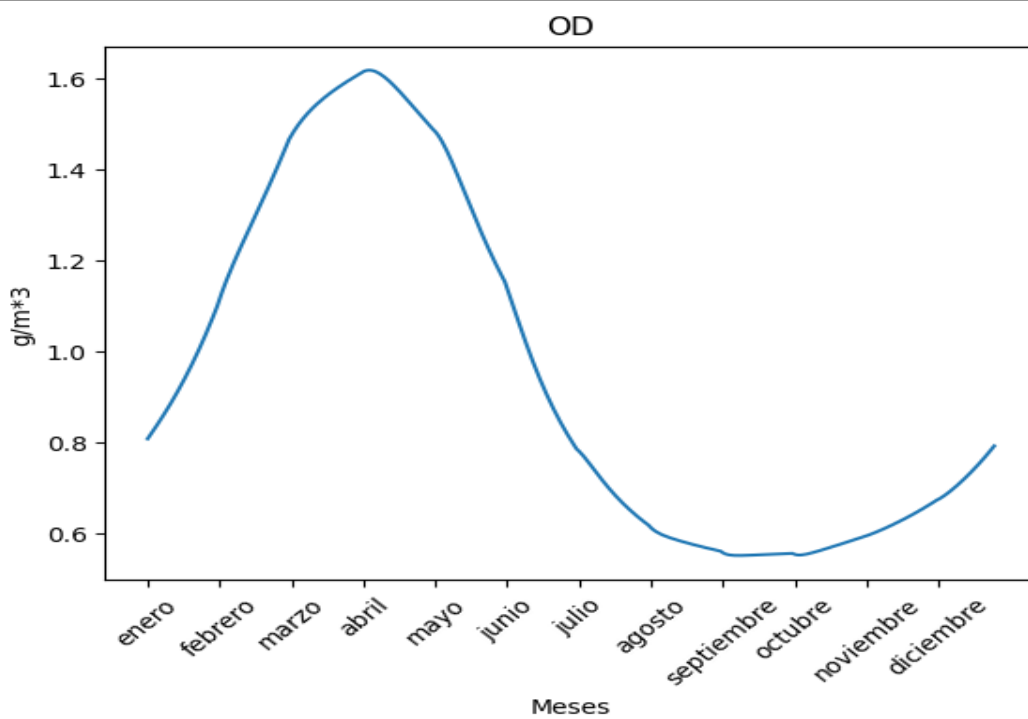
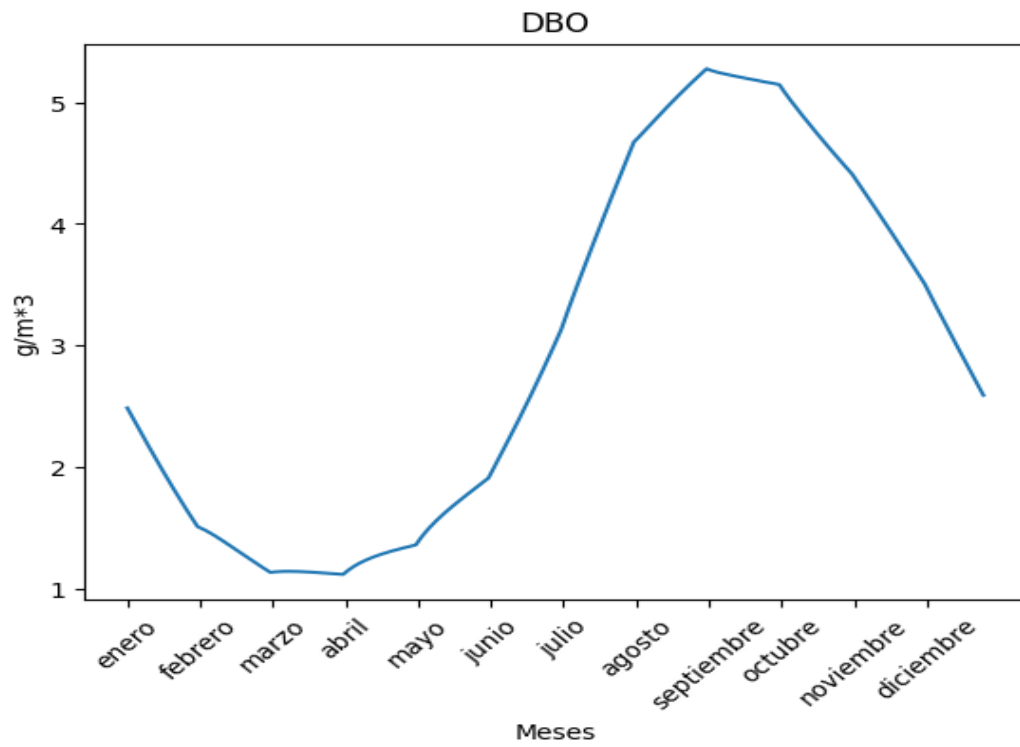
$$DBO(\text{inicial}) = 2.52$$

$$OD(\text{inicial}) = 0.80$$

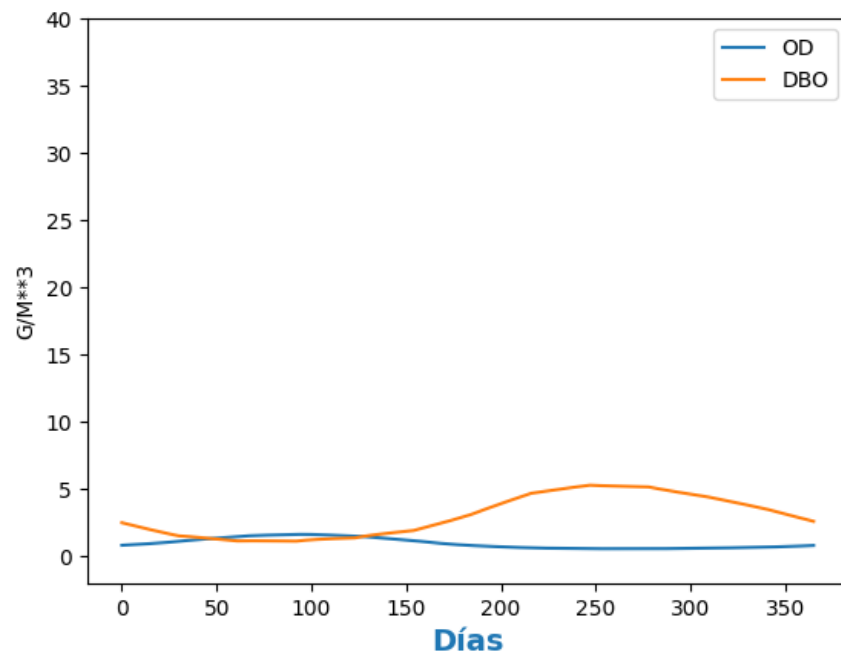
Estos son los resultados de la simulación.

Utilizando euler explícito simulamos un año normal y obtenemos los siguientes gráficos:

Empezamos graficando OD y DBO por separado

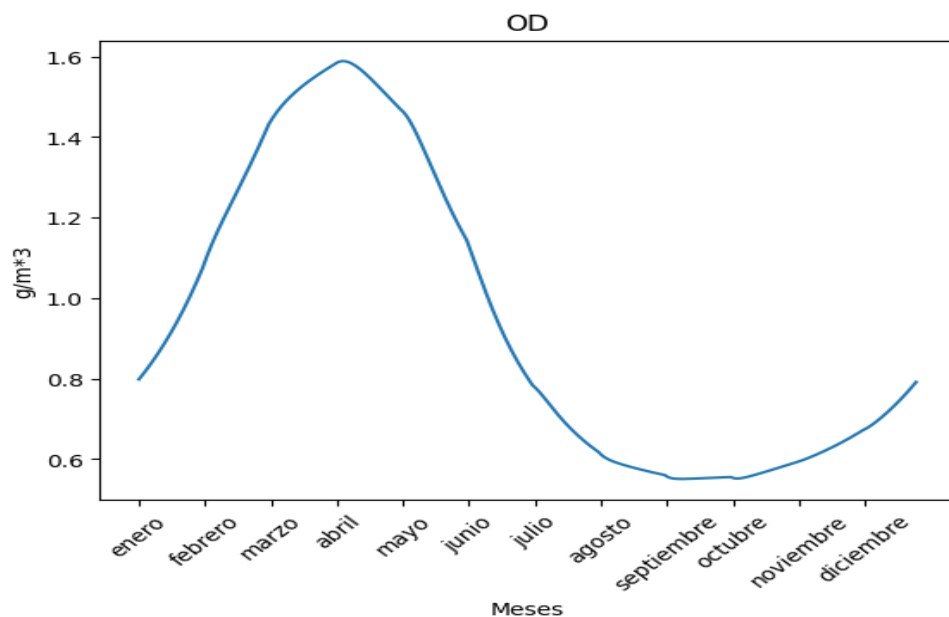


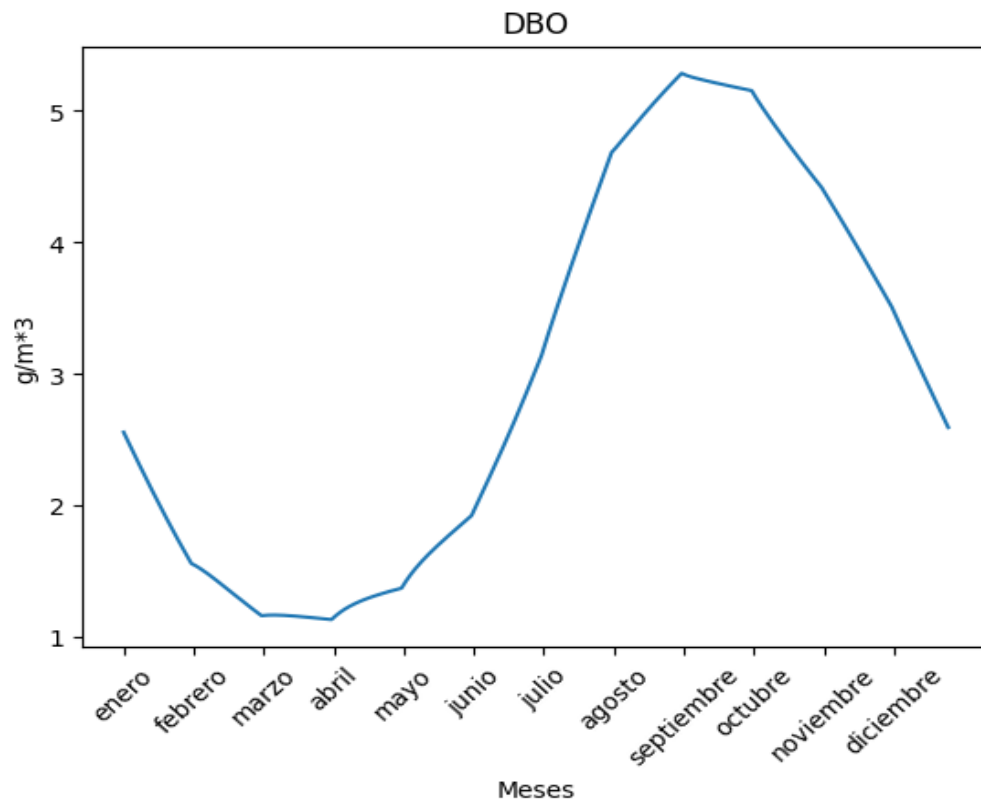
Graficamos OD y DBO juntos



Notamos que a medida que el OD baja el DBO aumenta y viceversa, por lo que los gráficos tienen sentido. También observamos que el lago podría estar muy contaminado si bien necesitamos analizar todos los factores además del análisis de DBO y OD vemos que la cantidad de OD en el lago es bastante más baja que la de DBO, esto podría indicar que el lago está contaminado.

Ahora simulamos 5 años y volvemos a graficar los resultados:





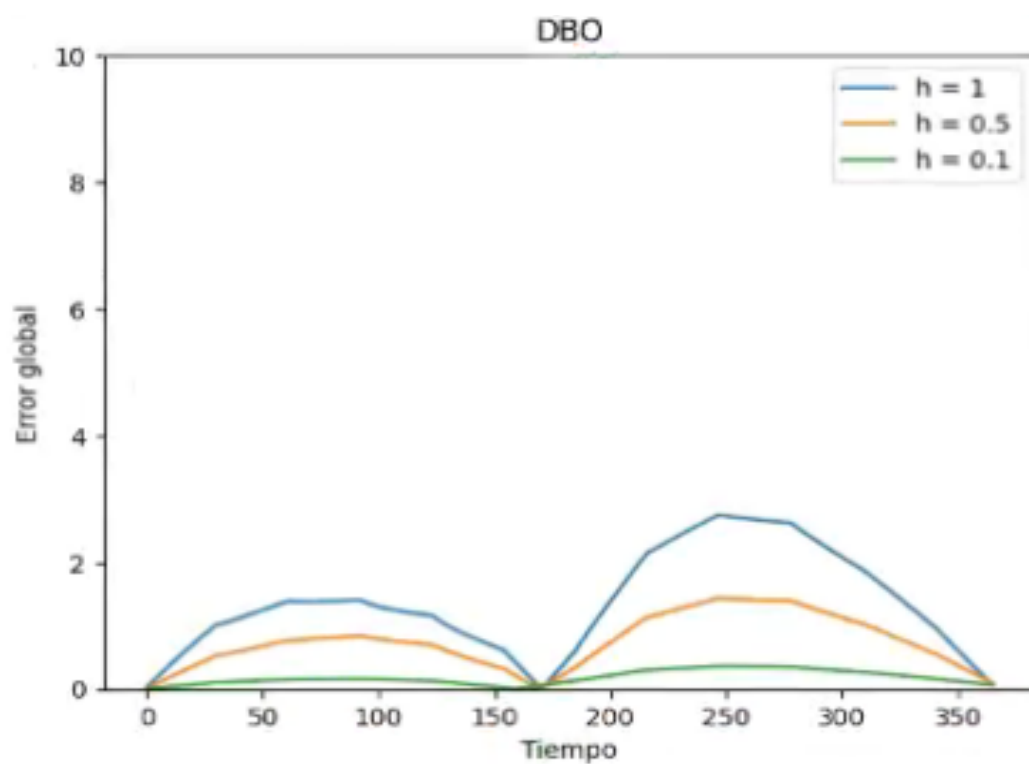
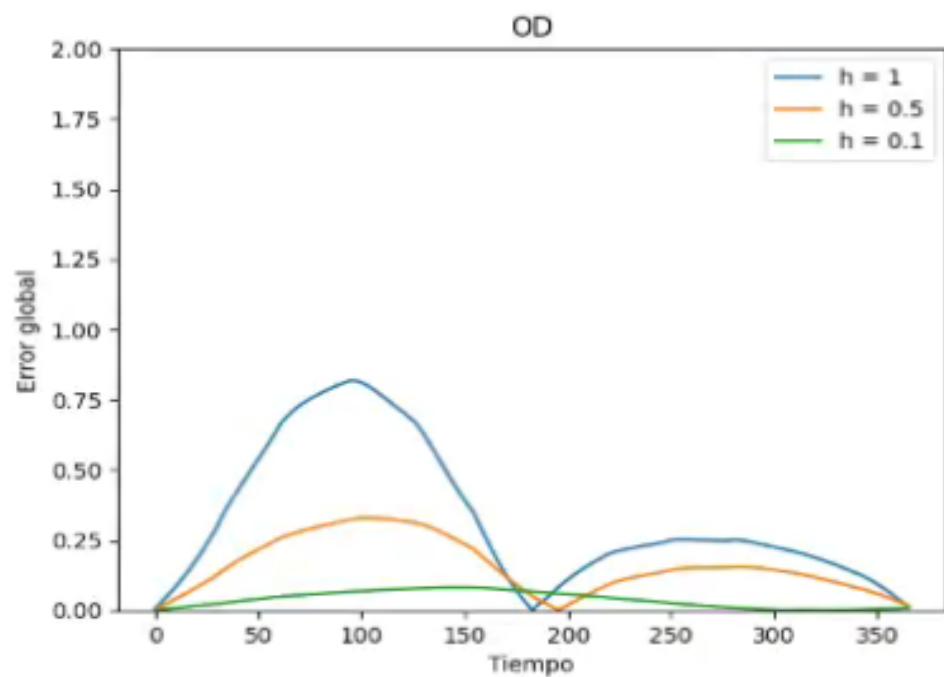
Observamos que el comportamiento es cíclico, ya que los gráficos no muestran un cambio significativo al pasar el tiempo. Debido a que las concentraciones cambiaron muy poco.

Error de truncamiento con paso de discretización temporal diario

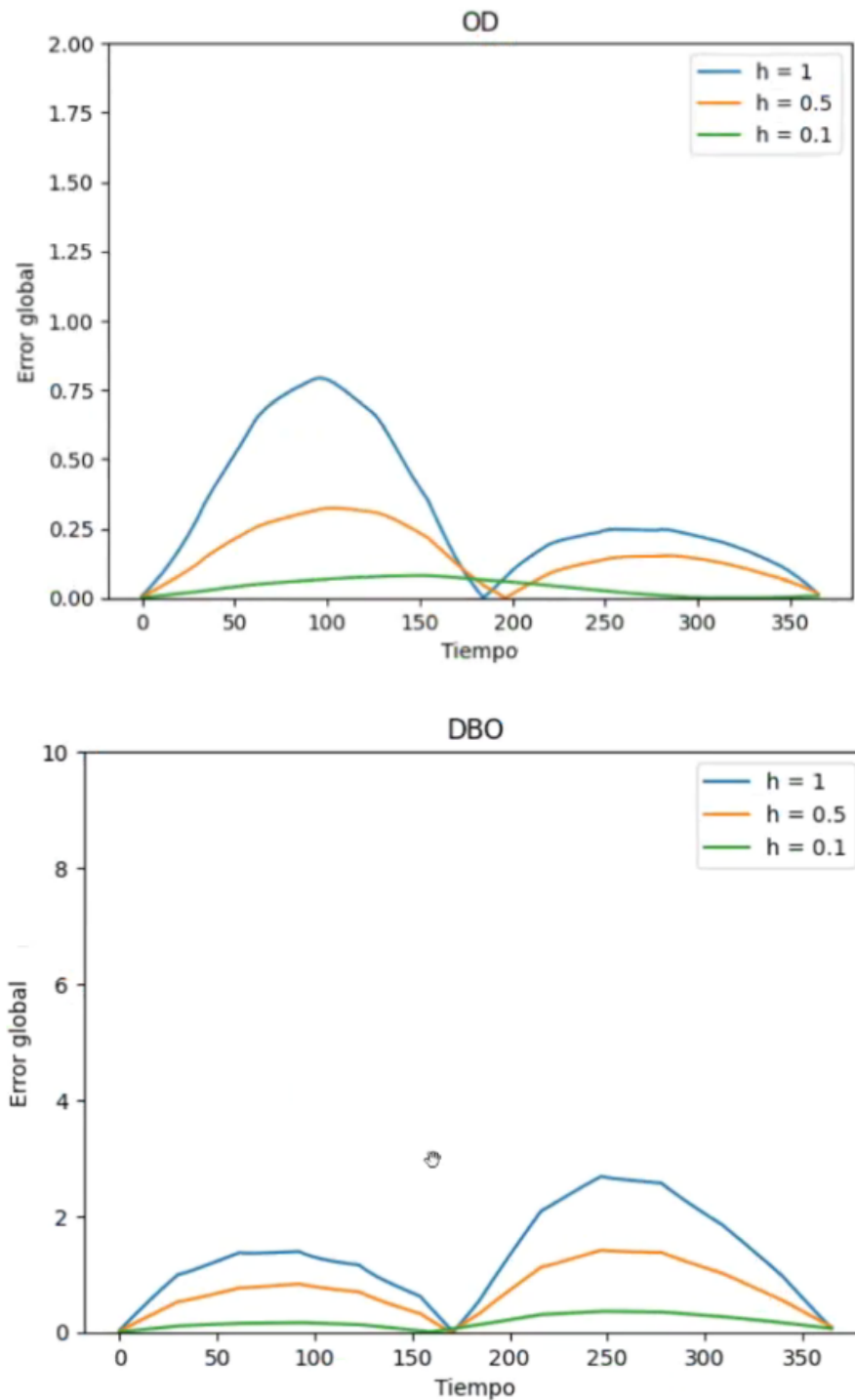
Para calcular el error de truncamiento para el DBO y OD, tomaremos un h muy pequeño (0.001) como "solución exacta" y calcularemos varias soluciones variando nuestro h (1, 0.5, 0.1) luego se lo restamos a nuestra "solución exacta" y obtendremos el error buscado.

Empezamos con el método de euler explícito:

Método Euler Explícito:



Método Punto Medio:

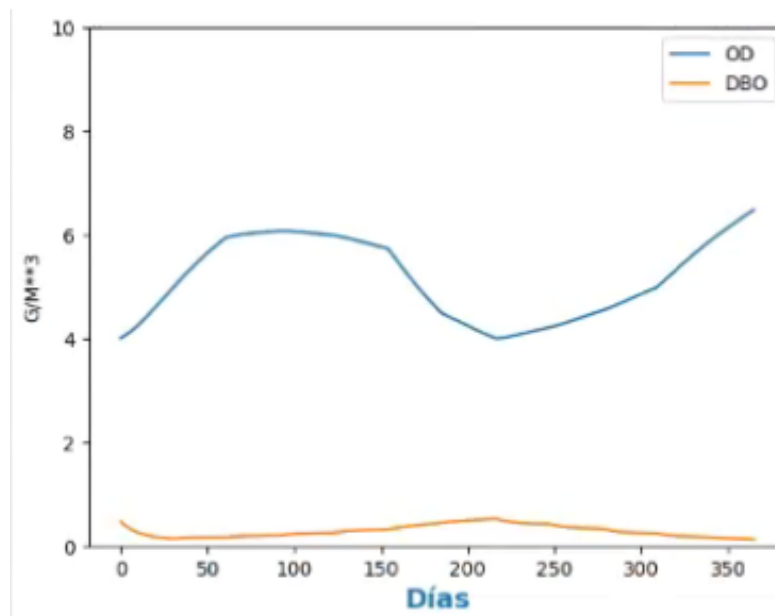


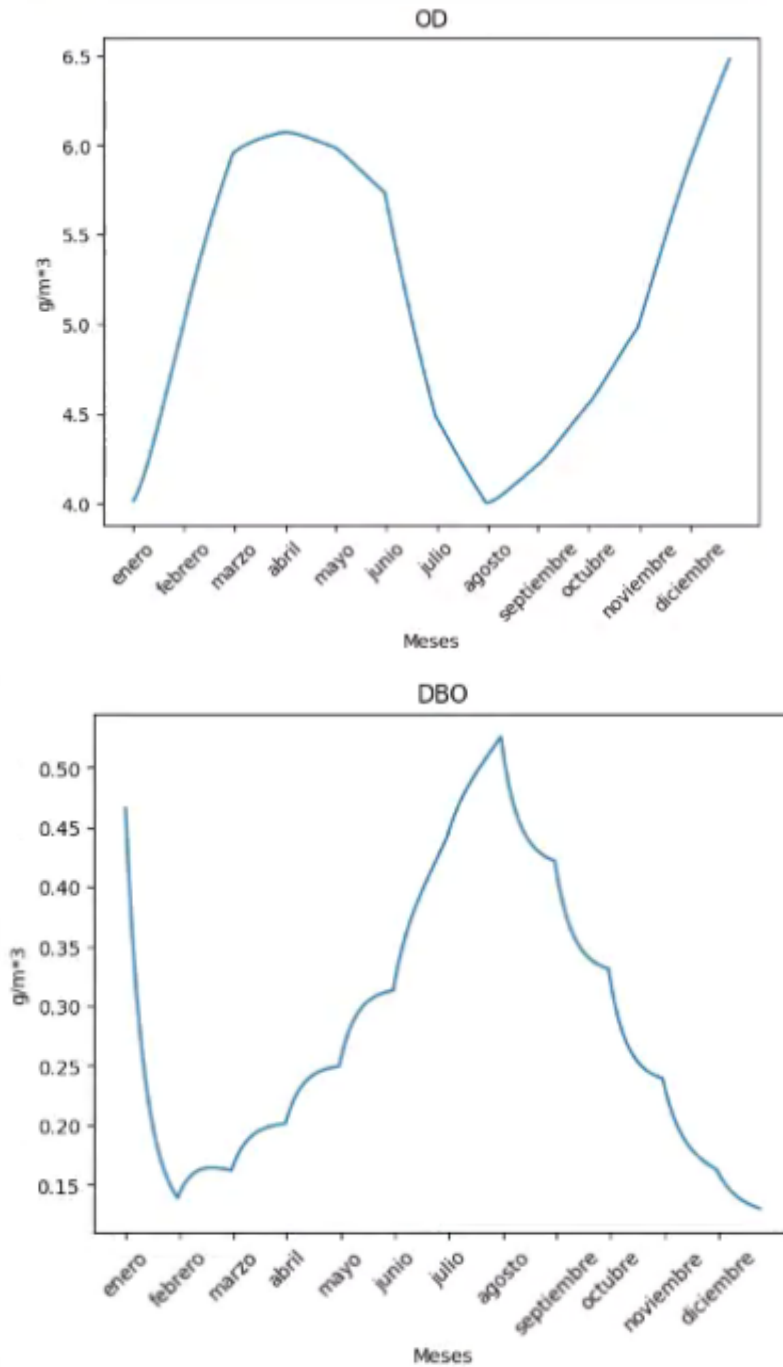
Como se puede observar ambos métodos dieron resultados de errores muy similares. Se puede observar que a medida que tomamos un h más pequeño, el error es cada vez menor. Al estar trabajando con métodos explícitos el error se vuelve más inestable a medida que se aumenta el paso de discretización.

Por lo que entonces, si tomamos un h muy pequeño, el error es menos inestable (además de ser mas pequeño)

Reducción del valor de DBO para aumentar el de OD

Para garantizar que los valores de OD no estén por debajo de 4 g/m³ redujimos el DBOe inicial de la laguna a 5.4 g/m³, es decir, en un 73% y el DBOi a 0.5 g/m³, es decir, en un 80.16%. Utilizando ese valor obtuvimos mediante la función min() de python el valor mínimo de OD encontrado es de 4.002 g/m³. Como observamos en anteriores gráficos, el DBO era muchísimo mayor que el OD (ya que estamos trabajando sobre una laguna que muy probablemente está contaminada) por lo que tuvimos que reducir muchísimo el DBO para llegar al resultado obtenido. A continuación mostramos los gráficos:



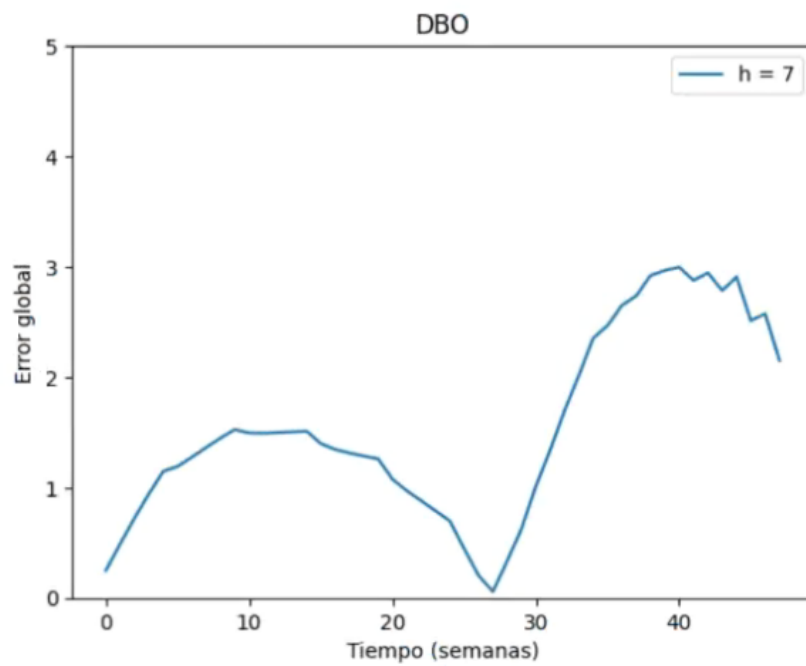
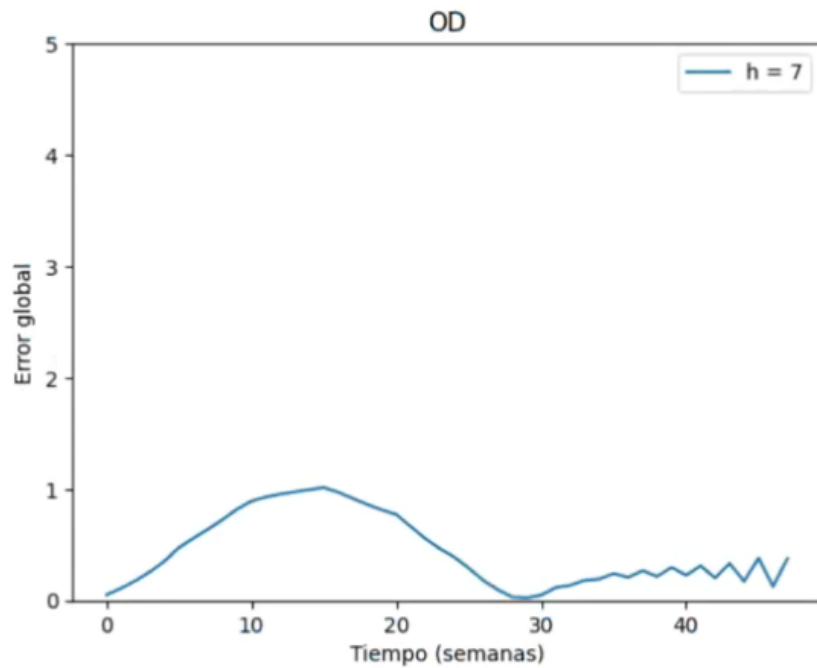


Se puede ver que, al igual que previo al cambio de DBO, a medida que aumenta este mismo, el OD disminuye y viceversa.

Euler explícito con paso de discretización semanal. Calculo de error de truncamiento

Para realizar este punto modificamos el valor de h a 7 días en las funciones que calculan el volumen, OD y DBO y así obtener los valores de DBO y OD con un paso de discretización semanal en vez de diario. Como ahora estamos tomando pasos muy grandes a

comparación de como hicimos durante el resto del tp, esperamos que el error sea muy inestable a comparación de los otros gráficos. Mostramos los errores tomando $h = 7$:



Como esperábamos, los gráficos no tienen sentido. eso se debe a que estamos tomando un h demasiado grande y mientras mas grande sea h mas error tendremos (además de que la inestabilidad aumenta como se ve en dichos gráficos).

Conclusiones

¿Por que cuando graficamos los errores de dbp dan tan parecido los métodos Euler explícito y Punto medio?

Estamos trabajando con 2 métodos explícitos, que al parecer ambos se comportan de manera inestable cuando calculamos el error de DBO, si bien notamos que al tomar h más pequeños el error baja, nunca llega a comportarse de manera estable.

Concluimos que estamos teniendo problemas con la estabilidad de los métodos en esos puntos, además de posibles errores en el código implementado.

(Revisamos el código varias veces y no encontramos nada que esté fuera de lo normal, así que intuimos que el error puede deberse a la inestabilidad de los métodos propuestos).

¿Por que tiene estos gráficos el error en el punto F?

Estamos trabajando con métodos explícitos, de orden 1 y 2 respectivamente, como anticipamos después de realizar los gráficos el error aumento y no solo eso, si no que observamos que tomar un paso muy grande vuelve muy inestable y hace que crezcan muchísimo los errores.

¿La laguna está contaminada?

Si bien al corregir el volumen y los valores iniciales de DBO y OD notamos que hay una diferencia bastante menos, el DBO sigue siendo mucho más grande que el OD por lo que podemos decir que la laguna muy probablemente está contaminada (no es lo unico el OD y DBO pero son los datos que tenemos)

¿Los métodos funcionaron de manera correcta?

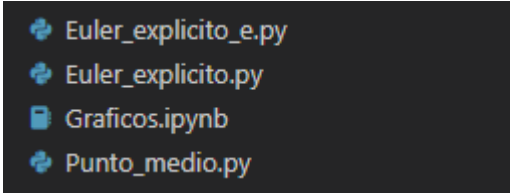
Corrigiendo los datos podemos ver que ahora los errores/graficos tienen más sentido y tanto DBO como OD tienen errores “similares” por lo que podemos decir que los métodos cumplieron su tarea.

Anexo 1

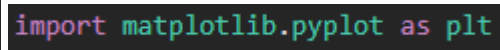
Código fuente

1.1 Estructura de carpetas

Para tener una idea general empezamos mostrando las bibliotecas y archivos utilizados en nuestro Tp



- Euler_explicito_e.py
- Euler_explicito.py
- Graficos.ipynb
- Punto_medio.py



```
import matplotlib.pyplot as plt
```

1.2 Código fuente

1.2.1 Euler_explicito.py

```

#Definimos las constantes
kbd0 = 0.1
ka = 0.01
ko2 = 1.4
DBOe = 20
ODe = 2
ODs = 9
Vi = 235*(10**6) #Pasamos el volumen iniciar a m**3

def calcular_volumen_euler_explicito(Vi:float,h:int,cant_de_iter:int,lista_Qe:list,lista_Qs:list):
    lista = []
    j = 0
    contador = 0

    V_i_mas_1 = Vi + h*(lista_Qe[j]-lista_Qs[j])
    lista.append(V_i_mas_1)

    for i in range(cant_de_iter):
        Vi = V_i_mas_1
        V_i_mas_1 = Vi + h*(lista_Qe[j]-lista_Qs[j])

        lista.append(V_i_mas_1)
        contador = contador + 1

        if contador == 31 :
            j = j + 1
            contador = 0

    return lista

```

```

def calcular_DBO_OD_euler_explicito(lista_de_volumen:list, h:int,lista_Qe:list, lista_Qs:list):
    DBO_i = 2.52
    OD_i = 0.80
    lista_DBO = []
    lista_OD = []
    j = 0
    contador = 0

    for i in range(len(lista_de_volumen)):
        V = lista_de_volumen[i]
        #Quitamos G ya que solo aplica en OD
        DBO_i_mas_uno = DBO_i + h*( ((lista_Qe[j]*DBOe) / V ) - ( (lista_Qs[j]*DBO_i) / V ) - kbd0*DBO_i*( (OD_i**2) / (OD_i**2 + ko2) ) )
        #Volvemos a escribir G
        OD_i_mas_uno = OD_i + h*( ( (lista_Qe[j]*ODe)/V ) - ( (lista_Qs[j]*OD_i)/V ) ) - kbd0 * ( ( (OD_i**2) *(DBO_i) ) / ( (OD_i**2 + ko2) ) )
        + ka*(ODs- OD_i) )

        lista_DBO.append(DBO_i_mas_uno)
        lista_OD.append(OD_i_mas_uno)

        DBO_i = DBO_i_mas_uno
        OD_i = OD_i_mas_uno

        contador = contador + 1

        if contador == 31:
            j = j + 1
            contador = 0

    return lista_DBO, lista_OD

```

```
def calcular_error(lista_Dbo_sol_exacta:list, lista_Od_sol_exacta:list , lista_Dbo_h:list, lista_Od_h:list):
    lista_e_od = []
    lista_e_dbo = []

    for i in range(len(lista_Od_sol_exacta)):
        e_od = abs(lista_Od_sol_exacta[i] - lista_Od_h[i])
        e_dbo = abs(lista_Dbo_sol_exacta[i] - lista_Dbo_h[i])

        lista_e_od.append(e_od)
        lista_e_dbo.append(e_dbo)

    return lista_e_dbo, lista_e_od
```

1.2.2 Euler_explicito_e.py

Este programa es igual al anterior pero cambiamos la constante DBOe para realizar el punto e.

```
#Definimos las constantes
kbd0 = 0.1
ka = 0.01
ko2 = 1.4
DBOe = 5.4 #Reducimos 96.5% el porcentaje de DBO
ODe = 4 #Cambiamos nuestro ODe por 4 ya que queremos mantener 4 durante todo el tramo
ODs = 9
Vi = 235*(10**6) #Pasamos el volumen iniciar a m**3
```

1.2.3 Punto_medio.py

```
kbd0 = 0.1
ka = 0.01
ko2 = 1.4
DBOe = 20
ODe = 2
ODs = 9
Vi = 235*(10**6) #Pasamos el volumen iniciar a m**3

def calcular_volumen_punto_medio(Vi:float,h:int,cant_de_iter:int,lista_Qe:list,lista_Qs:list):

    lista = []
    j = 0
    contador = 0

    V_i_mas_1_medio = Vi + (h/2)*(lista_Qe[j]-lista_Qs[j])
    V_i_mas_1 = Vi + (h)*(lista_Qe[j]-lista_Qs[j])

    lista.append(V_i_mas_1)

    for i in range(cant_de_iter):
        Vi = V_i_mas_1

        V_i_mas_1_medio = Vi + (h/2)*(lista_Qe[j]-lista_Qs[j])
        V_i_mas_1 = Vi + (h)*(lista_Qe[j]-lista_Qs[j])

        lista.append(V_i_mas_1)
        contador = contador + 1

        if contador == 31 :
            j = j + 1
            contador = 0

    return lista
```

```
def calcular_error(lista_Dbo_sol_exacta:list, lista_Od_sol_exacta:list , lista_Dbo_h:list, lista_Od_h:list):
    lista_e_od = []
    lista_e_dbo = []

    for i in range(len(lista_Od_sol_exacta)):
        e_od = abs(lista_Od_sol_exacta[i] - lista_Od_h[i])
        e_dbo = abs(lista_Dbo_sol_exacta[i] - lista_Dbo_h[i])

        lista_e_od.append(e_od)
        lista_e_dbo.append(e_dbo)

    return lista_e_dbo, lista_e_od
```

```
def calcular_OD_DBO_punto_medio(lista_de_volumen:list, h:int, lista_Qe:list, lista_Qs:list):
    DBO_i = 2.52
    OD_i = 0.80
    lista_DBO = []
    lista_OD = []
    j = 0
    contador = 0

    for i in range(len(lista_de_volumen)):
        V = lista_de_volumen[i]

        DBO_i_1_2 = DBO_i + (h/2)*(( (lista_Qe[j]*DBOe) / V ) - ( (lista_Qs[j]*DBO_i) / V ) - kbd0*DBO_i*(
            (OD_i**2) / (OD_i**2 + ko2) ))
        DBO_i_mas_uno = DBO_i + h*(( (lista_Qe[j]*DBOe) / V ) - ( (lista_Qs[j]*DBO_i_1_2) / V ) - kbd0*DBO_i_1_2*(
            (OD_i**2) / (OD_i**2 + ko2) ))

        OD_i_1_2 = OD_i + (h/2)*(( (lista_Qe[j]*ODe)/V ) - ((lista_Qs[j]*OD_i)/V) ) - kbd0 * (
            ((OD_i)**2) * (DBO_i) / ( (OD_i)**2 + ko2 ) ) + ka*(ODs- OD_i) )
        OD_i_mas_uno = OD_i + h*(( (lista_Qe[j]*ODe)/V ) - ((lista_Qs[j]*OD_i_1_2)/V) ) - kbd0 * (
            ((OD_i_1_2)**2) * (DBO_i) / ( (OD_i_1_2)**2 + ko2 ) ) + ka*(ODs- OD_i_1_2) )

        lista_DBO.append(DBO_i_mas_uno)
        lista_OD.append(OD_i_mas_uno)

        DBO_i = DBO_i_mas_uno
        OD_i = OD_i_mas_uno

        contador = contador + 1

    if contador == 31:
        j = j + 1
        contador = 0

    return lista_DBO, lista_OD
```

1.2.4 Graficos.ipynb

Punto C

Datos de OD y DBO

```
lista_OD = [0.8073433802899077, 0.8147218114627349, 0.8221597381521907, 0.8296776008045035, 0.8372926760304873, 0.8450197403891221, 0.8528715953535987, 0.86085940.3315972717288502, 0.3314242655358883, 0.3312585702352353, 0.33110005415133853, 0.3309486074725213, 0.33080413144252363, 0.33066653356198117, 0.3305357254365786]

lista_DBO = [2.4849632146637823, 2.449947898250844, 2.4149792272859223, 2.38007861605075, 2.3452645169160795, 2.310553052508507, 2.2759585156973627, 2.24149376591.448057031125559, 1.4364629146204193, 1.424539681966948, 1.4123409401307034, 1.3999136860679047, 1.387299190750952, 1.3745337627841983, 1.361649407540901, 1.34861649407540901]

meses = ['enero', 'febrero', 'marzo', 'abril', 'mayo', 'junio', 'julio', 'agosto', 'septiembre', 'octubre', 'noviembre', 'diciembre']
import matplotlib.pyplot as plt

plt.plot(range(1, len(lista_OD) + 1), lista_OD)
plt.xticks(range(1, len(lista_OD) + 1, 31), meses) # Configurar los meses en el eje x
plt.xlabel('Meses')
plt.ylabel('g/m3')
plt.title('OD')
plt.xticks(range(1, len(lista_OD) + 1, 31), meses, rotation=45)
plt.show()
```

```
#DBO
meses = ['enero', 'febrero', 'marzo', 'abril', 'mayo', 'junio', 'julio', 'agosto', 'septiembre', 'octubre', 'noviembre', 'diciembre']
import matplotlib.pyplot as plt

plt.plot(range(1, len(lista_DBO) + 1), lista_DBO)
plt.xticks(range(1, len(lista_DBO) + 1, 31), meses) # Configurar los meses en el eje x
plt.xlabel('Meses')
plt.ylabel('g/m3')
plt.title('DBO')
plt.xticks(range(1, len(lista_DBO) + 1, 31), meses, rotation=45)
plt.show()
```

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
temperaturas = {'OD': lista_OD, 'DBO': lista_DBO}
ax.plot(temperaturas['OD'], label = 'OD')
ax.plot(temperaturas['DBO'], label = 'DBO')
ax.legend(loc = 'upper right')
ax.set_xlabel("Días", fontdict = {'fontsize':14, 'fontweight':'bold', 'color':'tab:blue'})
ax.set_ylabel("G/M3")
ax.set_ylim([-2,40])
plt.show()
```

Punto E

```
lista_OD_e = [4.012827396144386, 4.02861997781436, 4.047062764516395, 4.067871234311972, 4.0907882035853, 4.115581062582763, 4.142039321792079, 4.1699725.9828643356048044, 5.976573682867025, 5.9698105421237875, 5.962641263563478, 5.955125087775184, 5.947314896296163, 5.93925786734016, 5.9309960646266555, 5.922837345734573]

lista_DBO_e = [0.46590893881790957, 0.4349045692302087, 0.40670159222616867, 0.38104201749261585, 0.3576923474137197, 0.33644108635052966, 0.3170965325820.24097840359024303, 0.24039609435535741, 0.2398645477236469, 0.23937889904062495, 0.23299125442272628, 0.2271455436690715, 0.22179155838586417, 0.216881455436690715]

#ENERO
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
medidas = {'OD': lista_OD_e, 'DBO': lista_DBO_e}
ax.plot(medidas['OD'], label = 'OD')
ax.plot(medidas['DBO'], label = 'DBO')

ax.legend(loc = 'upper right')
ax.set_xlabel("Días", fontdict = {'fontsize':14, 'fontweight':'bold', 'color':'tab:blue'})
ax.set_ylabel("G/M3")
ax.set_ylim([0,10])
plt.show()
```

Graficamos OD y DBO por separado

```
meses = ['enero', 'febrero', 'marzo', 'abril', 'mayo', 'junio', 'julio', 'agosto', 'septiembre', 'octubre', 'noviembre', 'diciembre']
import matplotlib.pyplot as plt

plt.plot(range(1, len(lista_OD_e) + 1), lista_OD_e)
plt.xticks(range(1, len(lista_OD_e) + 1, 31), meses) # Configurar los meses en el eje x
plt.xlabel('Meses')
plt.ylabel('g/m3')
plt.title('OD')
plt.xticks(range(1, len(lista_OD_e) + 1, 31), meses, rotation=45)
plt.show()
```

✓ 0.8s

```
meses = ['enero', 'febrero', 'marzo', 'abril', 'mayo', 'junio', 'julio', 'agosto', 'septiembre', 'octubre', 'noviembre', 'diciembre']
import matplotlib.pyplot as plt

plt.plot(range(1, len(lista_DBO_e) + 1), lista_DBO_e)
plt.xticks(range(1, len(lista_DBO_e) + 1, 31), meses) # Configurar los meses en el eje x
plt.xlabel('Meses')
plt.ylabel('g/m3')
plt.title('DBO')
plt.xticks(range(1, len(lista_DBO_e) + 1, 31), meses, rotation=45)
plt.show()
```

✓ 0.9s

Punto D

Graficamos los errores para OD y DBO con euler explícito

```
lista_error_h_1 = [0.007336027208275975, 0.014707105283943034, 0.022137678860682408, 0.029648188384694962, 0.037255910466766906, 0.04497562166585045, 0.05282012345
0.7282417594898956, 0.7223604450232807, 0.7164921863617557, 0.7106454667139428, 0.7048276787003134, 0.6990452416952097, 0.6933037072038842, 0.6876078534460449, 0.6818119781175

lista_error_h_0_5 = [0.003666762157069736, 0.007339843673710544, 0.011022503655753635, 0.014717726348262516, 0.018428250316869677, 0.022156594519625705, 0.025905
0.15065450543138115, 0.14452236643618055, 0.13843451637659554, 0.13239530945290168, 0.12640860811737598, 0.12047781778535727, 0.11460591980738954, 0.108795502746
0.15229787581309573, 0.15226513046645918, 0.1532965500623189, 0.15405394392070249, 0.15457257638885835, 0.15488303264920733, 0.15501182914244938, 0.1549819781175

lista_error_h_0_1 = [0.000727858751288446, 0.0014558921093200672, 0.0021841274722508297, 0.002912591766006778, 0.0036413114544011638, 0.004370312549034527, 0.005
0.05901311476697679, 0.05849703942236584, 0.05796989341193426, 0.05743194387534378, 0.05688345361371405, 0.05632468114071687, 0.05575588073345861, 0.055177302483
```

✓ 0.3s

OD

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
medidas = {'OD': lista_error_h_1, 'OD_1': lista_error_h_0_5, 'OD_2': lista_error_h_0_1}
ax.plot(medidas['OD'], label = 'h = 1')
ax.plot(medidas['OD_1'], label = 'h = 0.5')
ax.plot(medidas['OD_2'], label = 'h = 0.1')
plt.title('OD')
ax.legend(loc = 'upper right')
ax.set_xlabel("Tiempo")
ax.set_ylabel("Error global")
ax.set_ylim([0,2])
plt.show()
```

✓ 1.0s

DBO

```
lista_DBO_h_1 = [0.03500184556429975, 0.06998222220678763, 0.10491595340275417, 0.1397816248704955, 0.17456078423928734, 0.20923730888256165, 0.24379690  
1.1064599525188998, 1.1188743148969074, 1.1314759174201452, 1.144228451484451, 1.1570999117167569, 1.1700620272989026, 1.1830897714907627, 1.19616093867  
0.4643955003512308, 0.4277010447143925, 0.3909762610337939, 0.35416340269823854, 0.31721378628516517, 0.28008679150942273, 0.242748937343908, 0.20517303  
lista_DBO_h_2 = [0.017459201958714754, 0.03491553981133233, 0.05236567486705335, 0.06980652715840963, 0.08723524763353563, 0.10464919331202927, 0.122045  
0.7180146551576843, 0.7144708718136976, 0.7109801507036142, 0.7075394451300883, 0.7041459079507884, 0.7007968788643362, 0.6974898724740226, 0.6819464138  
1.1421510255790435, 1.131215297026655, 1.120327222190061, 1.1094848790283462, 1.0986865801288057, 1.0879308429907835, 1.0772163641363992, 1.066541996546
```

```
lista_DBO_h_3 = [0.0034600007224101503, 0.006919966957491752, 0.010379870746527686, 0.013839684575887912, 0.017299381367383848, 0.020758934468830592, 0.  
0.0643343916879715]
```

✓ 0.3s

```
import matplotlib.pyplot as plt  
fig, ax = plt.subplots()  
medidas = {'DBO_1': lista_DBO_h_1, 'DBO_2': lista_DBO_h_2, 'DBO_3': lista_DBO_h_3} # 'OD_1': lista_error_h_0_5 , 'OD_2': lista_error_h_0_1  
ax.plot(medidas['DBO_1'], label = 'h = 1')  
ax.plot(medidas['DBO_2'], label = 'h = 0.5')  
ax.plot(medidas['DBO_3'], label = 'h = 0.1')  
plt.title('DBO')  
ax.legend(loc = 'upper right')  
ax.set_xlabel("Tiempo")  
ax.set_ylabel("Error global")  
ax.set_ylim([0,10])  
plt.show()
```

✓ 0.9s

Metodo Punto Medio

```
lista_h_1 = [0.006801576048246272, 0.013698583945475629, 0.020701708449815115, 0.027820402311025783, 0.035063138483730105, 0.042437613051
```

```
lista_h_2 = [0.0035330246462226267, 0.007080369552025956, 0.010644349407856302, 0.01422710372581959, 0.017830614977266457, 0.021456724866  
0.15903504609486252, 0.16358510007087712, 0.1680291245480502, 0.1723772654162996, 0.17663890042347974, 0.18082269926636096, 0.18493667905  
0.08372614791073985, 0.07840714245838454, 0.07315399714002502, 0.06796722559019497, 0.06284716897861875, 0.05779401164024989, 0.052807795
```

```
lista_h_3 = [0.0007225057613465813, 0.0014452531157344017, 0.00216826783469648, 0.0028915752531538663, 0.003615200278716979, 0.0043391674  
0.05851153989860114, 0.05798936630879681, 0.057456446798960115, 0.05691304888938094, 0.05635940389097838, 0.05579578694950804, 0.0552224
```

✓ 0.2s

```
fig, ax = plt.subplots()  
medidas = {'OD_1': lista_h_1, 'OD_2': lista_h_2, 'OD_3': lista_h_3 }  
#,'DBO_3': lista_DBO_h_3} # 'OD_1': lista_error_h_0_5 , 'OD_2': lista_error_h_0_1  
ax.plot(medidas['OD_1'], label = 'h = 1')  
ax.plot(medidas['OD_2'], label = 'h = 0.5')  
ax.plot(medidas['OD_3'], label = 'h = 0.1')  
#ax.plot(medidas['DBO_3'], label = 'h = 0.1')  
plt.title('OD')  
ax.legend(loc = 'upper right')  
ax.set_xlabel("Tiempo")  
ax.set_ylabel("Error global")  
ax.set_ylim([0,2])  
plt.show()
```

✓ 0.9s

DBO

```
lista_1 = [0.034452249479649755, 0.06882393572942869, 0.1031026201363825, 0.13727709950931555, 0.1713371463812572, 0.2052733002721303, 0.23907669975335333, 0.2727389472365003,
```

```
lista_2 = [0.01732199355203967, 0.03463353230682355, 0.051932132452459534, 0.06921546327021044, 0.08648135840957583, 0.10372778515636938, 0.12095283290055692, 0.13815470016327
```

```
lista_3 = [0.0034545190188863366, 0.0069089415345269245, 0.010363241032853132, 0.0138173913821662, 0.017271366864200832, 0.020725142157381082, 0.024178692328262752, 0.027631992  
0.357558547526438, 0.3578185885601189, 0.3564910864641533, 0.35597579147316294, 0.3554724588576872, 0.35498080488209951, 0.35450072639080513, 0.35403186135658826, 0.3535740281002  
0.11027425635914812, 0.1063754787945701, 0.10248447446472175, 0.0986011326572074, 0.0947252679669317, 0.09085681414723812, 0.08699563013281386, 0.0831415969363607, 0.079294598
```

✓ 0.2s

```
fig, ax = plt.subplots()  
medidas = {'DBO_1': lista_1, 'DBO_2': lista_2, 'DBO_3': lista_3} # 'OD_1': lista_error_h_0_5 , 'OD_2': lista_error_h_0_1  
ax.plot(medidas['DBO_1'], label = 'h = 1')  
ax.plot(medidas['DBO_2'], label = 'h = 0.5')  
ax.plot(medidas['DBO_3'], label = 'h = 0.1')  
plt.title('DBO')  
ax.legend(loc = 'upper right')  
ax.set_xlabel("Tiempo")  
ax.set_ylabel("Error global")  
ax.set_ylim([0,10])  
plt.show()
```

✓ 0.7s

Punto F

Gráficos de errores de DBO y OD con 47 semanas y contador hasta 5

```
lista_error_OD_h_7 = [0.05099464618397631, 0.10946479777419937, 0.17797465276074476, 0.2582424899719432, 0.3522997135401327, 0.47620885691396597, 0.5888888888888889]
```

✓ 0.0s

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
medidas = {'OD': lista_error_OD_h_7}
ax.plot(medidas['OD'], label = 'h = 7')
```

```
plt.title('OD')
ax.legend(loc = 'upper right')
ax.set_xlabel("Tiempo (semanas)")
ax.set_ylabel("Error global")
ax.set_ylim([0,5])
plt.show()
```

✓ 0.9s

```
lista_error_DBO_h_7 = [0.24923918521905986, 0.4915328614541985, 0.7236725254265104, 0.9431213792833064, 1.1469141977129915, 1.1907459351067775, 1.275880896328734]
```

✓ 0.0s

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
medidas = {'DBO': lista_error_DBO_h_7}
ax.plot(medidas['DBO'], label = 'h = 7')
```

```
plt.title('DBO')
ax.legend(loc = 'upper right')
ax.set_xlabel("Tiempo (semanas)")
ax.set_ylabel("Error global")
ax.set_ylim([0,5])
plt.show()
```


Anexo II

Corridas del Tp

Lo que haremos en este anexo será mostrar algunas corridas del tp, gráficos y resultados de DBO y OD

2.1 Euler_explicito.py

Este programa imprime dos listas con los 365 valores obtenidos mediante euler explicito de OD y DBO respectivamente, con un paso de discretización diario.

DBO:

```
[2.4849632146637823, 2.449947898250844, 2.4149792272859223, 2.380078616058075, 2.3452645169160795, 2.310553052508507, 2.275958515697362, 975005876, 2.0380372515398824, 2.0047727530655033, 1.9717162383061906, 1.9388762415027148, 1.9062612325421884, 1.8738796456594067, 1.8407005, 1.805445265104989, 1.77242423851163166, 1.7393417056323779, 1.70647590940078437, 1.67355030023548562, 1.6405818379966804, 1.60760884364629146204193, 1.574539681966948, 1.541423409401307034, 1.50839136860679047, 1.47537299190750952, 1.44235337627841983, 1.409361649407540901, 1.3764074838, 1.3436301562603, 1.310996919028987, 1.27844035894792831, 1.2461656149622576, 1.21392950789236828, 1.181805004222584664, 1.150133179108886134, 1.118163540466851, 1.1367185770722235, 1.1378928995685043, 1.1387300070030997, 1.1392687511722746, 1.13954381966159243457193, 1.1363079985265323, 1.135348032601199, 1.1343083625811412, 1.1331999700304518, 1.132032627496762, 1.1308150284350236, 1.111, 1.1214365800796426, 1.1200335241455093, 1.118625491957529, 1.1172150965844363, 1.115804643363036, 1.1143961631732342, 1.13074714742664638663863, 1.2165271166660212, 1.2257980981453873, 1.2345380793747711, 1.2428014784898782, 1.2506372075168033, 1.258089196898015,
```

OD:

```
[0.8073433802899077, 0.8147218114627349, 0.8221597381521907, 0.8296776008045035, 0.8372926760304873, 0.8450197403891221, 0.8528715953535987, 0.860835, 0.9031600160416151, 0.9121468928692885, 0.9213254952754235, 0.9307021917376704, 0.9402831872966261, 0.9500745628055408, 0.9600823048547602, 0.97359, 1.0250002131890714, 1.0366854909568544, 1.0486331897097876, 1.0608488574440915, 1.0733379730241157, 1.0861059332319338, 1.0991580380803665, 1.11742815, 1.1812720888715222, 1.1932445407188221, 1.2049333862972427, 1.2163940817221421, 1.2276749176309623, 1.2388179846556855, 1.2498599893243042030652547, 1.3155295337739392, 1.3265684466486503, 1.337674591845228, 1.3488590657024386, 1.3601316001955674, 1.3715007266236192, 1.38297391708716023160214435, 1.4543110039366463, 1.4666548609146866, 1.4755008682683504, 1.4837505831994195, 1.4914648442440506, 1.498698085661596, 1.5054989939134390403411578, 1.5393658328576691, 1.54413200931166, 1.548708745117149, 1.5531137428278774, 1.557362820647, 1.561470114016653, 1.5654482558101293, 1.570183, 1.5871602176951243, 1.5904916454791473, 1.593757025327214, 1.596960087916727, 1.6001076808352488, 1.6032080283950856, 1.6062437378817849, 1.6054325, 1.6185225668496954, 1.6176970853860275, 1.616124549462177, 1.6138947743789351, 1.6110885446716074, 1.607778434524171, 1.6040295671431453,
```

2.2 Euler_explicito_e.py

Este programa imprime el valor mínimo de OD para un DBO reducido

```
4.001837973336424
```

2.3 Punto_medio.py

Este programa imprime dos listas con los 365 valores obtenidos mediante punto fijo de OD y DBO respectivamente, con un paso de discretización diario.

DBO:

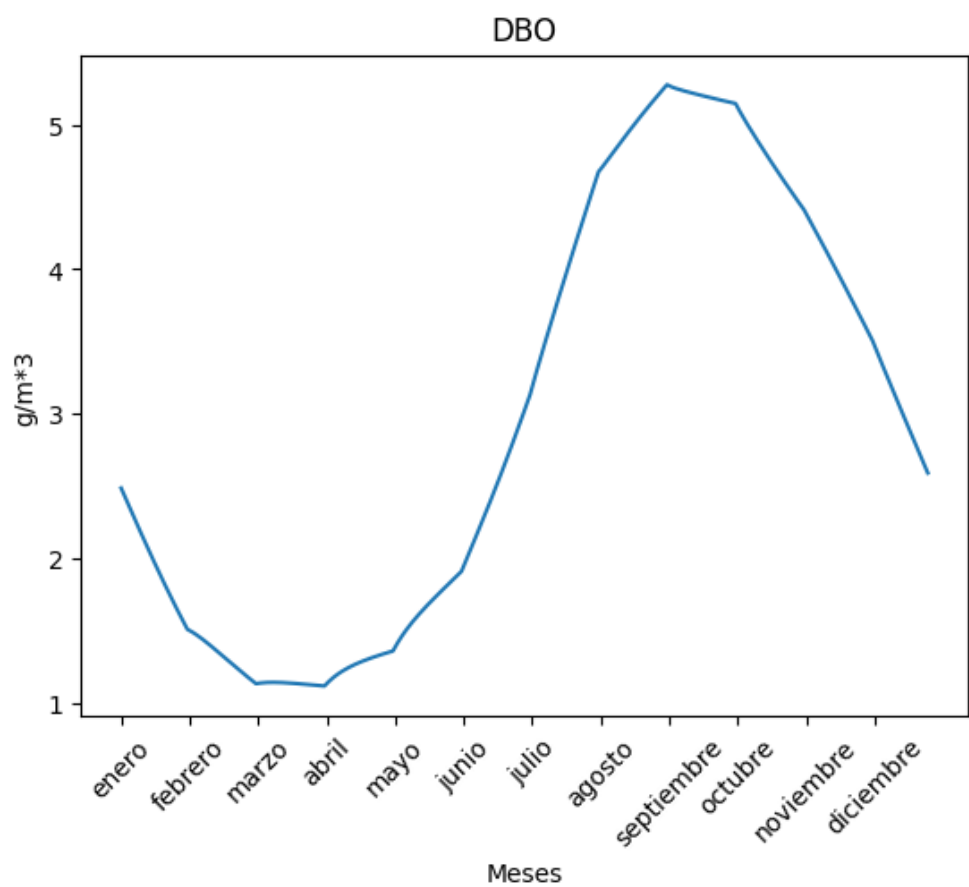
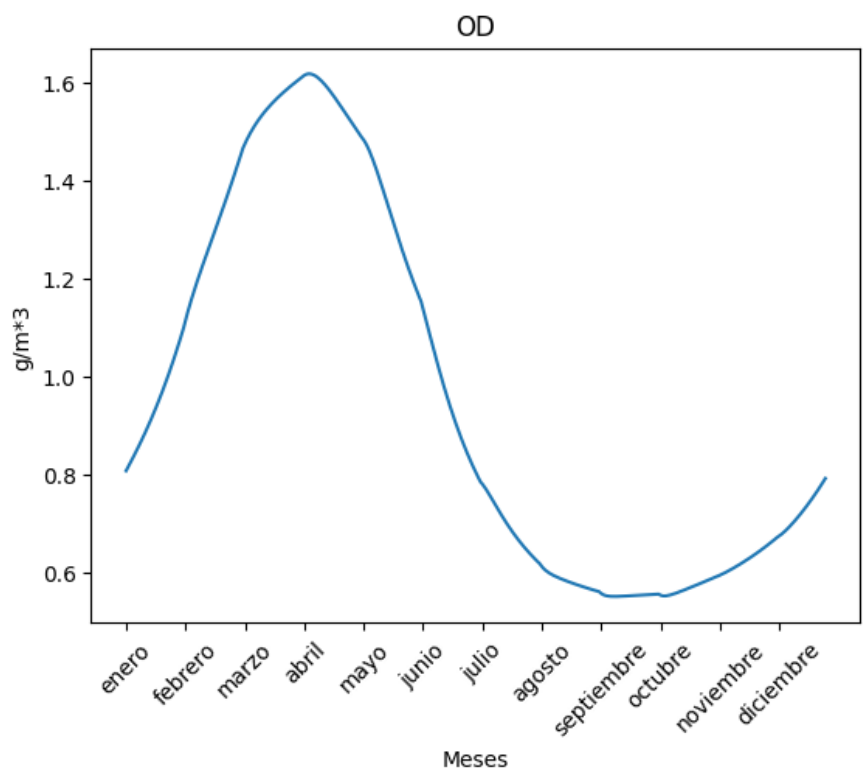
```
[18.66299638263628, 17.99812455539651, 17.675751035374656, 17.515722295139618, 17.43271441478456, 17.386861313290005, 17.359548979931837, 17.341752700753307, 17.328895202999373, 17.318570868142082, 17.309485334926347, 17.300932383001573, 17.292523888655165, 17.28404604038534, 17.27538128086355, 17.26646543547149, 17.257264005867565, 17.247759009571745, 17.237941657441485, 17.22780827396496, 17.217358022495727, 17.206591636315824, 17.195510710624884, 17.184117307494535, 17.17241373549837, 17.16040242685793, 17.14808586903926, 17.135466566757636, 17.122547020965282, 17.109329717323963, 17.095817119974004, 17.122560884250255, 17.146640827568422, 17.168944410970887, 17.189976631134105, 17.210024262802822, 17.229250900642818, 17.24775099961954, 17.265580314493057, 17.282772957155686, 17.29935092748853, 17.315329430197465, 17.330179838293902, 17.345531344284353, 17.359771880245987, 17.37344863080167, 17.38656831948929, 17.399137369019087, 17.411161991350642, 17.422648238711307, 17.433602032869693, 17.44402918229458, 17.453935392555596, 17.463326272944514, 17.472207340974087, 17.4805840256761, 17.488461670211542, 17.495845534078665, 17.502740795078267, 17.50915255112538, 17.515085821957406, 17.520545550767157, 17.440418992091548, 17.3607471431095, 17.281572444585393, 17.202916534263117, 17.12478963784159, 17.04719572220029, 16.970135323975434, 16.893607103094663, 16.817608696686964, 16.742137188838477, 16.667189369235686, 16.592761875682804, 16.518851272679576, 16.445454094758205, 16.37256687037195, 16.30018613507724, 16.228308438700815, 16.156930349326817, 16.086048455385736, 16.01565936676622, 15.945759715362158, 15.876346155309323, 15.8074153630
```

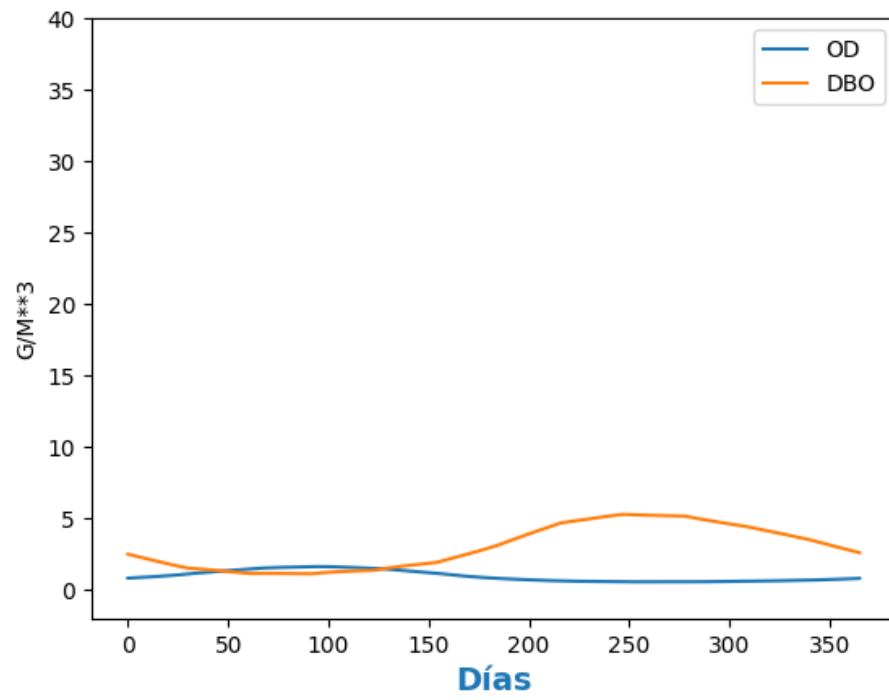
OD:

```
[0.9932951823491836, 0.651136170131185, 0.4835423510626018, 0.39393145181917016, 0.34551620649496535, 0.3191704767922949, 0.30471457880307484, 0.2967281322936338, 0.29229478424637406, 0.28982632153026094, 0.2884497009112597, 0.28768203542343523, 0.287255217246122, 0.28702002258400117, 0.2868933062187824, 0.2868286722059883, 0.2868002208784017, 0.2867934822235424, 0.28680036466703096, 0.2868163375863033, 0.2868388590394564, 0.28686649808088416, 0.2868984446831516, 0.28693423602504275, 0.2869736035878086, 0.2870163877194782, 0.28706248988587746, 0.28711184597705797, 0.287164411379612, 0.28722015262515793, 0.28727904271513666, 0.29011125619232264, 0.291553386944122, 0.2922218223390179, 0.2924619965177503, 0.29246685701596464, 0.2923439590497947, 0.2921531694090528, 0.2919277650764497, 0.29168621181459997, 0.29143873065547937, 0.29119095560137395, 0.29094596994754013, 0.2907054395147206, 0.2904702431147377, 0.29024082321391237, 0.2900173809190713, 0.289799984365994, 0.28958862894574056, 0.2893832707498398, 0.2891838451261434, 0.28899027695836677, 0.28880248634628297, 0.28862039173074483, 0.28844391159982, 0.2882729654075682, 0.28810747405639753, 0.2879473601379771, 0.2877925480410026, 0.2876429639859465, 0.2874985360201595, 0.28735919399182297, 0.2872009231967447, 0.2870725256664739, 0.286970556061615876, 0.29041738502396325, 0.2911175514740605, 0.2918113202258831, 0.29250158858984787, 0.29318995177150786, 0.2938772867183457, 0.2945640737643843, 0.2952505740056508, 0.2959369272167698, 0.29662320595977004, 0.2973094455157578, 0.2979956604597036, 0.29868185384847046, 0.2993680223195196, 0.3000541589234197, 0.3007402546991908, 0.30142629955151956,
```

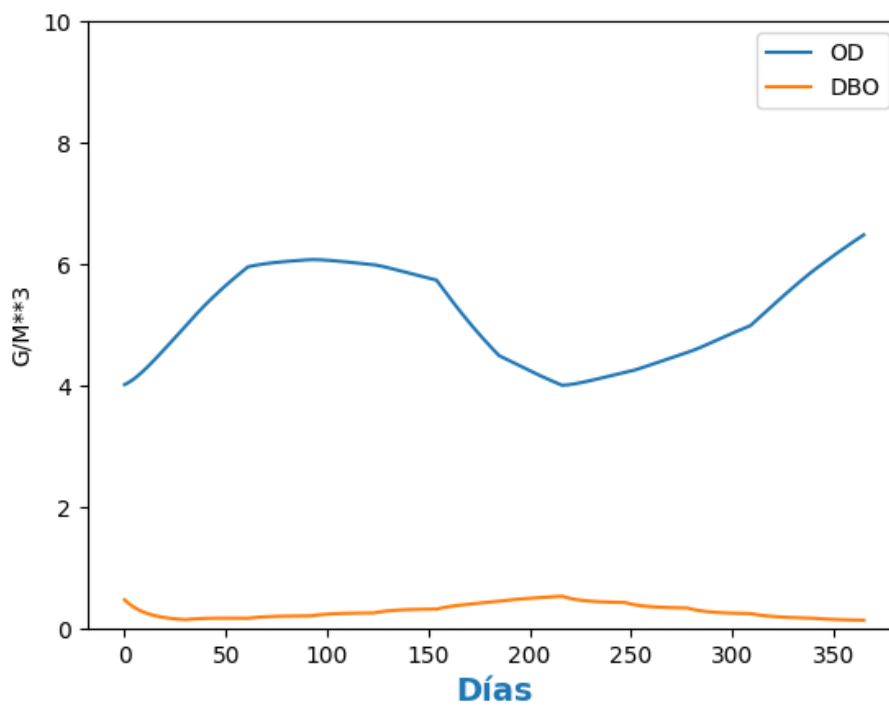
2.4 Graficos.ipynb

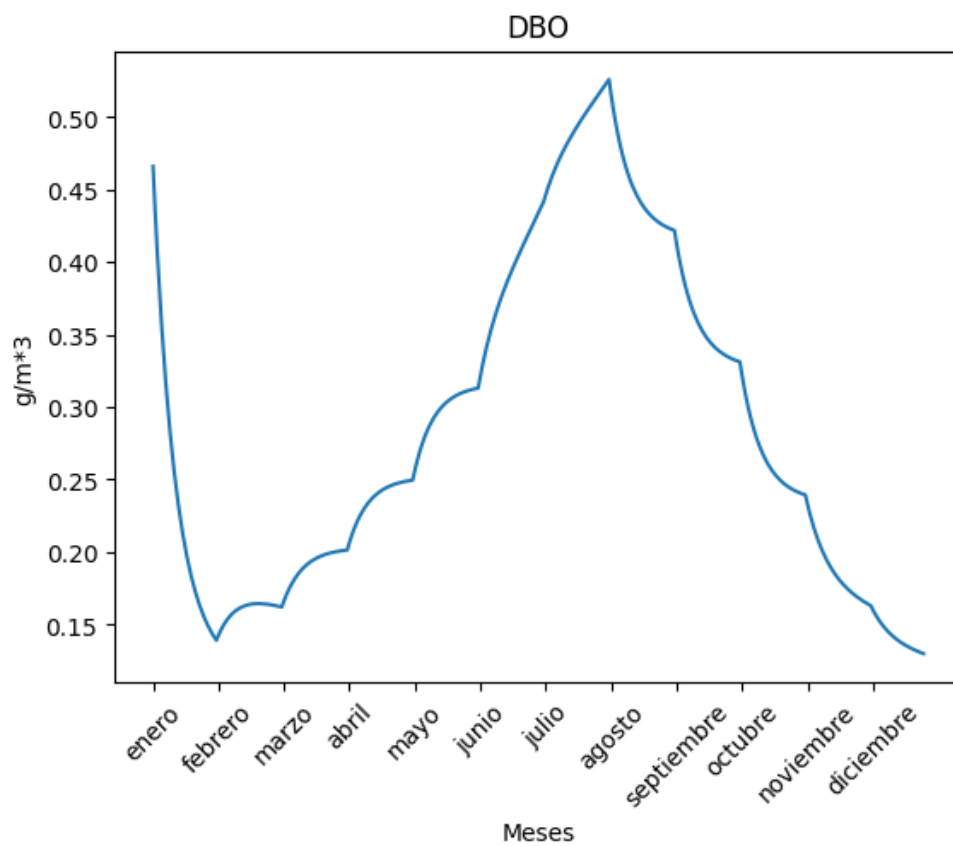
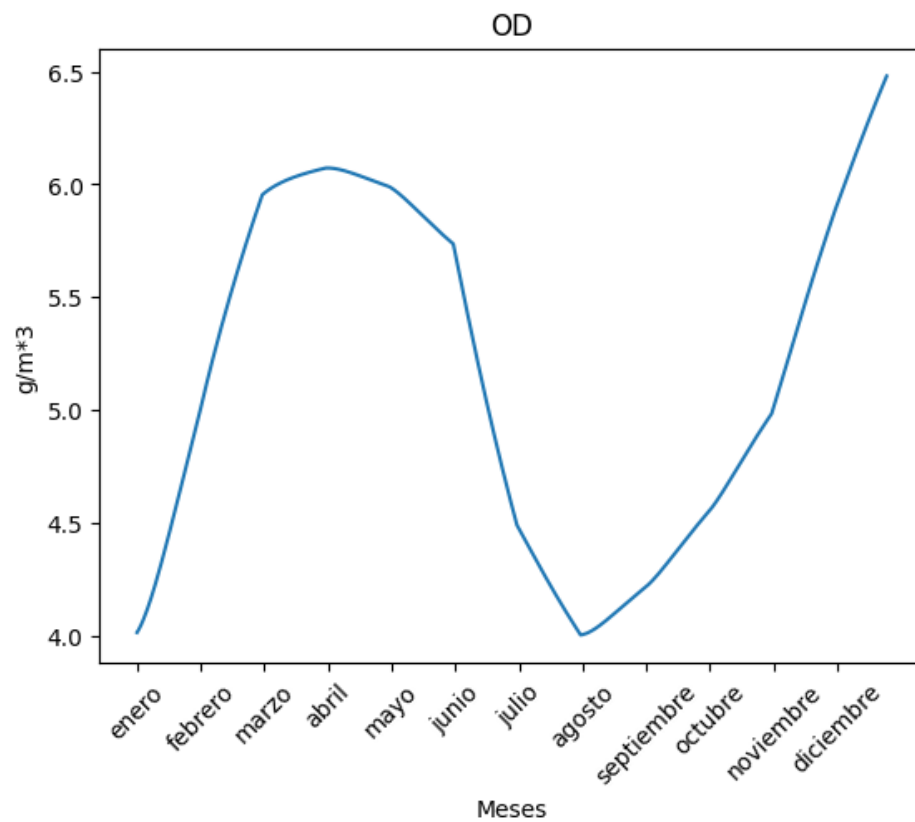
Punto C:



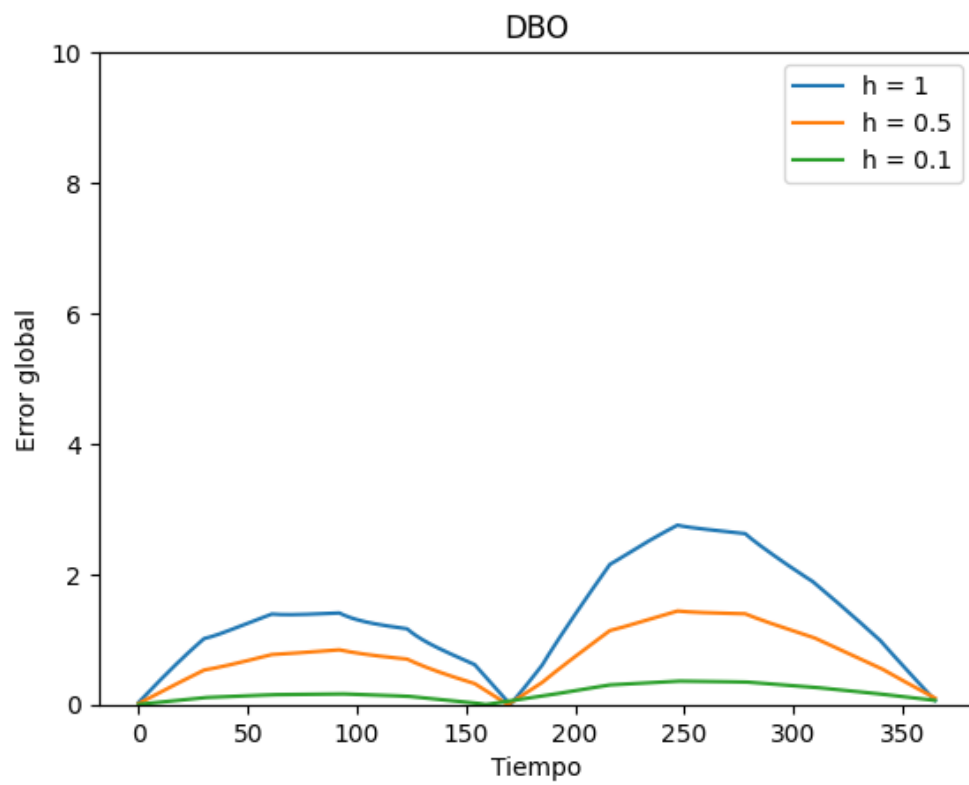
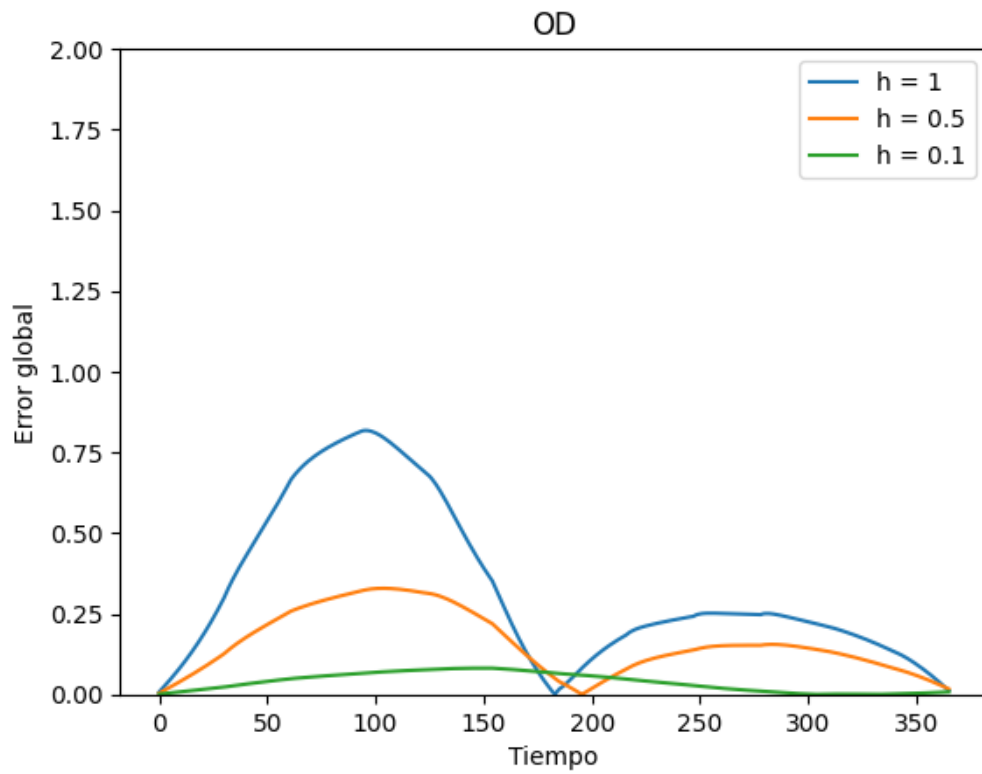


Punto E

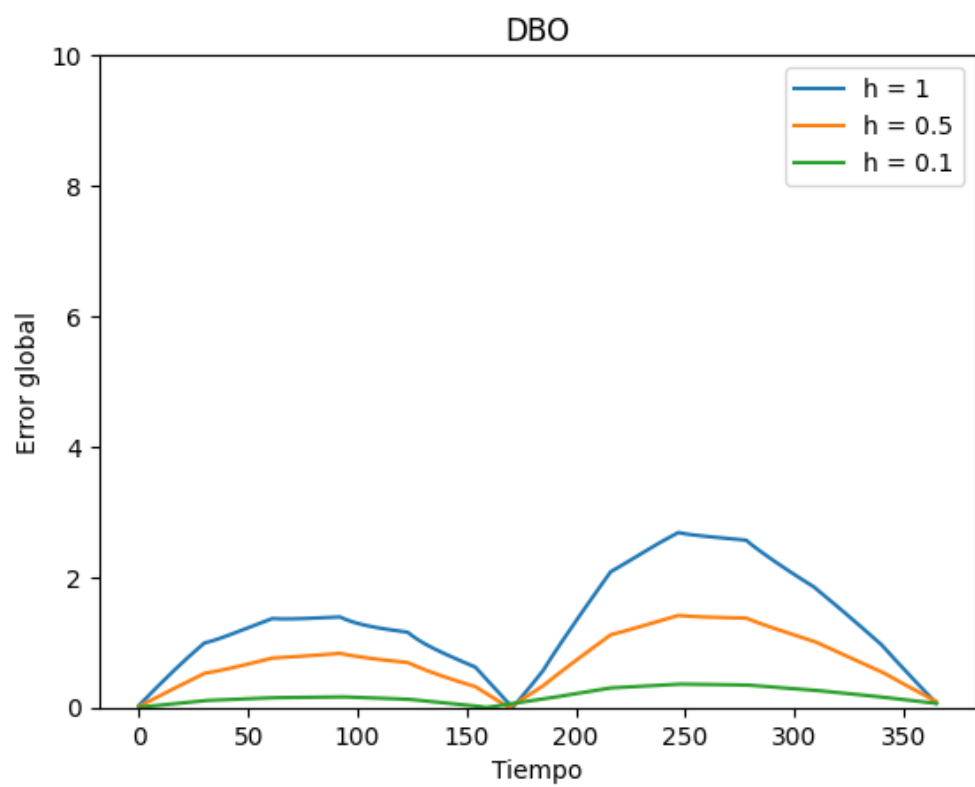
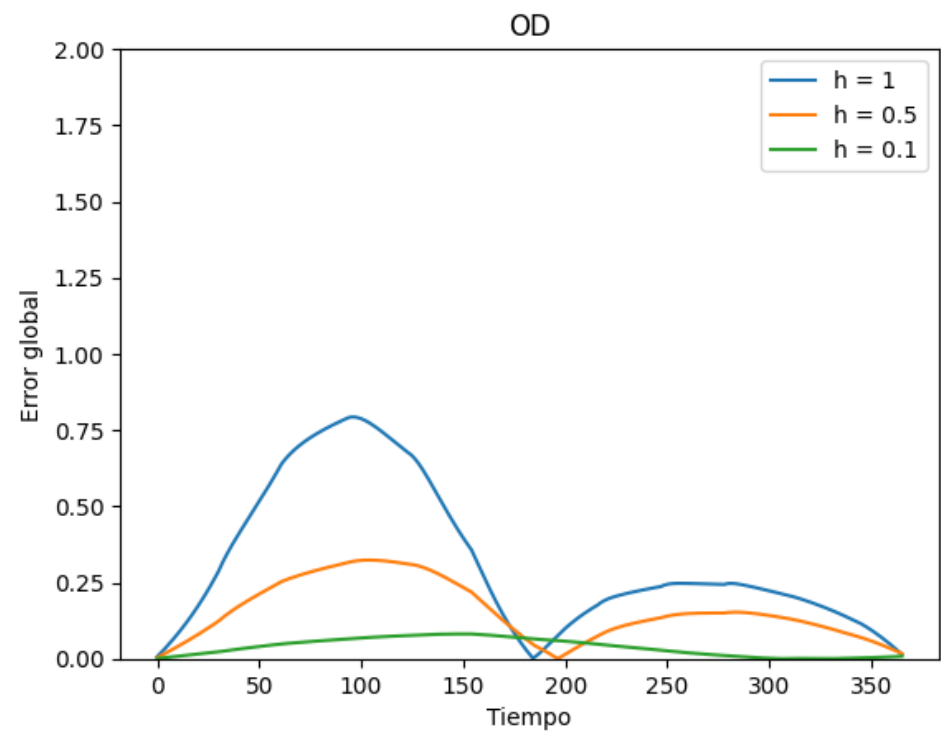




Punto D:
Euler explícito



Punto medio:



Punto F

