

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

**DATOS DEL TRABAJO PRÁCTICO**

<b>2</b>	<b>AÑO</b>	<b>2023</b>
	<b>1</b>	<b>Balance de Oxígeno Disuelto en la laguna Mar Chiquita</b>
<b>TP NRO</b>	<b>CUAT</b>	<b>TEMA</b>

**INTEGRANTES DEL GRUPO**

<b>-</b>	<b>Lopez francisco</b>	<b>107614</b>
<b>-</b>	<b>Corn Franco</b>	<b>109025</b>

## 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  $4 \text{ g/m}^3$  . 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.

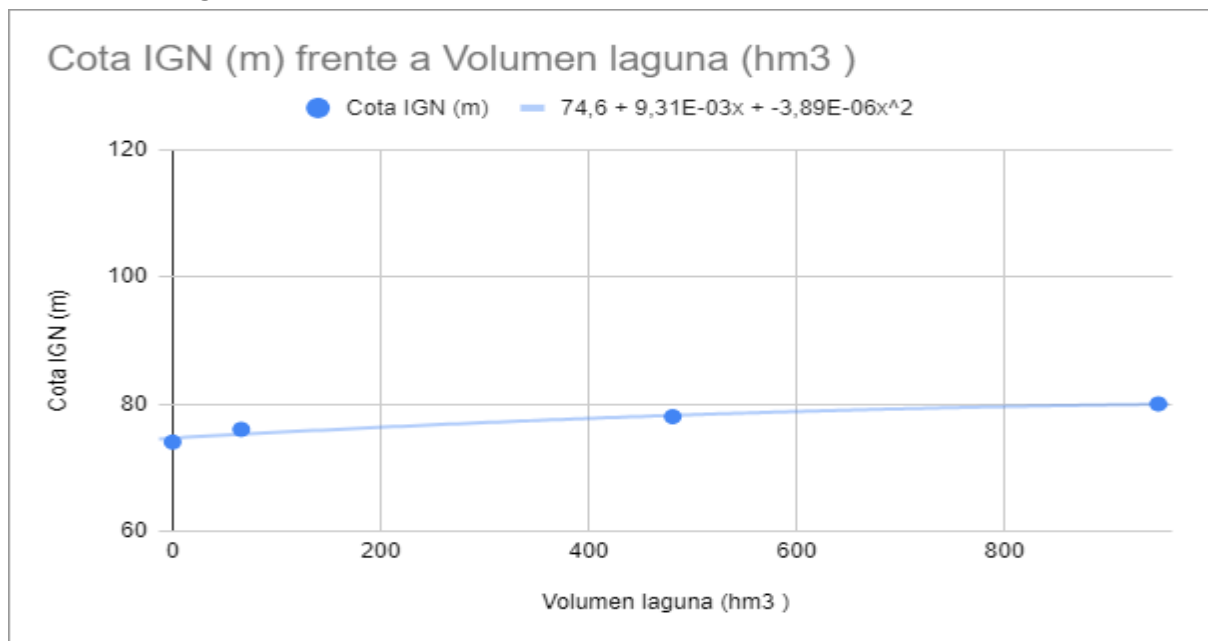
Importante: Toda la parte que tenga que ver con mostrar el código, la dejamos para el índice de “Anexo”

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

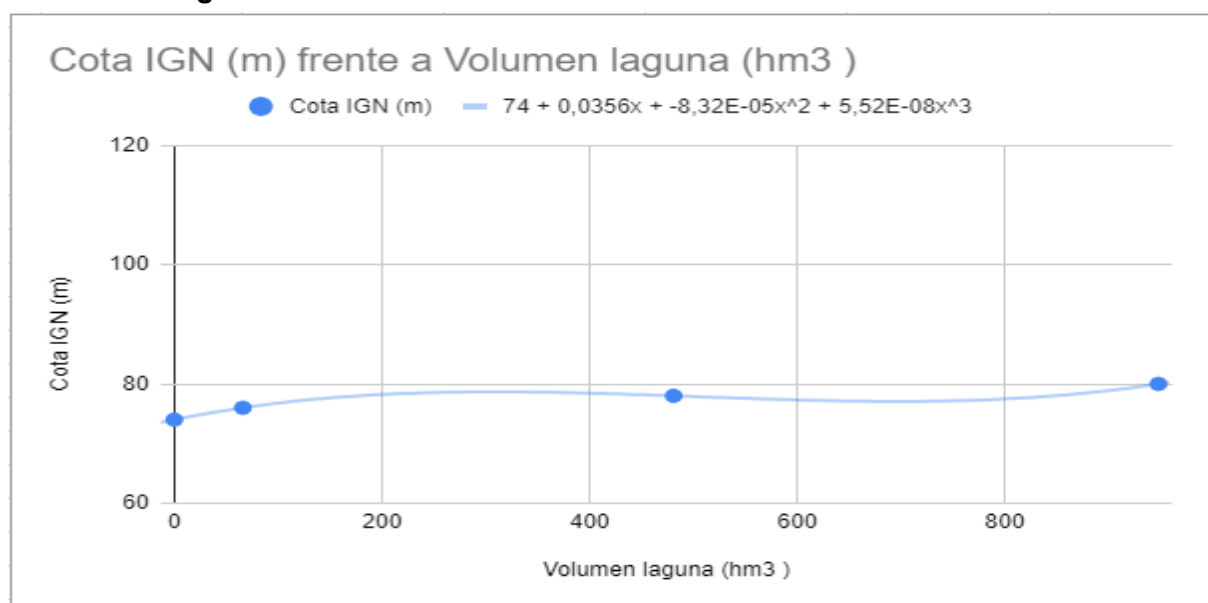
Para la búsqueda de los polinomios utilizamos el legajo(109025) y utilizando excel y su función “línea de tendencia” hallaremos los polinomios pedidos .

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  $Cota0 = 77m$  (utilizando el padrón 109025) utilizando la ecuación del polinomio llegamos a que el volumen inicial resulta:  $V_i = 110,9 \text{ h m}^3$ .

Para implementar nuestra solución numérica tenemos que discretizar nuestras ecuaciones, buscamos 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:

$$\text{OD: } V \frac{dOD}{dt} = Qe \cdot ODe - Qs \cdot OD + Ka \cdot V (OD_s - OD) - kbd0 \left( \frac{OD^2 DBO}{OD^2 + K0^2} \right)$$

$$\text{DBO: } V \frac{dDBO}{dt} = Qe \cdot DBOe - Qs \cdot DBO + 0 - kbd0 \left( \frac{OD^2 DBO}{OD^2 + K0^2} \right) \quad (G = 0)$$

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)

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

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

## (DBO)

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

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

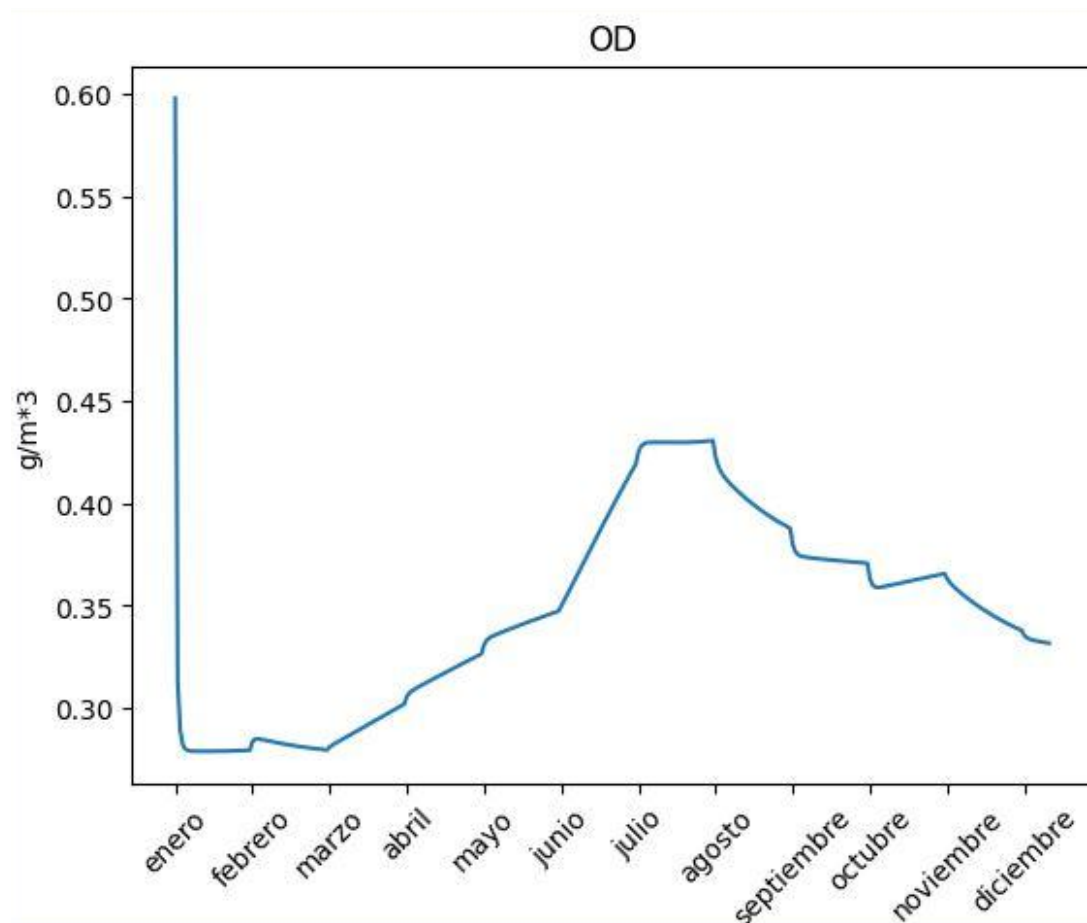
Con estos dos métodos vamos a aproximar como varía el DBO y el OD en la laguna. Recordamos que el volumen cambia día a día, mientras que los caudales son mensuales, es importante aclarar que el volumen se encuentra  $h m^3$  y nuestros caudales en  $m^3/s$  entonces vamos a pasar nuestro volumen a  $m^3$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

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

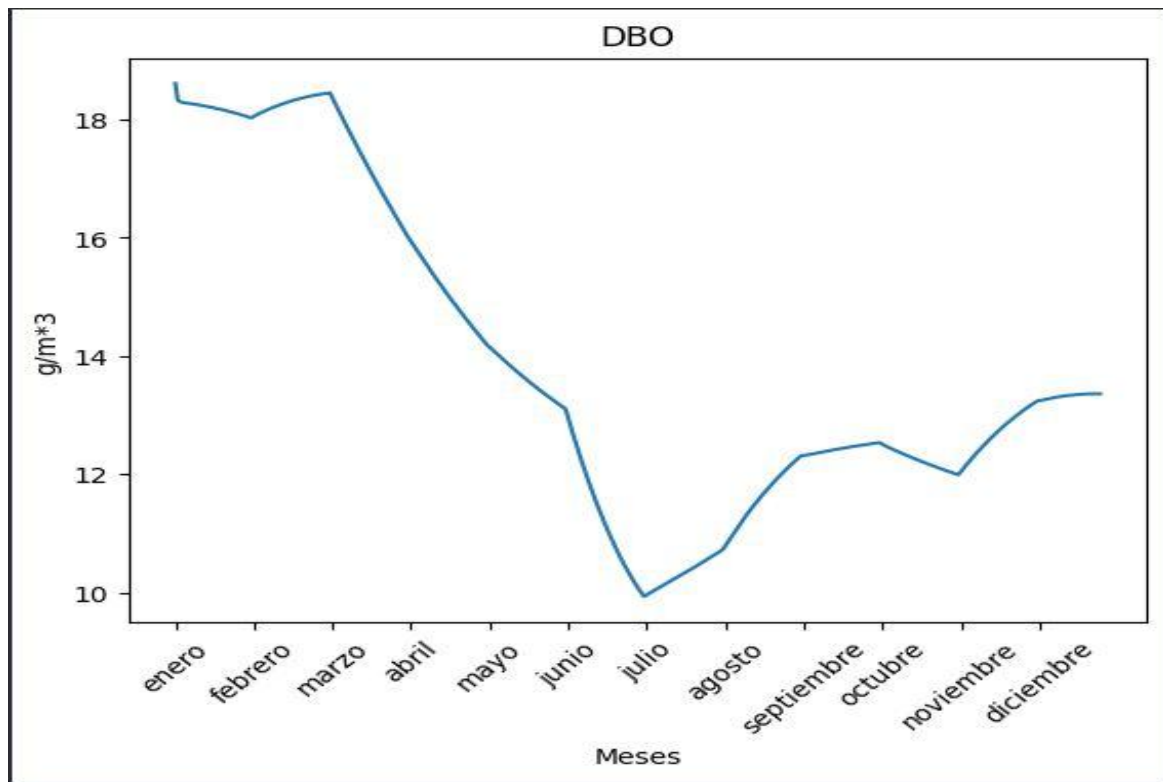
$$Q = m^3/s \rightarrow Q = (m^3/dia) 86400$$

Una vez pasados V y Q podemos iterar con nuestro código realizado en python (utilizamos el método euler explícito) y obtenemos los siguientes gráficos:

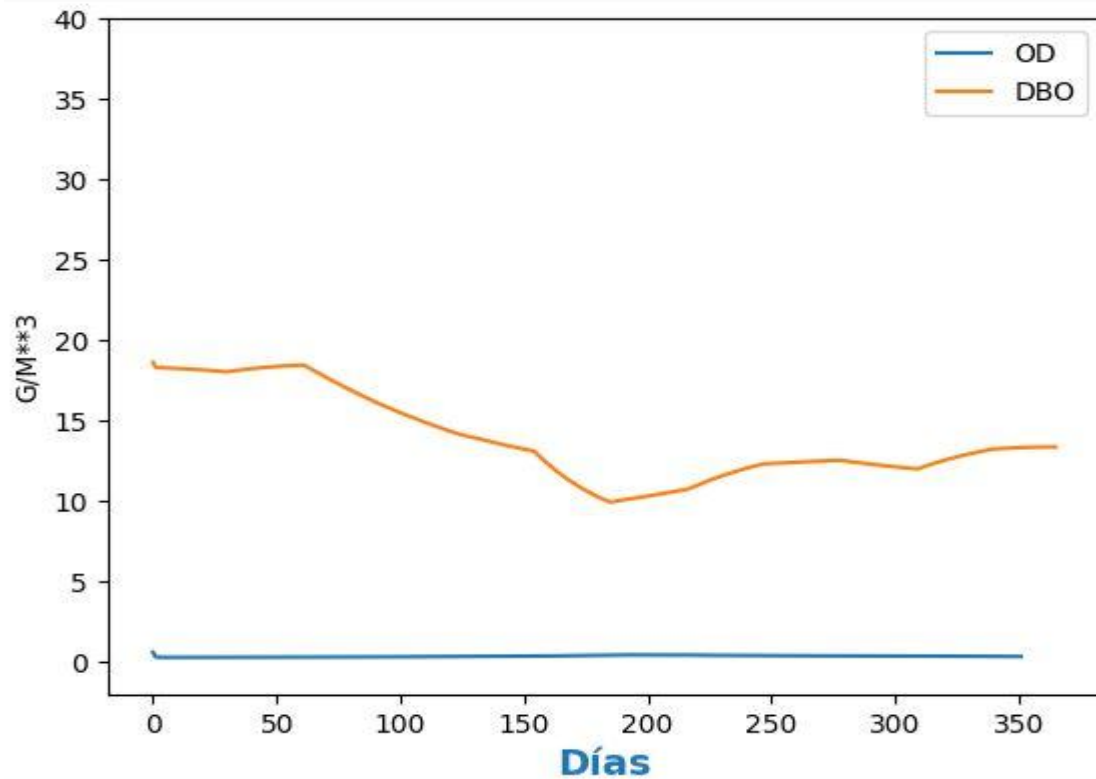
Empezamos graficando OD y DBO por separado







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

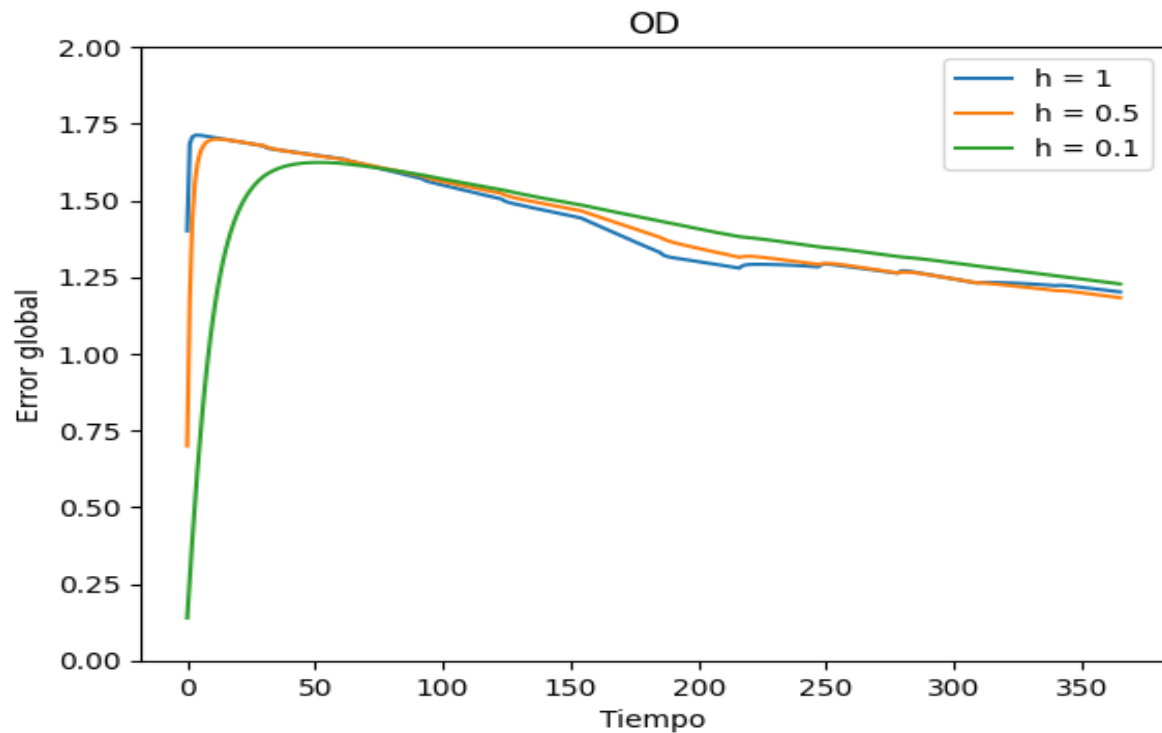
necesitamos analizar todos los factores además del análisis de DBO y OD vemos que la cantidad de OD en el lago es muchísimo más baja que la de DBO, esto podría indicar que el lago está contaminado.

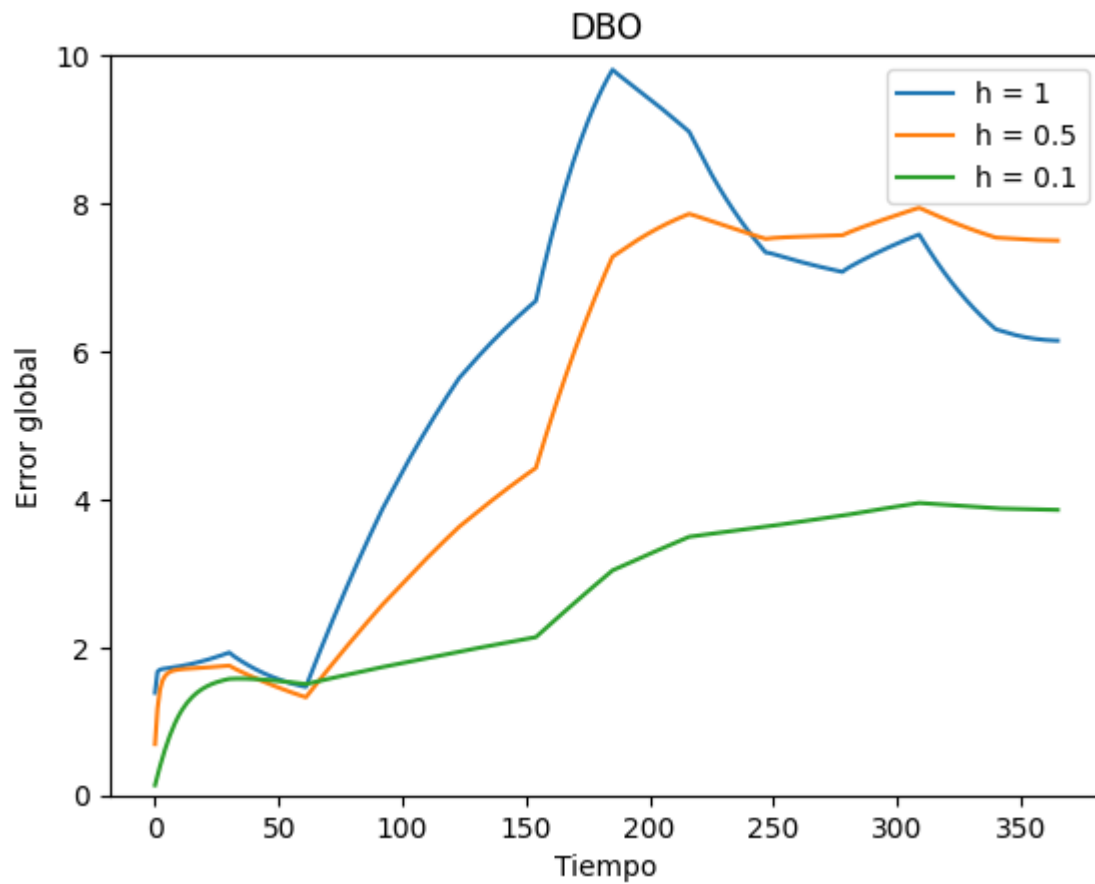
## Error de truncamiento con paso de discretización temporal diario

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.

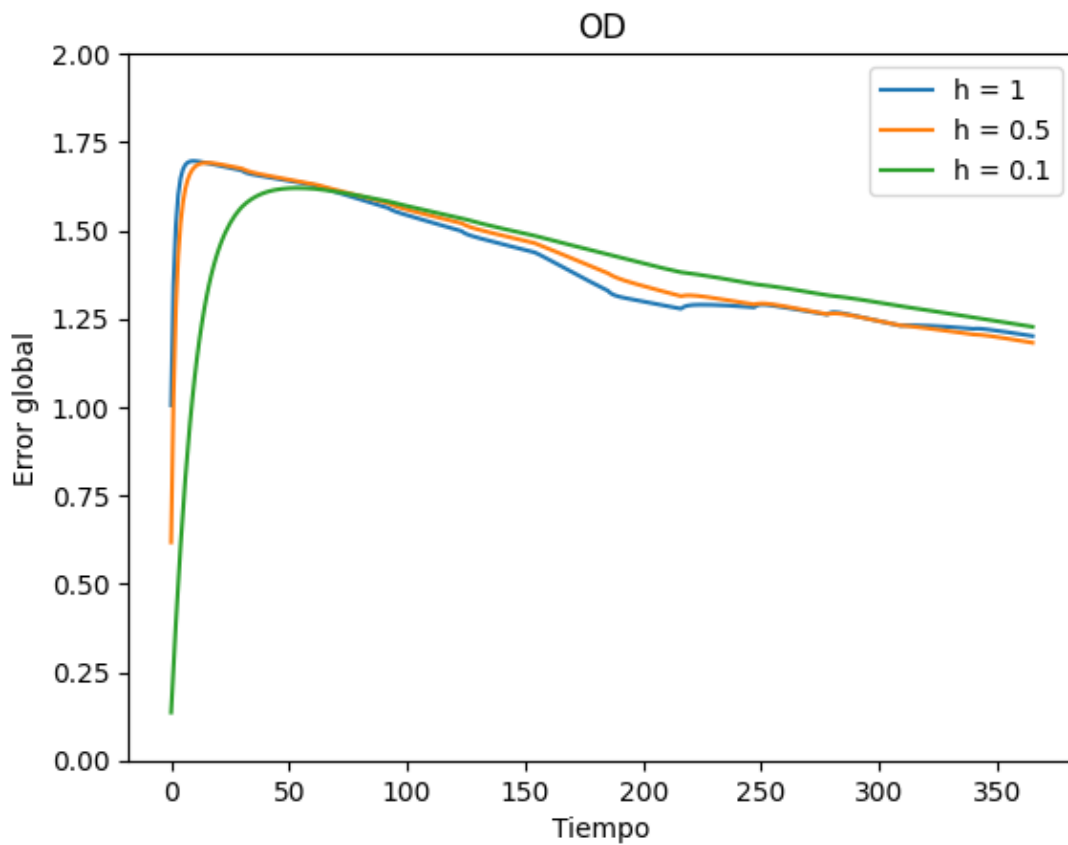
Empezamos con el método de euler explícito:

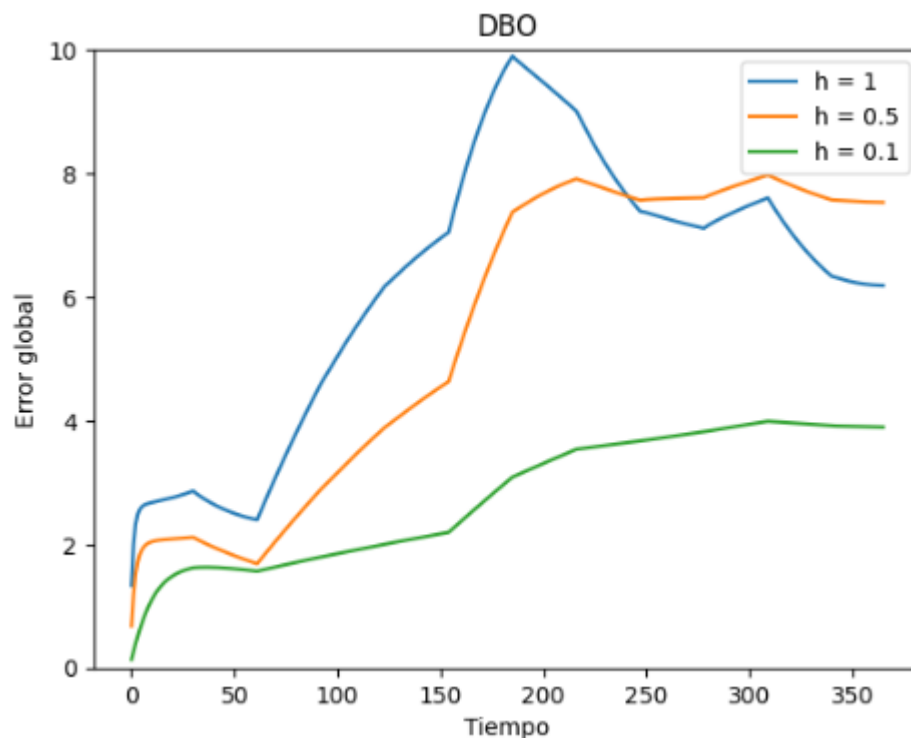
Método Euler Explícito:





Método Punto Medio:

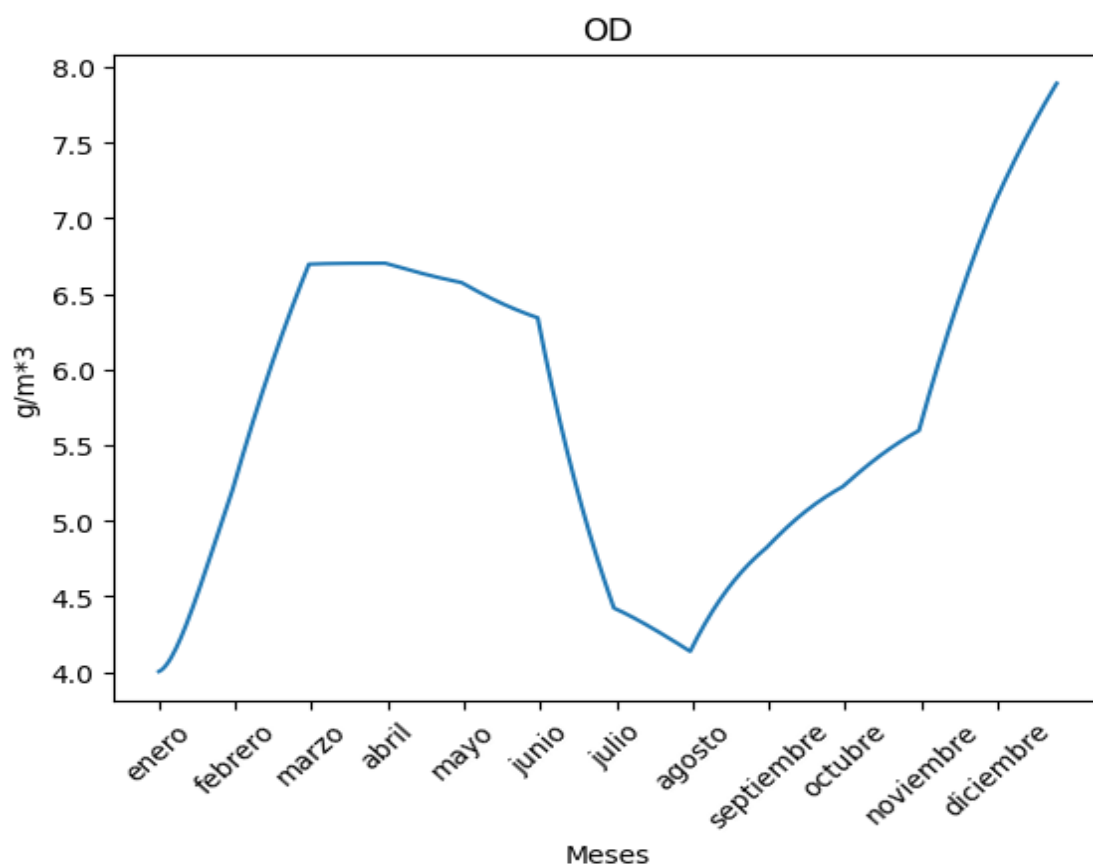
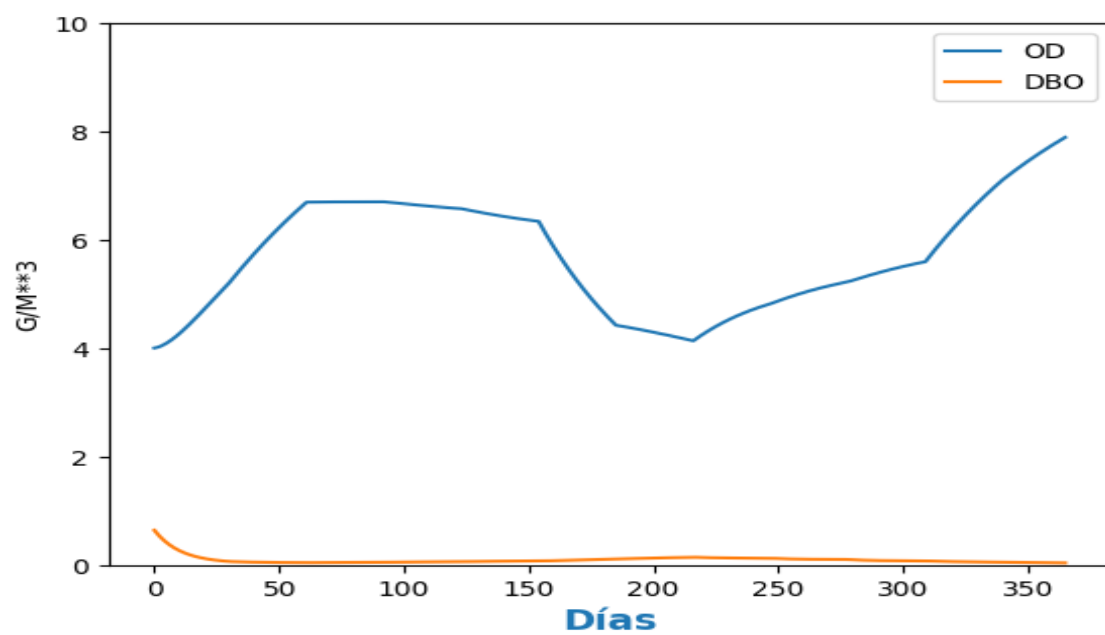


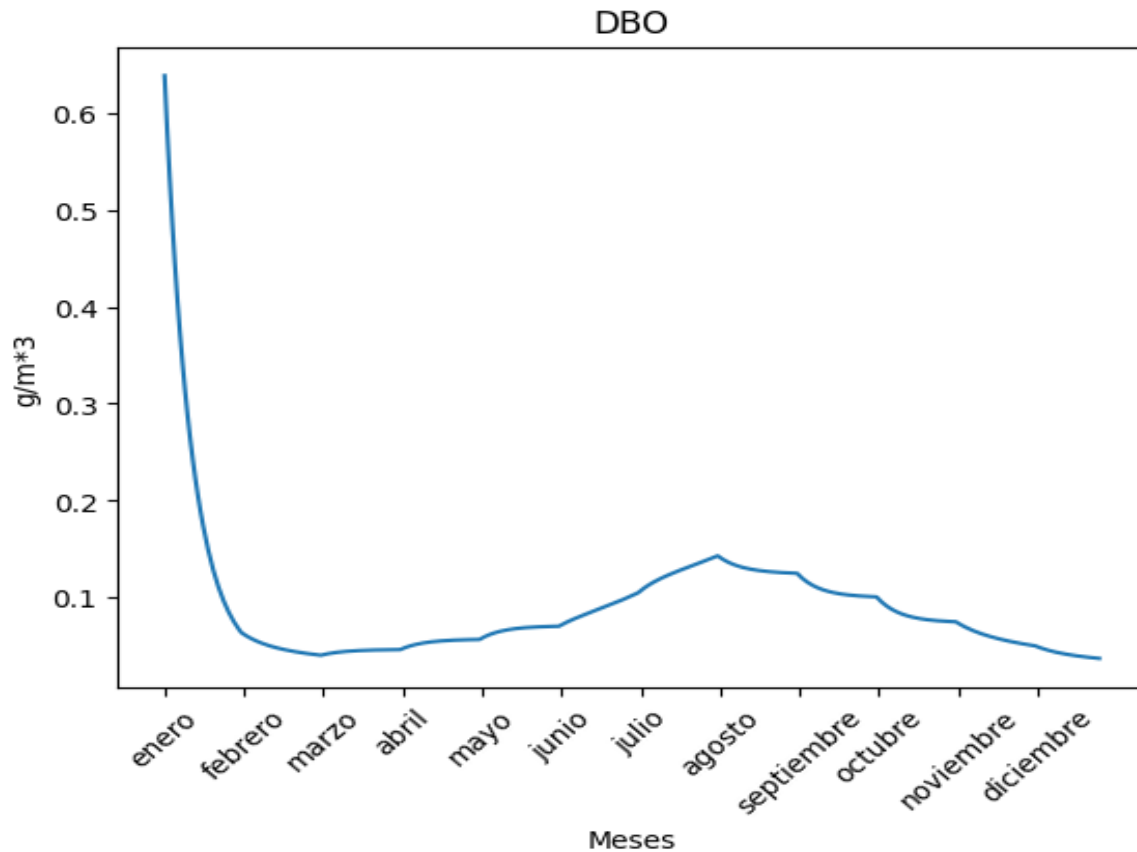


Como se puede observar ambos métodos dieron resultados de errores muy similares. Para OD se obtuvo un error bastante estable que desciende a medida que aumenta el tiempo y que es muy similar entre distintos valores de  $h$ . En cambio, para DBO se obtuvieron valores inestables, con muchos picos y que a medida que aumenta el valor de  $h$  aumenta el error.

## Reducción del valor de DBO para aumentar el de OD

Para garantizar que los valores de OD no estén por debajo de  $4 \text{ g/m}^3$  redujimos el DBO inicial de la laguna a  $0,7 \text{ g/m}^3$  es decir, en un 96,5%. Utilizando ese valor obtuvimos mediante la función `min()` de python el valor mínimo de OD encontrado es de  $4.004243113521108 \text{ g/m}^3$ . 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:

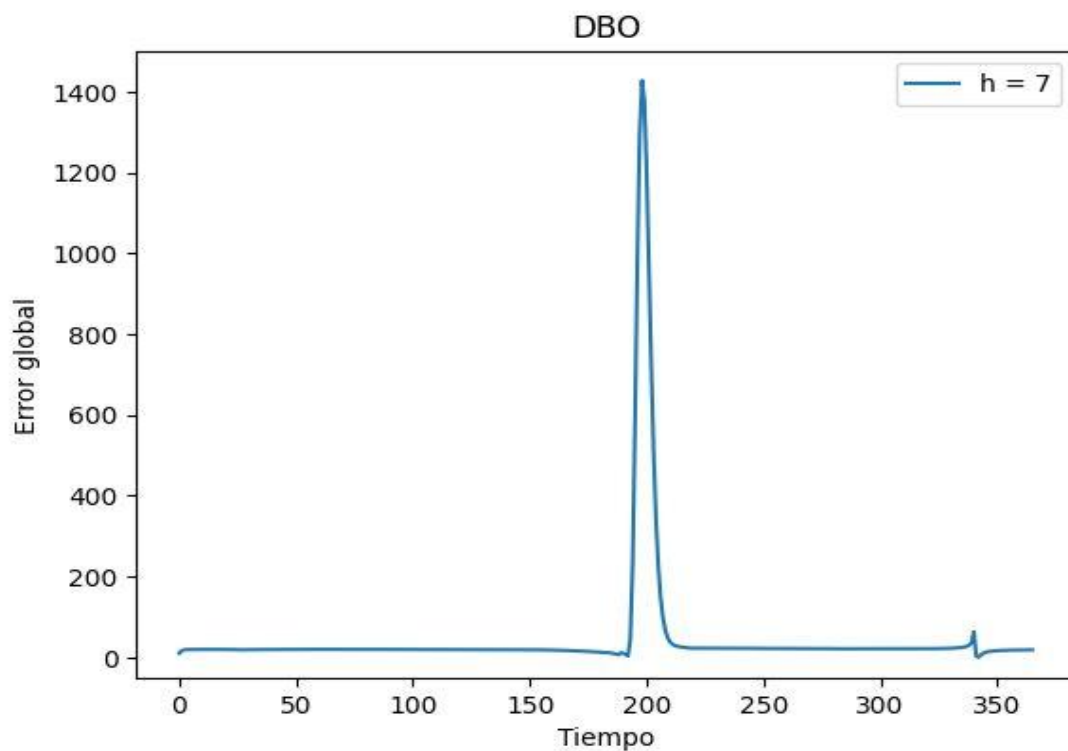
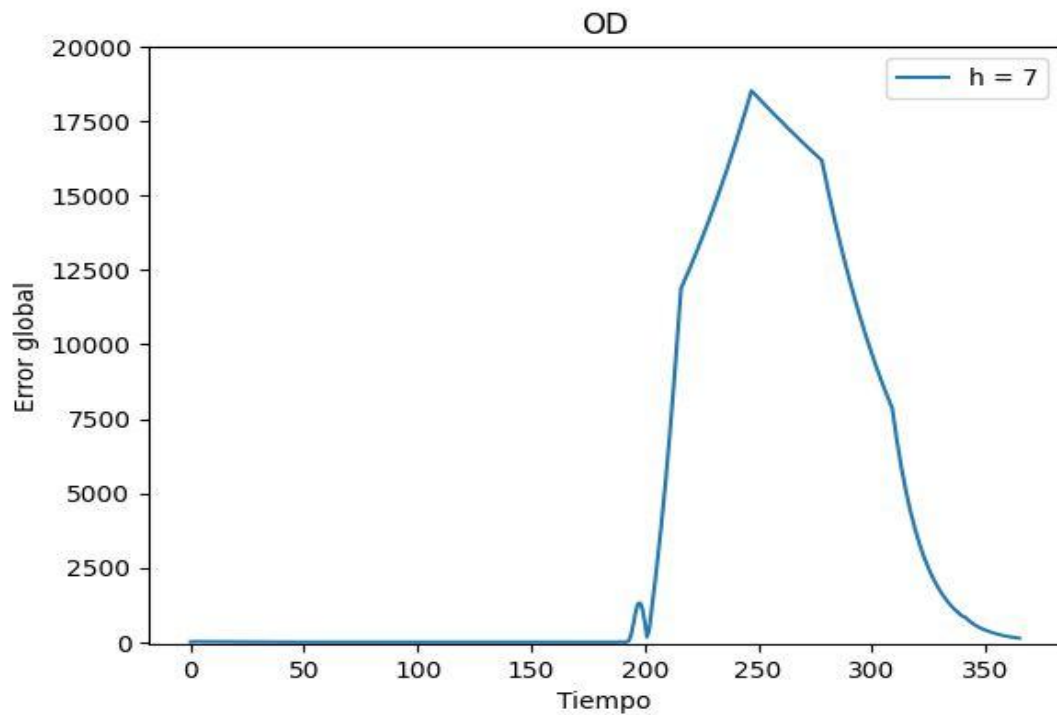




Se puede ver que, al igual que previo al cambio de DBO, a medida que aumenta este mismo, el OD disminuye y viceversa.

### **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.

## Conclusiones

¿Por que cuando graficamos los errores de dbo 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 qué en el punto F los gráficos tienen tantos picos?

Estamos trabajando con métodos explícitos tanto de orden 1 como orden 2, al tomar un paso de discretización muy grande vemos que nuestro error se comporta de manera inestable, por lo que pudimos observar lo que está sucediendo es que  $h = 7$  no esta en nuestro "rango de estabilidad" por lo que el método se comporta de cualquier manera y es inestable para calcular. Concluimos entonces que al tomar un  $h$  muy grande, nuestros métodos tienden a ser cada vez menos estables en el cálculo y aumentan su error.

¿La laguna está contaminada?

Si bien en el enunciado está aclarado, podemos ver que realmente el OD y el DBO tiene una diferencia muy grande en cuanto a cantidades entre sí, por lo que al ser el OD bajísimo en comparación al DBO, podemos decir que la laguna muy probablemente esté contaminada (no es lo único a analizar el DBO y OD para saber esto, pero como es lo que nos daban en el tp sacamos esta conclusión ).

¿Los métodos funcionaron de manera correcta?

Si bien los métodos en algunos casos fueron útiles para el análisis como por ejemplo para medio OD y DBO o para medir cuánto hay que reducir el DBO para que el od se mantenga en  $4 \text{ g/m}^3$ . En otros casos fallaron, por lo tanto la elección de ambos métodos no fue la más adecuada. Sin embargo pudimos resolver los ítems y llegar a las conclusiones necesarias pedidas en el problema.




# Anexo 1

## Código fuente

### 1.1 Estructura de carpetas

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



```
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 = 110.9*(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

    Vimas_1 = Vi + h*(lista_Qe[j]-lista_Qs[j])
    lista.append(Vimas_1)

    for i in range(cant_de_iter):
        Vi = Vimas_1
        Vimas_1 = Vi + h*(lista_Qe[j]-lista_Qs[j])

        lista.append(Vimas_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 = DBOe
    OD_i = ODe
    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 = 0.7 #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 = 110.9*(10**6) #Pasamos el volumen iniciar a m**3
```

### 1.2.3 Punto\_medio.py

```

kbd0 = 0.1
ka = 0.01
ko2 = 1.4
DB0e = 20
ODe = 2
ODs = 9
Vi = 110.9*(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 = DBOe
    OD_i = ODe
    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 = [0.5978239833250498, 0.3126126025761546, 0.2891309759861442, 0.28231085878133544, 0.28012599273364674, 0.2793974261325166, 0.2791445053417159, 0.
0.36042176960791206, 0.3593171728099308, 0.3590090640347364, 0.3590429284376806, 0.35922348242647834, 0.3594662890334357, 0.35973471776654, 0.36001284797895
0.3315972717288502, 0.3314242655358883, 0.3312585702352353, 0.33110005415133853, 0.3309486074725213, 0.33080413144252363, 0.33066653356198117, 0.33053572543
```

```
lista_DBO = [18.611573166583828, 18.32570135098747, 18.298322893725796, 18.286990374263173, 18.27984762783491, 18.273765486163608, 18.2677812429402, 18.2615
17.935035006275434, 17.850965665923827, 17.767482672312756, 17.684582332943123, 17.602260760434984, 17.520514022867612, 17.439338191996615, 17.3587293587202
12.4047906601327, 12.412636698872744, 12.420357839075375, 12.427955996986102, 12.43543310077724, 12.442791065722316, 12.45003178407005, 12.457157121080229,
```

Python

```
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/m*3')
plt.title('OD')
plt.xticks(range(1, len(lista_OD) + 1, 31), meses, rotation=45)
plt.show()
```

Python

```
#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/m*3')
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/M**3")
ax.set_ylim([-2,40])
plt.show()
```

## Punto E

```
✓ lista_OD_e = [4.004243113521108, 4.013966984544379, 4.028593209023968, 4.047601489803472, 4.07052320969111, 4.096935799987886, 4.126457804229069, 4.15874454
4.867360928585621, 4.916305868601843, 4.965266894657699, 5.0141926726382415, 5.063037048186313, 5.111758580016629, 5.160320114232809, 5.2086883960114205, 5.
6.700957835684577, 6.701246189592653, 6.701504479126589, 6.701735828576908, 6.70194303775125, 6.702128615667149, 6.702294810747248, 6.7024436378798535, 6.70
6.703497465586888, 6.6988973093554876, 6.694272189985952, 6.6896342206559485, 6.684994058794938, 6.680361063173968, 6.675743434341331, 6.671148340162377, 6.
5.800587837521877, 5.72928457617973, 5.659421040583897, 5.590965115457665, 5.523884696469572, 5.458147750080761, 5.393722365193685, 5.33057679756935, 5.2686
✓ lista_DBO_e = [0.6388890965903318, 0.5833724891069452, 0.5329259254931719, 0.48707748556615604, 0.445401734996033, 0.4075146104725963, 0.3730689437742132, 0.
0.04175442387401579, 0.04139839065445271, 0.04105586132687098, 0.04072580643627344, 0.04040728844616952, 0.04009945329070541, 0.040724148385285056, 0.041283
0.045864299283607884, 0.04589027576591932, 0.04591355429995045, 0.047052788910161306, 0.04807199203031406, 0.04898383995824761, 0.04979966956118544, 0.05052
```

Python

```
#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/M**3")
ax.set_ylim([0,10])
plt.show()
```

Python

### 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/m*3')
plt.title('OD')
plt.xticks(range(1, len(lista_OD_e) + 1, 31), meses, rotation=45)
plt.show()
```

```
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/m*3')
plt.title('DBO')
plt.xticks(range(1, len(lista_DBO_e) + 1, 31), meses, rotation=45)
plt.show()
```

## Punto D

Graficamos los errores para OD y DBO con euler explícito

+ Code

+ Markdown

```
lista_error_h_1 = [1.400773884112797, 1.684583787966757, 1.706664593689889, 1.7120845464165875, 1.7128699047355147, 1.7121996207176817, 1.711054348359633, 1.7
1.66306660256321, 1.661949365410256, 1.660827412115549, 1.659700035777689, 1.6585670792565692, 1.6574285623235863, 1.656284560362815, 1.6551351649752404, 1.
1.6457569306838775, 1.6445630709144785, 1.6433646851083894, 1.6421618484820075, 1.6409546343342245, 1.6397431141031085, 1.638527357420522, 1.637307432164760
1.5278141568532908, 1.5258452020406514, 1.5238786972086482, 1.5219146562302357, 1.5199530928342475, 1.5179940206010543, 1.5160374529600822, 1.51408340318806
1.2704982355319374, 1.269599152206872, 1.2683588577830323, 1.2669726360770301, 1.2655249242801292, 1.2640523531071526, 1.262570843426984, 1.261087321006403,

lista_error_h_0_5 = [0.6996750265767047, 1.182663224882619, 1.4383305982276389, 1.5587398335480813, 1.6200076288114666, 1.6539426915779325, 1.67370739662853
1.688856622271183, 1.6874888514224358, 1.6861151832107721, 1.6847375353350793, 1.6833571856965854, 1.681974986392635, 1.6805915060774614, 1.6792071246923586

lista_error_h_0_1 = [0.13881155609438123, 0.27072041390644963, 0.39535314062638593, 0.5123795686280026, 0.6215390088796438, 0.7226659483685536, 0.8157112164
1.3519874813280828, 1.3507713733547435, 1.3495511555124275, 1.3483271613311365, 1.3476379796499347, 1.3469090969671857, 1.3461432844260062, 1.34534311662446
1.2383584845321596, 1.237298477472089, 1.236234310939208, 1.235166285801851, 1.23400946842583866, 1.2330197710093924, 1.2319417943547313, 1.2308609872205212,
```

✓

0.2s

Python

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()
```

DBO

```
lista_DBO_h_1 = [1.3870388411279748, 1.6715233056887193, 1.6975150535347616, 1.707461505569011, 1.7132188269233062, 1.7179161862420713, 1.7225162902012592, 1.725629624559697, 8.219101315590898, 8.16369620116475, 8.109391905380798, 8.05616634915587, 8.00399782462801, 7.952865023515789, 7.902747045276696, 7.853627.142017535839566, 7.160932119668937, 7.179284244860067, 7.197268270935455, 7.2149696740538385, 7.232426956548258, 7.249658256099913, 7.26667284948995, 7.286.44977897519637, 6.417926196728866, 6.386832557624114, 6.3564883322418275, 6.326883975344446, 6.29801011992811, 6.287965075032233, 6.277296718739116, 6.26

lista_DBO_h_2 = [0.692716932433715, 1.1723500507652567, 1.4271814676400538, 1.5481368124933468, 1.6106560423440008, 1.646251710382316, 1.6679490836178559, 1.8689644641455914, 1.9097333395042462, 1.9503337693541312, 1.990766245846153, 2.031031320858986, 2.071129584518804, 2.111061651142471, 2.1508281500335897, 7.5430087162421735, 7.544487156343452, 7.545905142601917, 7.5472751818751025, 7.548606289643763, 7.5499049752491185, 7.551175948066476, 7.552422624112507, 7.8217268918790115, 7.8075312162867565, 7.793498891238581, 7.7796320560087135, 7.76593201804555, 7.752399472456169, 7.739034664224153, 7.72583750809928, 7.77.5369334478051915, 7.534836267988691, 7.532598368409998, 7.530286471466114, 7.527949284582498, 7.5256223907734725, 7.523331782435065, 7.521096427901222, 7.

lista_DBO_h_3 = [0.1374155609438148, 0.26807255050966816, 0.3915906288735691, 0.507632347967121, 0.6159294081535975, 0.7163084251309009, 0.8087121914242736, 1.4897064767431196, 1.5042541053581502, 1.517343666511909, 1.529135611538237, 1.5397703924250656, 1.5493711096597202, 1.5580458315707553, 1.56588960964572273.479540478566072, 3.49373960362184, 3.498464111480871, 3.503145799285999, 3.5077871609636446, 3.512390511429995, 3.516958000159832, 3.521491623644522, 3.523.765295980462575, 3.7699428127954917, 3.7745779811296103, 3.779201871446057, 3.7851550602267388, 3.7910638607080216, 3.796930921477891, 3.802758707944534, 7.
```

✓ 0.2s

Python

```
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()
```

✓ 0.6s

Python

```

Metodo Punto Medio

lista_h_1 = [1.0053029614629898, 1.3460607732702892, 1.5122540480657818, 1.600465059534083, 1.647481073486647, 1.6724282299453261, 1.6854862118140845, 1.69
1.3567977937673692, 1.353320867487481, 1.3498591048440547, 1.3464130536570904, 1.342983215713236, 1.339570041875781, 1.3361739269139745, 1.3327952040558921

lista_h_2 = [0.6174356627248903, 1.030191834431895, 1.2788268048586944, 1.4279310362966642, 1.5201501661824042, 1.578983601045568, 1.617459436831262, 1.643
1.6601008615435808, 1.6587073558077476, 1.6573694517514554, 1.6560711822420733, 1.654801049347979, 1.6535507786510353, 1.6523144190916192, 1.65108769326029
1.2393213060347121, 1.2379336010049402, 1.2365464748999508, 1.2351601124701528, 1.2337746551614974, 1.2323902112570468, 1.2310068636386244, 1.2305739946252
1.2189774032902, 1.218096349490408, 1.2172108946233293, 1.2163214115760252, 1.215428196574453, 1.21453148652484, 1.213631724088235, 1.2127283096296817, 1.

lista_h_3 = [0.135953968157974, 0.2649142616009643, 0.3865618750181068, 0.5006350219505884, 0.6069516174585303, 0.7054285913452347, 0.7960947743362565, 0.8
1.2923402760706417, 1.2912060320405723, 1.2900650520831591, 1.2889178015193221, 1.2877647165533936, 1.2866062061288144, 1.285442653662002, 1.28450741272032
1.2305216582134852, 1.2294400639950926, 1.2283560703400374, 1.227269863088851]

✓ 0.1s Python

(variable) medidas: dict[str, list[float]]

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()

✓ 0.4s Python

```

```
DBO

lista_1 = [1.335615676482643, 1.9991002039413885, 2.3200870656493535, 2.478729789404589, 2.5603522954807616, 2.6048206652640467, 2.6307489098451278, 2.6471
7.503465495431623, 7.463084388584889, 7.423481844597669, 7.384642508936302, 7.380926569367947, 7.374097935860954, 7.365536345336537, 7.356032562559776, 7.3
7.45480483963056, 7.469065828436301, 7.483151957869531, 7.497065248913941, 7.5108076860645046, 7.524381223276437, 7.5377877875908545, 7.551029281383466, 7.
7.56687568285387, 7.535790437565353, 7.534793769900277, 7.533897544742981, 7.533101636883231, 7.532405841163554, 7.5318098915272635, 7.531313475697518]

lista_2 = [0.6798631889385796, 1.181675724736266, 1.5073005689270111, 1.70651534849274, 1.828847987365851, 1.9061630739071163, 1.9566897857518804, 1.990743
7.62505216887917, 7.613869723001722, 7.602843902585647, 7.591973819737822, 7.581258588006829, 7.5706973237577415, 7.568634544291237, 7.5664588147458485, 7.
7.536887568285387, 7.535790437565353, 7.534793769900277, 7.533897544742981, 7.533101636883231, 7.532405841163554, 7.5318098915272635, 7.531313475697518]

lista_3 = [0.13690152511674114, 0.26721639096216165, 0.3906012559635812, 0.5067567899825498, 0.6154502003852365, 0.7165363634755302, 0.8099743354246804, 0.
1.5256756710013448, 1.5419258399152227, 1.5566082330522804, 1.5698886279377824, 1.5819132609725344, 1.59281120015633, 1.6026964700749353, 1.611669934473667
0.1s Python
```

## 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 explícito de OD y DBO respectivamente, con un paso de discretización diario.

DBO:

```
[18.611573166583828, 18.32570135098747, 18.298322893725796, 18.286990374263173, 18.27984762783491, 18.273765486163608, 18.2677812429402, 18.261586080750934, 18.255080866924313, 18.248235058097457, 18.241040687588164, 18.233497222398093, 18.22506056241837, 18.217371386581977, 18.208794605650528, 18.199879181305732, 18.190628066590254, 18.1810441865752, 18.171130432366883, 18.16088965953452, 18.150324687998413, 18.139438302395405, 18.128233252594757, 18.116712254256, 18.104877989392786, 18.092733106930947, 18.08028022325705, 18.067521922756367, 18.054460758340092, 18.041099251961906, 18.027439895124143, 18.054147544815535, 18.077291964502702, 18.098802366950775, 18.11934348251932, 18.13914080549524, 18.15827332878018, 18.176771807192242, 18.194651136659964, 18.211920954882952, 18.22858910397636, 18.24462761245262, 18.26014880932894, 18.27505395844337, 18.289384787856964, 18.30314776103843, 18.31634923219728, 18.328995449982568, 18.341092560215756, 18.352646608268472, 18.3636635412849, 18.374149210314936, 18.384109372380756, 18.393549692485376, 18.402475745567106, 18.41089301840228, 18.418806911458127, 18.426222740697348, 18.43314573933598, 18.43958105955596, 18.445533774173786, 18.45100887826663, 18.46015768462238, 18.4716591586091, 18.480775969863633, 18.48942689874846, 18.49693720435377, 18.50305006275434, 18.50965665923827, 18.51682672312756, 18.52482332943123, 18.5326260760434984, 18.540514022867612, 18.548338191996615, 18.5562935872027, 18.56436836798497, 18.572669717120266121938815, 18.5814886685270942, 18.590405078313937, 18.599467544742053, 18.608620353684646, 18.61793809799554393, 18.6273322018752, 18.636293905110822, 18.6453612784945, 18.654260715859732, 18.663067863547219, 18.671611479862818, 18.68015505715662075, 18.6887833435348, 18.6971464347519495, 18.705421795860547, 18.71396715321426204, 18.72239986238716, 18.730669872631593, 18.73874191592212128, 18.7467722173469984, 18.7546001409173573, 18.76224670115112, 18.769819450225418, 18.777426108829161658,
```

OD:

```
[0.5978239833250498, 0.3126126025761546, 0.2891309759861442, 0.28231085878133544, 0.28012599273364674, 0.2793974261325166, 0.2791445053417159, 0.2790503179058597, 0.27901085016466803, 0.27899176202882, 0.27898167683159786, 0.2789768079304983, 0.27897588126216455, 0.27897845563499946, 0.2789843664692879, 0.2789935414202056, 0.27900959302092103, 0.2790215293085473, 0.2790402851826349, 0.27906218203200517, 0.27908719603403687, 0.27911530408185065, 0.2791464836893037, 0.2791807129515678, 0.27921797052447567, 0.27925823561035323, 0.27930148794623444, 0.2793477077930798, 0.2793968759255334, 0.2794489736220539, 0.27950398265536136, 0.283628124827352, 0.28481337527687506, 0.285010314317177, 0.28488567819099875, 0.2846604404113281, 0.28440700074482905, 0.28414893101780536, 0.2838938394508568, 0.2836441279081878, 0.2834005036463016, 0.2831631241602135, 0.282931970033239, 0.28270696623581726, 0.28248802152127706, 0.28227504117381763, 0.2820679310932313, 0.28186659906625094, 0.2816709551261576, 0.28148091161841643, 0.2812963831728671, 0.2811772866474704, 0.2809435410646194, 0.28077506754674864, 0.2806117892533508, 0.28045363132001283, 0.28030052079960377, 0.280152386605914, 0.2800091594574206, 0.27987077182787445, 0.27973715789233483, 0.2796082534798663, 0.2809913219087438, 0.28189938541570964, 0.2826580719328352, 0.28336980170942605, 0.28406681721245597, 0.2847593072270008, 0.285450503422511, 0.286141431424117, 0.28683241345180766, 0.2875235461646458, 0.2882148534662695, 0.2889063356425369, 0.28959798525071057, 0.2902897922859596, 0.29098174587047488, 0.2916738348105374, 0.292366047777853, 0.29305837337379, 0.2937508001496, 0.29444331661312806, 0.2951359112329341, 0.29582857243925065, 0.2965212886250954, 0.29721404814710684, 0.2979068393263297, 0.2985996504489885, 0.29929246976726237, 0.29998528550006637, 0.300677888583384076, 0.3013708589233477, 0.3020635928924769, 0.3056594511286206, 0.3073891137413407, 0.30843740550850611, 0.3092382115826312, 0.30994851859471384, 0.3106250279419577, 0.31128825686857853, 0.31194564861602936, 0.312599907737161, 0.31325201481165893, 0.31390232078769725, 0.3145509457895455, 0.31519792503636634, 0.31584326244653926,
```

## 2.2 Euler\_explicito\_e.py

Este programa imprime el valor mínimo de OD para un DBO reducido en un 96,5%

**4.004243113521108**

## 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:

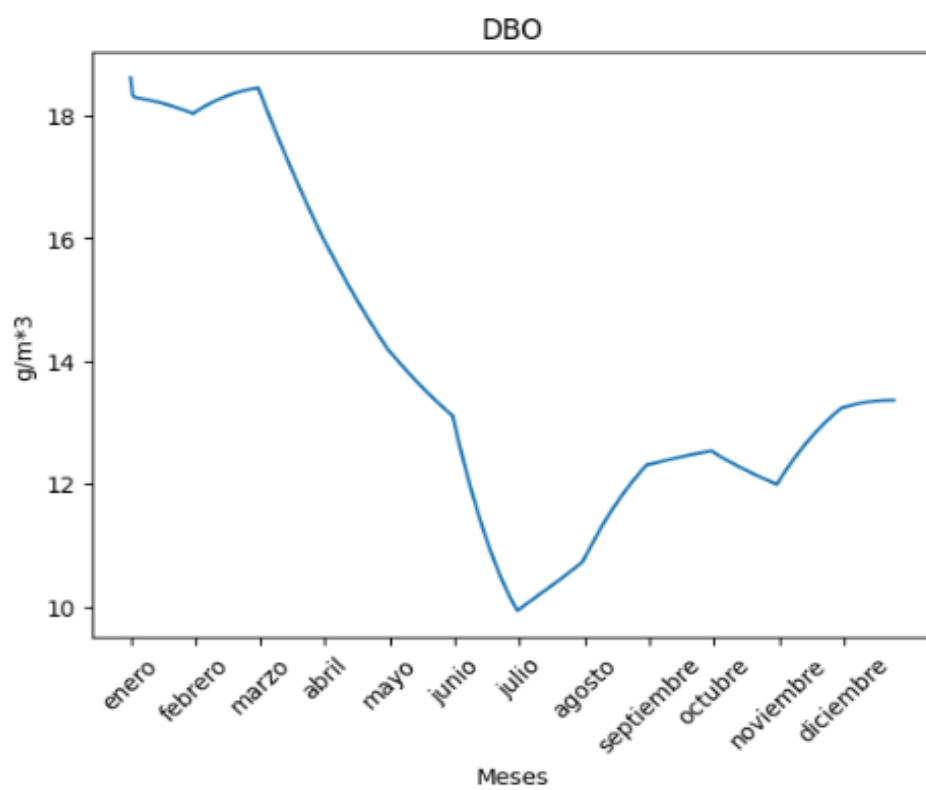
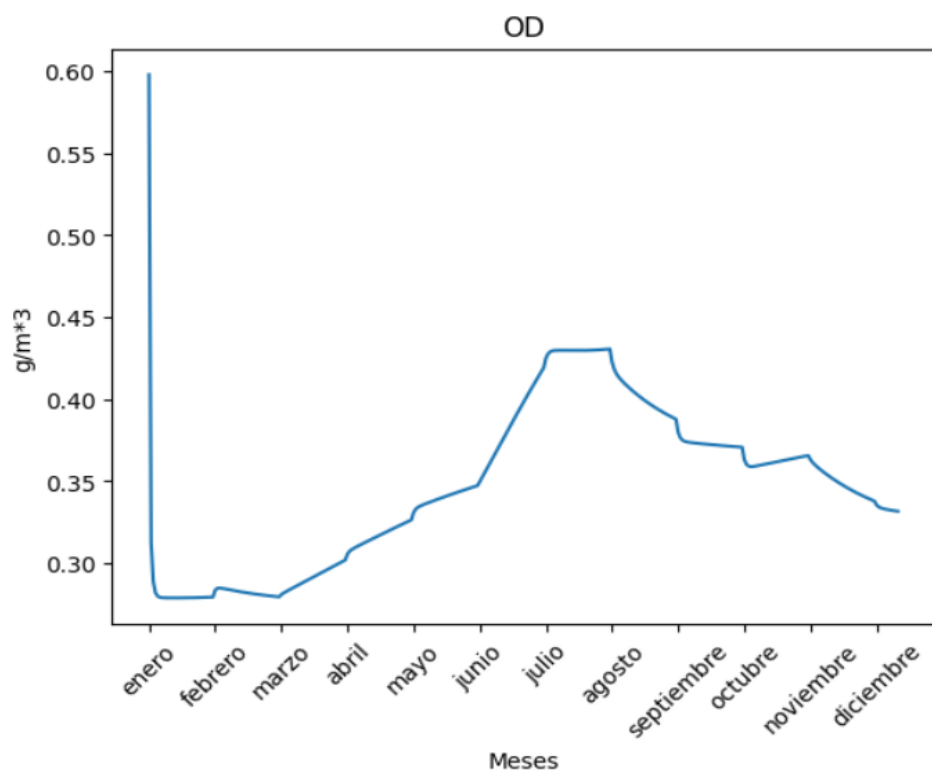
```
[18.66299638263628, 17.99812455539651, 17.675751035374656, 17.515722295139618, 17.43271441478456, 17.386861313290005, 17.359548979931837, 17.341752700753307, 17.328895202999373, 17.318570868142082, 17.309485334926347, 17.300932383001573, 17.292523888655165, 17.28404604038534, 17.27538128086355, 17.26646543547149, 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, 17.330719838293902, 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, 16.970135232975434, 16.893607103094663, 16.817608696686964, 16.742137188838477, 16.667189369235686, 16.592761875682804, 16.518851272679576, 16.445454094758205, 16.372566870377195, 16.30018613507724, 16.228308438700815, 16.156930349326817, 16.086048455385736, 16.01565936676622, 15.945759715362158, 15.876346155309323, 15.8074153630
```

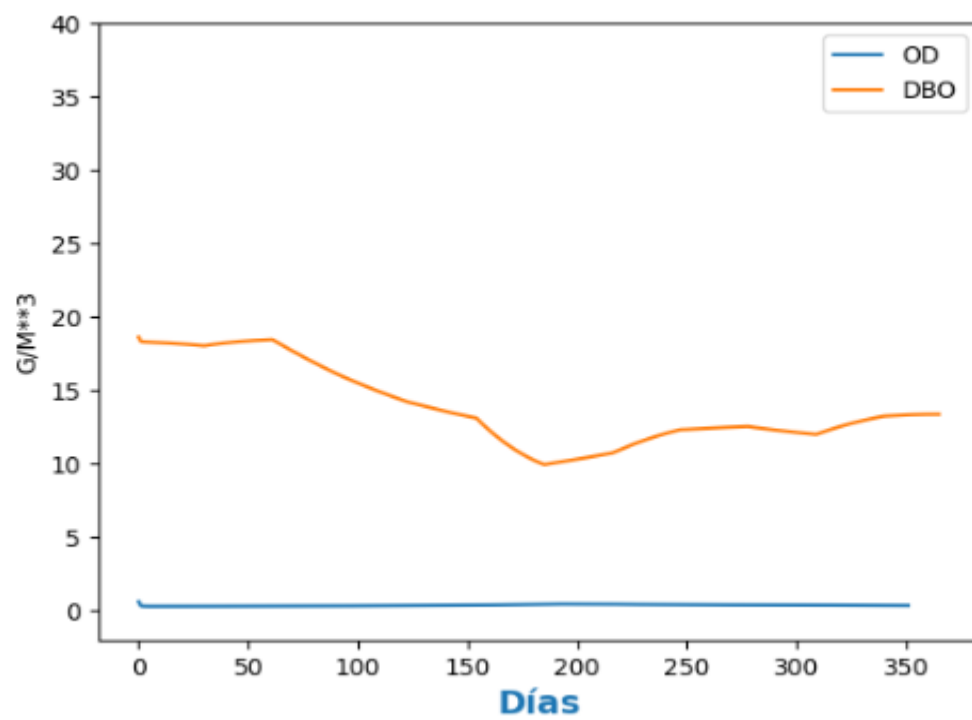
OD:

```
[0.9932951823491836, 0.651136170131185, 0.4835423510626018, 0.39393145181917016, 0.34551620694965635, 0.3191704767922949, 0.30471457880307484, 0.2967281322936338, 0.29229478424637406, 0.28982632153026094, 0.2884497809112597, 0.28768203542343523, 0.2872552127246122, 0.28702002258400117, 0.2868933062187824, 0.2868286722059883, 0.286802208784017, 0.2867934822235424, 0.28680036466703096, 0.2868163735863033, 0.2868388590394564, 0.28686649808088416, 0.2868984446831516, 0.28693423602504275, 0.2869736035878086, 0.2870163877194782, 0.28706248988587746, 0.2871184597705797, 0.287164411379612, 0.28722015262515793, 0.28727904271513666, 0.29011125619232264, 0.2915533869444122, 0.2922218223390179, 0.2924619965177503, 0.29246685701596464, 0.2923439590497947, 0.2921531694090528, 0.2919277650764497, 0.29168621181459997, 0.29143873065547937, 0.29119095560133795, 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, 0.2877925480418026, 0.2876429639859465, 0.2874985360201595, 0.28735919399182297, 0.2882009231967447, 0.288972525666739, 0.28970556061615876, 0.29041738582396325, 0.2911175514740605, 0.2918113202258831, 0.29250158858984787, 0.29318995177150786, 0.2938772867183457, 0.2945640737643843, 0.2952505748005608, 0.2959369272167698, 0.29662320595977004, 0.2973094455157578, 0.2979956604597036, 0.29868185384847046, 0.2993680223195196, 0.3000541589234197, 0.3007402546991908, 0.30142629955151956,
```

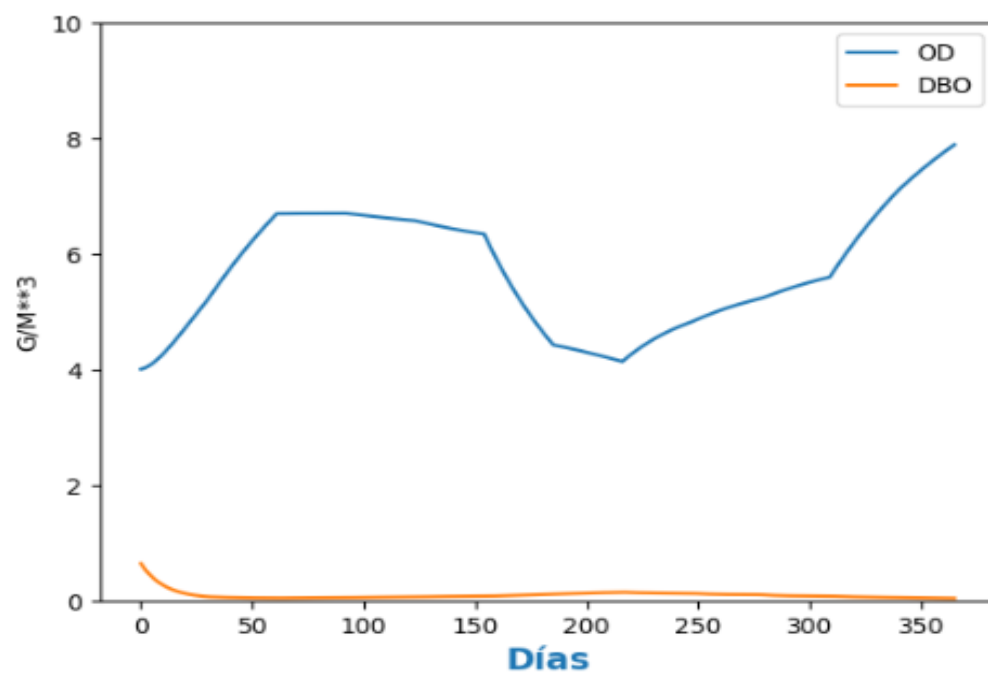
## 2.4 Graficos.ipynb

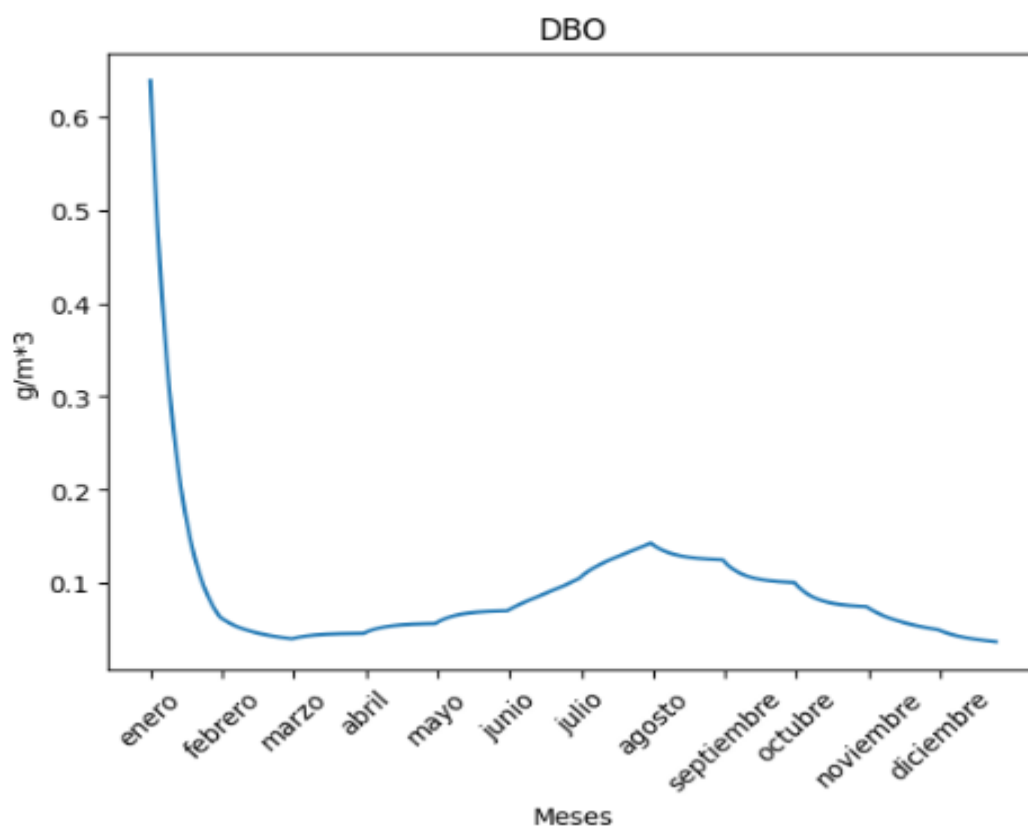
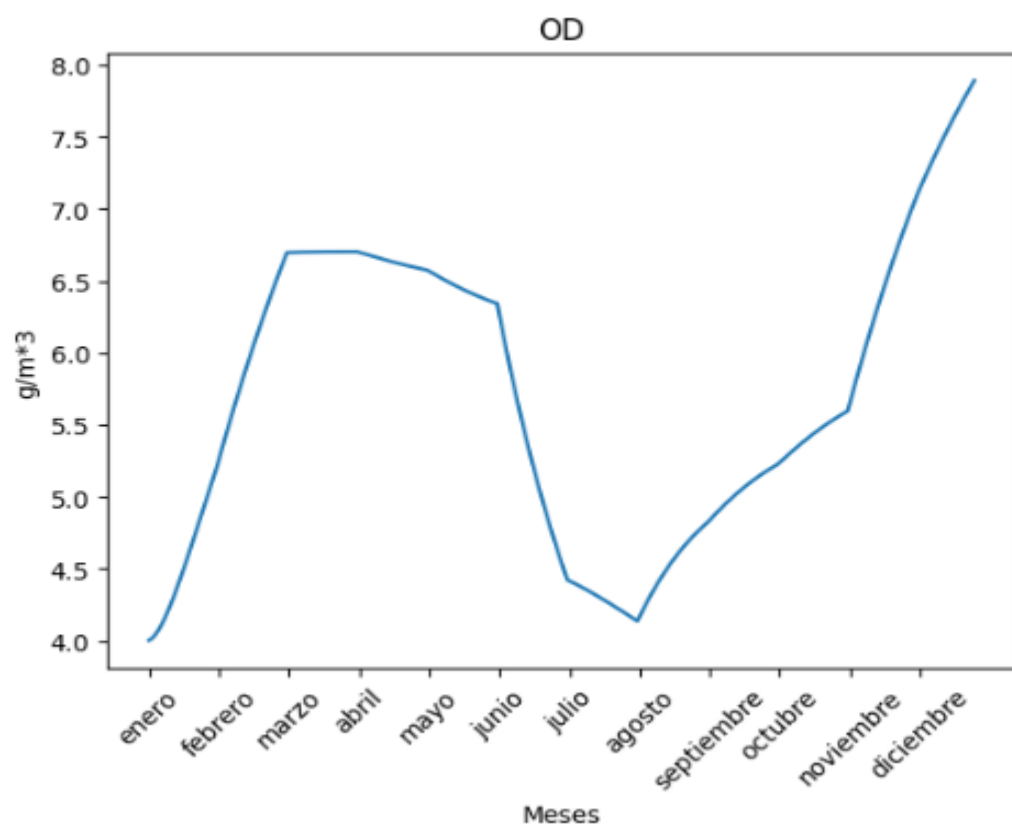
Punto C:



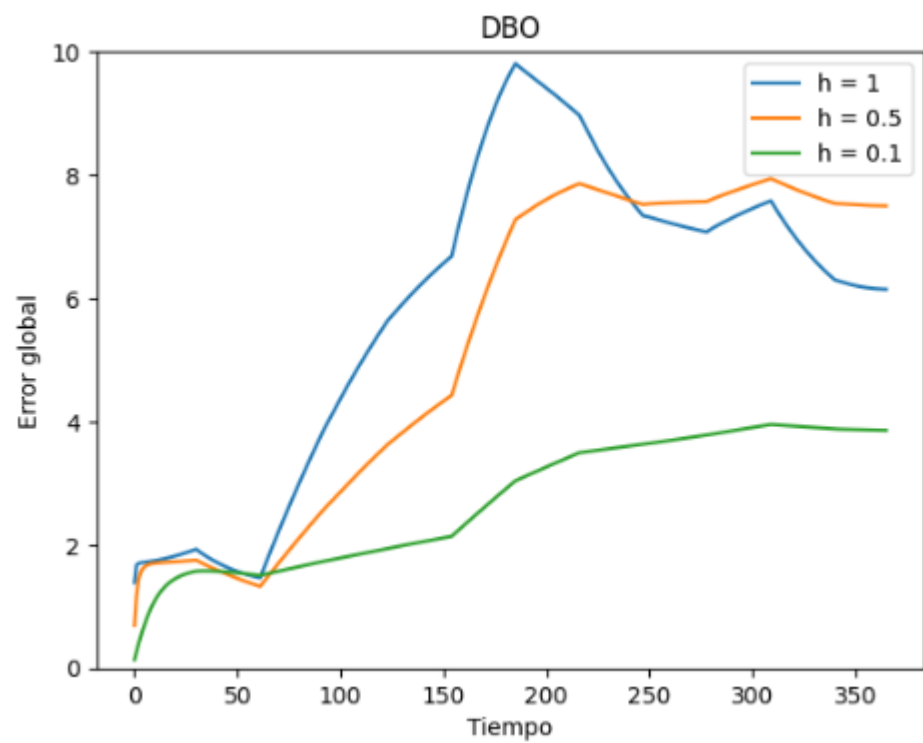
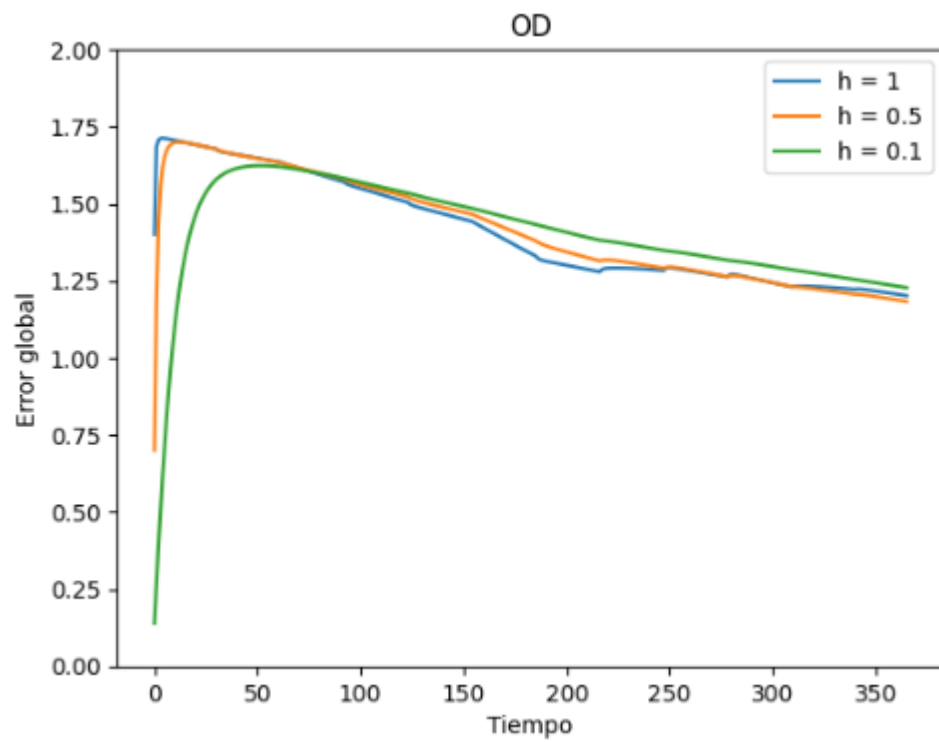


Punto E





Punto D:  
Euler explícito



Punto medio:

