Facultad de Ingeniería | Universidad de Buenos Aires

95.10 | Modelación numérica
75.12 | Análisis numérico I A
95.13 | Métodos matemáticos y numéricos

Trabajo Práctico #2 – Cuatrimestre 1 2025

Título del Trabajo Práctico

Grupo Nº 25	Matias Mendiola	110379
	Franco Altieri Lamas	105400
	Santiago Novaro	110938

Fecha	Correcciones / Observaciones	Docente

Calificación Final	Docente	Fecha

1 Introducción

Problema

El rompimiento de la ola y su avance a través de la zona de rotura resulta un proceso complejo que tiene una gran influencia en la morfodinámica costera al poner en movimiento gran cantidad de sedimentos en suspensión. El estudio de este proceso es de un interés indiscutible para los ingenieros costeros debido a que las zonas costeras, además de la importancia ambiental, son áreas que albergan a gran cantidad de población, y de ellas depende un gran porcentaje de sus usos y/o actividades económicas.

Uno de los fenómenos más importantes a modelar es el **decaimiento de la altura de ola luego del rompimiento**. A fin de representar este comportamiento, en este trabajo se implementa el modelo propuesto por **Dally, Dean y Dalrymple (1985)**, el cual plantea que la disipación de energía en la zona de rompiente es proporcional a la diferencia entre el flujo de energía actual y un flujo de energía estable hacia el cual tiende la ola.

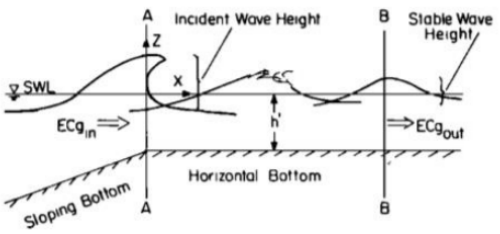


Fig. 1. Shelf beach idealization of the surf zone. Breaking is initiated at section A-A and continues until the stable wave condition is attained at section B-B.

Este modelo se basa en observaciones experimentales previas y describe la evolución de la energía mediante una **ecuación diferencial ordinaria (ODE)**, que permite estimar cómo decae la altura de la ola. Siendo esta la siguiente ecuación:

$$\frac{\partial ECg}{\partial x} = \frac{-K}{h}\left[ECg - ECg_s\right]$$

donde:

- ECg: Flujo energético promediado temporalmente e integrado a la profundidad (como lo proporciona la teoría de ondas lineales de aguas poco profundas).
- K: Coeficiente adimensional de decaimiento.
- h*: Profundidad del agua quieta.
- ECgs: Flujo energético de la ola estable que la ola rompiente trata de alcanzar.

Siendo los flujos energéticos definidos por:

$$ECg = H^2 \cdot (h^*)^{1/2} \quad ECg_s = \Gamma^2 \cdot (h^*)^{5/2}$$

2. Metodología

Para resolver la ecuación diferencial, se implementaron los siguientes esquemas numéricos:

a) Método de Euler (orden 1)

El método de Euler es un esquema explícito de primer orden que se basa en la aproximación:

$$EC_g^{n+1} = EC_g^n + dx \cdot f(x_n, EC_g^n)$$

donde:

$$f(x, EC_g) = -\frac{K}{h^*}(EC_g - EC_{gs})$$

b) Método predictor-corrector (orden 2)

Este esquema consiste en dos pasos: primero se predice el valor siguiente con un paso de Euler (predictor), y luego se corrige con el promedio de la pendiente en el punto actual y el punto predicho (corrector):
La fórmula del predictor es la misma utilizada en el método anterior, ya que se usa justamente ese método. Mientras que la fórmula del corrector utilizada es:

$$EC_g^{n+1} = EC_g^n + \frac{dx}{2}\left[f(x_n, EC_g^n) + f(x_{n+1}, \tilde{EC}_g^{n+1})\right]$$

c) Método de Runge-Kutta de orden 4 (RK4)

El método de Runge-Kutta de cuarto orden estima el valor siguiente con un promedio ponderado de cuatro pendientes evaluadas en distintos puntos intermedios:

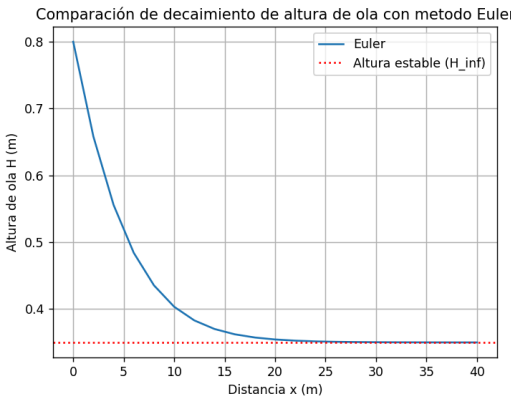
$$\begin{aligned}
k_1 &= dx \cdot f(x_n, EC_g^n) \\
k_2 &= dx \cdot f(x_n + \tfrac{dx}{2}, EC_g^n + \tfrac{k_1}{2}) \\
k_3 &= dx \cdot f(x_n + \tfrac{dx}{2}, EC_g^n + \tfrac{k_2}{2}) \\
k_4 &= dx \cdot f(x_n + dx, EC_g^n + k_3) \\
EC_g^{n+1} &= EC_g^n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}$$

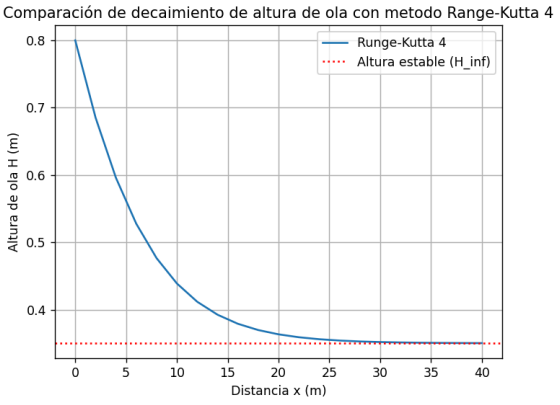
Una vez obtenido ECg(x) en cada punto del dominio, se calculó la altura de ola asociada mediante la relación:

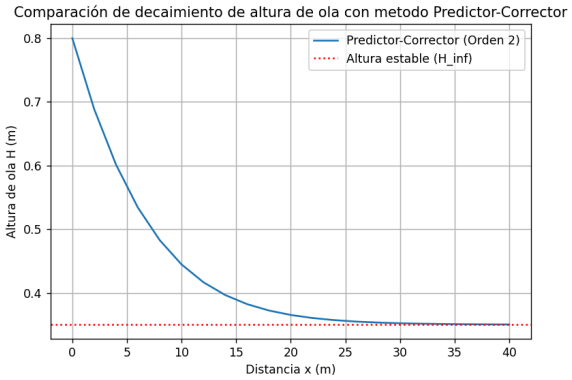
$$H(x) = \sqrt{\frac{EC_g(x)}{\sqrt{h^*}}}$$

Luego, se graficaron los resultados para cada método con el fin de analizar la evolución de la altura de ola y su convergencia hacia el valor teórico de equilibrio. Además, se estimaron errores de truncamiento y se analizó la estabilidad variando el paso espacial.
Para todo esto se utilizaron los valores constantes dados por el enunciado:

- H = 0.8 m (altura inicial de ola).
- h* = 1 m.
- K = 0.2.
- Γ = 0.35.







3. Resolución

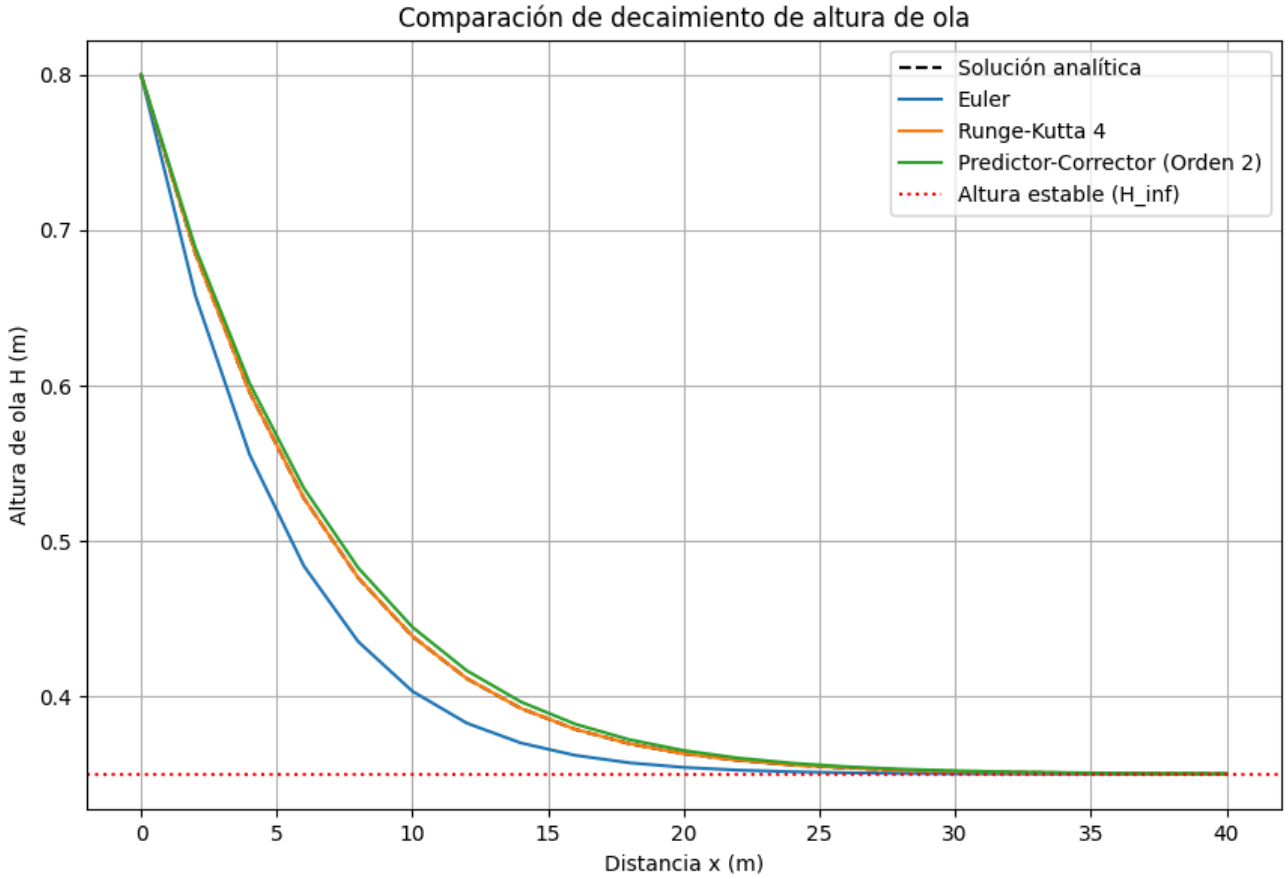
En el trabajo de Dally, Dean y Dalrymple (1985) se deduce una solución analítica

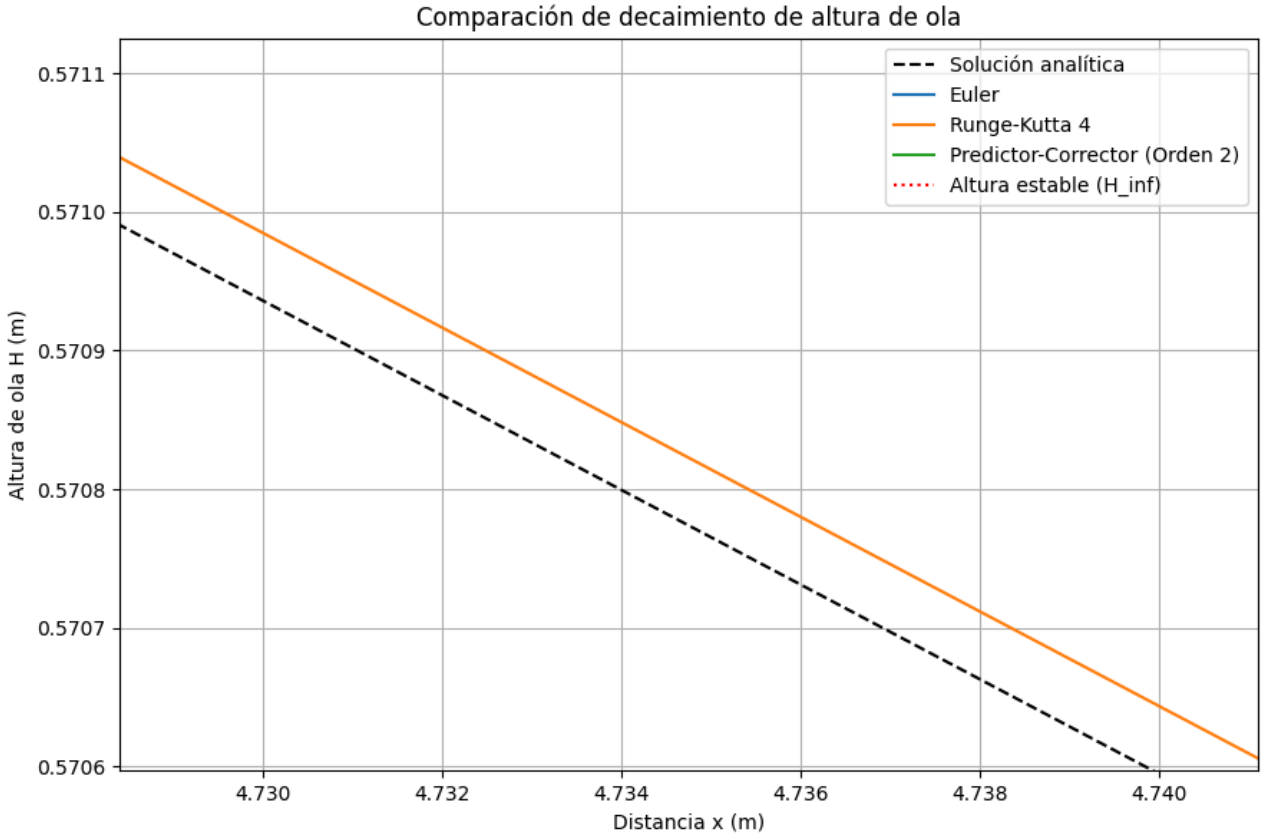
para el decaimiento de altura de ola post-rompimiento, bajo ciertas hipótesis simplificadas. Resolviendo la ecuación 9 del trabajo, dicha solución puede expresarse implícitamente como:

$$H(x) = \sqrt{\frac{\Gamma(h^*)^{\frac{5}{2}} + Ce^{\frac{-Kx}{h^*}}}{(h^*)^{\frac{1}{2}}}}$$

donde, C = 0.5175 con los valores de K, gamma y de h* pedidos por el enunciado. Simplificando queda:

$$H(x) = \sqrt{0.1225 + 0.5175^* \; e^{-0.2^*x}}$$





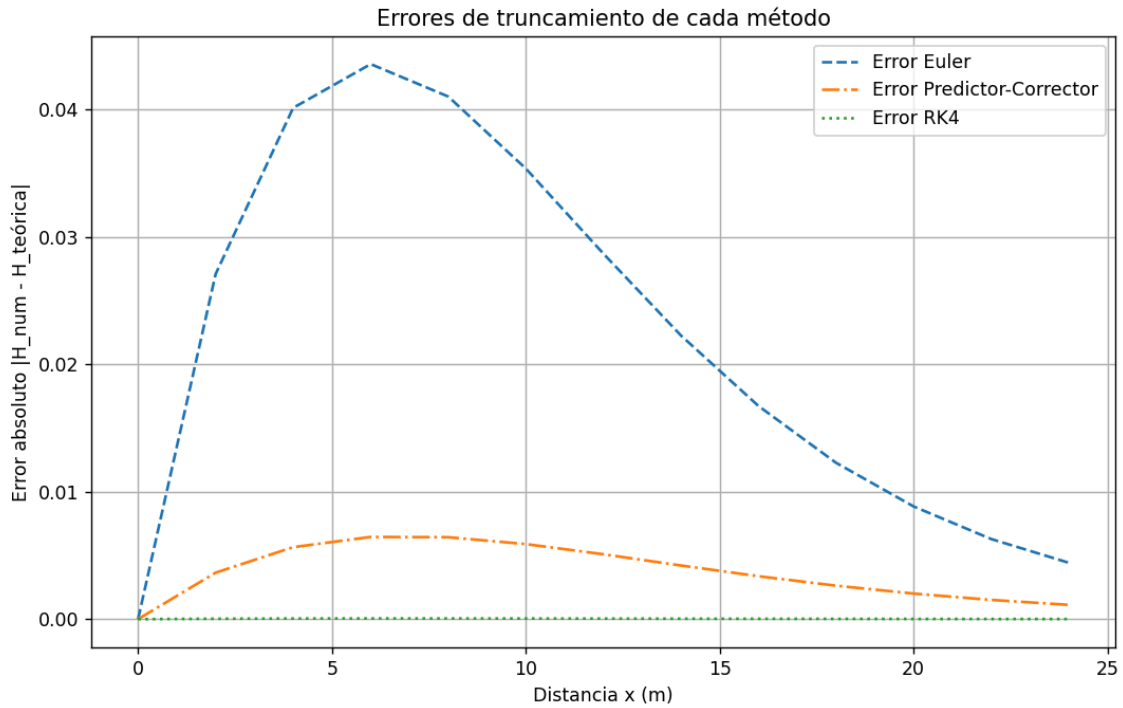
En la Figura se comparan los resultados obtenidos mediante tres métodos numéricos (Euler, Runge-Kutta de orden 4 y Predictor-Corrector de segundo orden) con la solución analítica propuesta por Dally, Dean y Dalrymple (1985), utilizando un paso espacial de 2m y una longitud total de simulación de 40m.

Se observa que las soluciones numéricas reproducen la tendencia general de la solución analítica, mostrando un decaimiento progresivo de la altura de ola. Sin embargo, el método de Euler tiene un error importante con respecto a la solución analítica, a diferencia de los otros 2 métodos, lo que genera una diferencia apreciable a lo largo del dominio. Esta diferencia seguramente se deba a la asunciones del método analítico y a las características propias de los métodos numéricos utilizados.
Además, se destaca que los métodos Runge-Kutta y Predictor-Corrector generan curvas prácticamente superpuestas, lo cual refleja una buena precisión y estabilidad para los parámetros de discretización utilizados. Por otro lado, el método de Euler presenta un mayor error numérico, siendo el que más se aleja tanto de la solución analítica como de los otros métodos. No obstante, todos los métodos convergen hacia la altura estable teórica, validando así la consistencia cualitativa del modelo implementado.

Con el objetivo de comparar los resultados obtenidos mediante los distintos métodos numéricos implementados frente a la solución teórica semi-empírica propuesta por Dally, Dean y Dalrymple (1985), se procedió a calcular el error absoluto en cada punto del dominio. Para ello, se utilizó la fórmula del error absoluto, definida como el valor absoluto de la diferencia entre la solución numérica y la teórica en cada

posición.

Tal como se realizó en la sección anterior al analizar el decaimiento de la altura de ola, se graficó el error absoluto a lo largo del dominio espacial para cada método. En el gráfico resultante puede observarse que, incluso empleando un paso relativamente grande (dx=2), el método de Runge-Kutta de cuarto orden presenta una excelente precisión, con errores prácticamente nulos. Por otro lado, el método de Euler evidencia los mayores desvíos respecto de la solución teórica, mientras que el método predictor-corrector (de segundo orden) mantiene un error acotado y considerablemente menor que Euler, aunque superior al de Runge-Kutta.

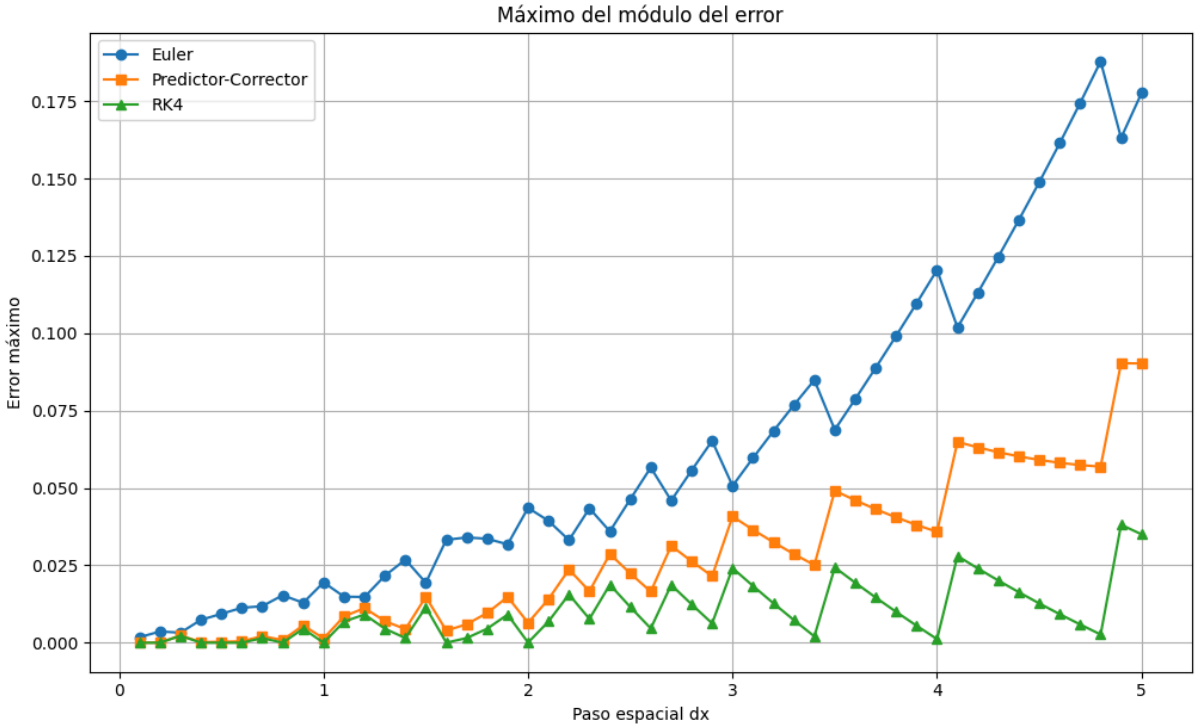


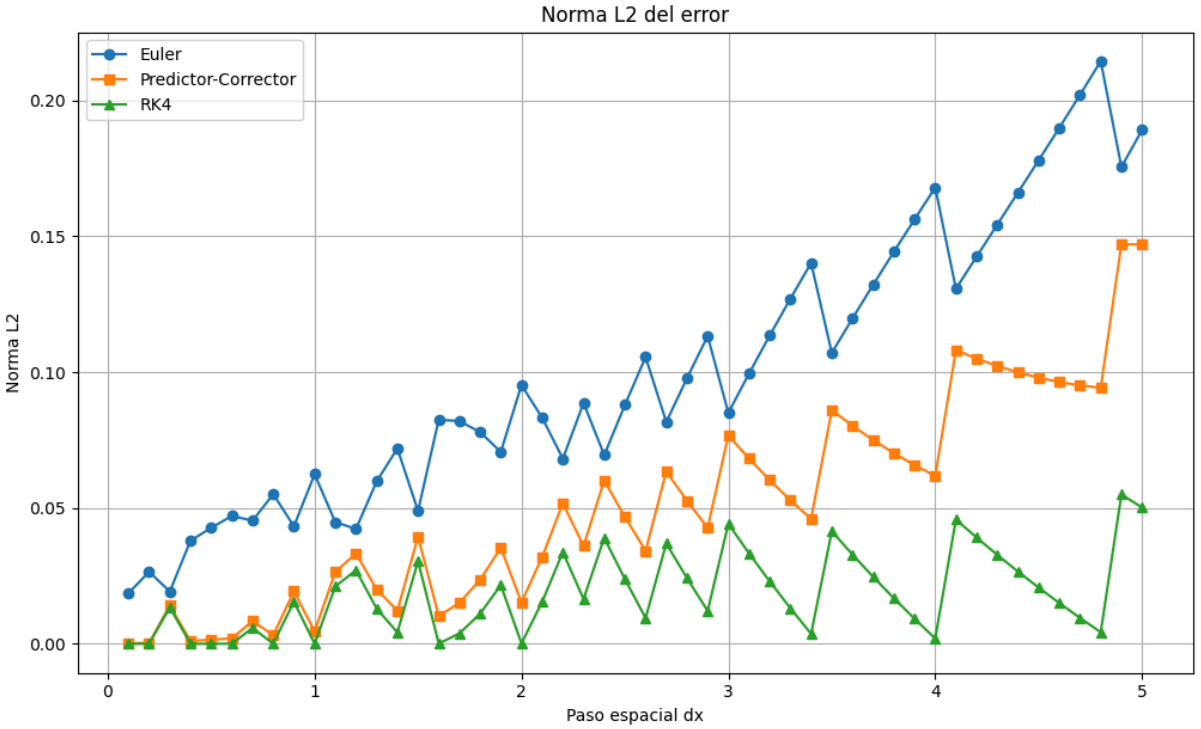
Al reducir el paso espacial dx, las diferencias entre los resultados obtenidos por los distintos métodos numéricos disminuyen notablemente, llegando en algunos casos a superponerse. Esto ocurre porque, al trabajar con una mayor resolución, el error asociado a cada método se reduce y todos tienden a aproximarse mejor a la solución teórica. En consecuencia, se puede considerar que el valor de dx elegido es suficientemente pequeño para representar de forma precisa el comportamiento que describe la ecuación diferencial.

Para analizar los umbrales de inestabilidad de los distintos métodos numéricos, haciendo un análisis tanto del promedio del módulo, el máximo del módulo y la norma, del vector de errores con la solución teórica, evaluando su evolución en función del paso espacial dx. En particular, se consideraron valores de dx entre 0.1 y 5, con incrementos de 0.1 en cada iteración.

Dado que se puede detectar el umbral de inestabilidad cuando los errores se vuelven erráticos y oscilan en mayoría, podemos decir que para el caso de Euler el umbral se encuentra en 2.3 y tanto para el caso del Predictor-Corrector y Runge-Kutta 4 su umbral se encuentra en un paso dx 2.7.


Promedio del módulo del error
Euler
Predictor-Corrector
RK4
Error promedio
Paso espacial dx
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
1
2
3
4
5


Máximo del módulo del error
Euler
Predictor-Corrector
RK4
Error máximo
Paso espacial dx
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
1
2
3
4
5



4. Conclusiones

Los resultados muestran que, si bien los métodos de orden superior (Runge-Kutta de orden 4 y Predictor-Corrector de orden 2) presentan errores significativamente menores para pasos pequeños, su desempeño se vuelve más inestable a medida que aumenta dx. Esta inestabilidad se manifiesta en errores crecientes y con mayor oscilación, reflejando la sensibilidad de estos métodos a pasos grandes.

Estos resultados permiten destacar que, al elegir un método numérico, no alcanza con enfocarse únicamente en la precisión que puede ofrecer, sino que también es clave considerar su estabilidad frente a distintos tamaños de paso. Como se vio en los gráficos para notar la estabilidad de los distintos métodos, Euler al tener menor precisión que los otros 2 métodos se vio menos favorecido y su umbral se vio afectado más prontamente que el de los demás métodos.

Referencias

Skiba, Y., 2005. Métodos y Esquemas Numéricos: Un Análisis Computacional. 1ra. edición. Dirección General de Publicaciones y Fomento Editorial, Universidad Nacional Autónoma de México, México, 440 pp. ISBN: 970-32-2023-1

JOURNAL OF GEOPHYSICAL RESEARCH, VOL. 90, NO. C6, PAGES 11,917-11,927, NOVEMBER 20, 1985

ANEXO: Códigos

Incorporar los códigos en el ANEXO y explicitar su utilización.

Código de graficación para los métodos de Euler, predictor-corrector 2 y Runge-Kutta 4,asi como tambien el codigo para mostrar el gráfico con los errores de truncamiento de los tres métodos

```
import numpy as np
import matplotlib.pyplot as plt

INITIAL_HEIGHT = 0.8
DEPTH_HEIGHT = 1.0
GAMMA = 0.35
K = 0.2
dx = 2
x_max = 24
N = int(x_max / dx) + 1
ECgs = GAMMA**2 * DEPTH_HEIGHT**(5/2)
ECg_0 = INITIAL_HEIGHT**2 * np.sqrt(DEPTH_HEIGHT)

def f(x, ECg):
    return -K / DEPTH_HEIGHT * (ECg - ECgs)

def euler_method():
    ECg = np.zeros(N)
    ECg[0] = ECg_0
    for n in range(N - 1):
        dECg_dx = f(n * dx, ECg[n])
        ECg[n + 1] = ECg[n] + dx * dECg_dx
    return ECg

def runge_kutta_4_method():
    x_vals = np.arange(0, x_max + dx, dx)
    ECg_vals = [ECg_0]
    for i in range(1, len(x_vals)):
        x_n = x_vals[i-1]
        ECg_n = ECg_vals[-1]

        k1 = dx * f(x_n, ECg_n)
        k2 = dx * f(x_n + dx/2, ECg_n + k1/2)
        k3 = dx * f(x_n + dx/2, ECg_n + k2/2)
```

```
        k4 = dx * f(x_n + dx, ECg_n + k3)
        ECg_next = ECg_n + (k1 + 2*k2 + 2*k3 + k4) / 6
        ECg_vals.append(ECg_next)
    return np.array(ECg_vals)

def predictor_corrector_method():
    x_vals = np.arange(0, x_max + dx, dx)
    ECg_vals = [ECg_0]
    for i in range(1, len(x_vals)):
        x_n = x_vals[i-1]
        ECg_n = ECg_vals[-1]

        predictor = ECg_n + dx * f(x_n, ECg_n)
        corrector = ECg_n + (dx/2) * (f(x_n, ECg_n) + f(x_n + dx,
predictor))

        ECg_vals.append(corrector)
    return np.array(ECg_vals)


def main():
    x = np.linspace(0, x_max, N)

    H_inf = GAMMA * DEPTH_HEIGHT
    H_analytical = []
    for i in range(len(x)):
        H = ((0.1225 + 0.5175*np.exp(-0.2*x[i]))**(1/2))
        H_analytical.append(H)

    ECg_euler = euler_method()
    ECg_rk4 = runge_kutta_4_method()
    ECg_pc = predictor_corrector_method()

    H_euler = np.sqrt(ECg_euler / np.sqrt(DEPTH_HEIGHT))
    H_rk4 = np.sqrt(ECg_rk4 / np.sqrt(DEPTH_HEIGHT))
    H_pc = np.sqrt(ECg_pc / np.sqrt(DEPTH_HEIGHT))

    print("x (m) | H_analitycal | H_euler | H_rk4 | H_pc")
    for i in range(0, N, N//10):
        print(f"{x[i]:.2f} | {H_teorica[i]:.4f} | {H_euler[i]:.4f}
| {H_rk4[i]:.4f} | {H_pc[i]:.4f}")

    plt.plot(x, H_analytical, 'k--', label='Solución analítica')
```

```
    plt.plot(x, H_euler, label='Euler')
    plt.plot(x, H_rk4, label='Runge-Kutta 4')
    plt.plot(x, H_pc, label='Predictor-Corrector (Orden 2)')
    plt.axhline(y=H_inf, color='r', linestyle=':', label='Altura
estable (H_inf)')

    plt.xlabel('Distancia x (m)')
    plt.ylabel('Altura de ola H (m)')
    plt.title('Comparación de decaimiento de altura de ola')
    plt.legend()
    plt.grid(True)
    plt.show()

    H_analytical = np.array(H_analytical)

    error_euler = np.abs(H_euler - H_analytical)
    error_pc = np.abs(H_pc - H_analytical)
    error_rk4 = np.abs(H_rk4 - H_analytical)



    print("\n x (m) | Error Euler | Error PC | Error RK4")
    for i in range(0, N, N // 10):
        print(f"{x[i]:.4f} | {error_euler[i]:.4f} |
{error_pc[i]:.4f} | {error_rk4[i]:.4f}")

    plt.figure(figsize=(10, 6))
    plt.plot(x, error_euler, '--', label='Error Euler')
    plt.plot(x, error_pc, '-.', label='Error Predictor-Corrector')
    plt.plot(x, error_rk4, ':', label='Error RK4')
    plt.xlabel('Distancia x (m)')
    plt.ylabel('Error absoluto |H_num - H_teórica|')
    plt.title('Errores de truncamiento de cada método')
    plt.legend()
    plt.grid(True)
    plt.show()

if __name__ == "__main__":
    main()
```

Utilización del código en la resolución del trabajo

Para resolver los distintos puntos planteados en el trabajo práctico se desarrolló un código en lenguaje Python, empleando las librerías **NumPy** y **Matplotlib**. El objetivo principal fue implementar diferentes métodos numéricos para aproximar la solución de la ecuación diferencial ordinaria que describe la evolución del flujo energético de las olas una vez que rompen, según el modelo propuesto por Dally, Dean y Dalrymple (1985).El código comienza definiendo los parámetros físicos del problema, como la altura inicial de la ola, la profundidad constante del fondo marino, el parámetro empírico de rompimiento y el coeficiente de disipación de energía. También se establece el paso espacial Δx que se extiende hasta una distancia de 24 metros.Luego se calcula el flujo energético inicial ECg(0) y el flujo energético estable ECgs que es el valor hacia el cual tiende el sistema conforme la ola se propaga y disipa energía. Se define una función `f(x, ECg)` que representa la derivada de ECg respecto de x, es decir, la ecuación diferencial que se quiere resolver numéricamente.

Para resolver esta ecuación se implementaron tres esquemas numéricos. El primero es el método de Euler, un método explícito de primer orden que estima el valor siguiente de la función utilizando únicamente la pendiente en el punto actual. El segundo esquema es el método Predictor-Corrector de orden dos,que mejora la precisión al realizar un paso predictor (estimación con Euler) y luego corregirlo promediando las pendientes al inicio y al final del paso. El tercer esquema es el método de Runge-Kutta de orden cuatro, que evalúa la función derivada en cuatro puntos dentro del intervalo para obtener una estimación mucho más precisa de la solución.Además de resolver la ecuación diferencial mediante estos métodos, el código calcula la solución analítica teórica esperada para el problema, basada en una expresión cerrada deducida a partir del modelo, que sirve como referencia para comparar los resultados numéricos. Luego, se traduce el flujo energético ECg(x) obtenido con cada método en términos de altura de ola H(x).El programa imprime una tabla comparativa de valores y genera gráficos que muestran la evolución de la altura de ola en función de la distancia para cada uno de los métodos, junto con la solución analítica. También se estima el error de truncamiento de cada método numérico con respecto a la solución exacta, y se grafican estos errores para visualizar el comportamiento y la precisión de cada método.

En conclusión, el código fue utilizado para resolver la ecuación diferencial del problema por métodos numéricos, contrastar las soluciones obtenidas con la solución analítica y analizar cuantitativamente los errores cometidos por cada esquema. De esta forma, se cumplieron los objetivos de los puntos a, c y d del trabajo práctico.

CÓDIGO UMBRAL DE INESTABILIDAD

```
import numpy as np
import matplotlib.pyplot as plt

INITIAL_HEIGHT = 0.8
DEPTH_HEIGHT = 1.0
GAMMA = 0.35
K = 0.2
x_max = 24
N = int(x_max / dx) + 1
H_inf = GAMMA * DEPTH_HEIGHT

ECgs = GAMMA**2 * DEPTH_HEIGHT**(5/2)
ECg_0 = INITIAL_HEIGHT**2 * np.sqrt(DEPTH_HEIGHT)

def f(x, ECg):
    return -K / DEPTH_HEIGHT * (ECg - ECgs)

def euler_method(dx, N):

    ECg = np.zeros(N)
    ECg[0] = ECg_0
    for n in range(N - 1):
        dECg_dx = f(n * dx, ECg[n])
        ECg[n + 1] = ECg[n] + dx * dECg_dx
    return ECg

def runge_kutta_4_method(dx, N):
    x_vals = np.linspace(0, x_max, N)
    ECg_vals = [ECg_0]
    for i in range(1, len(x_vals)):
        x_n = x_vals[i-1]
        ECg_n = ECg_vals[-1]

        k1 = dx * f(x_n, ECg_n)
        k2 = dx * f(x_n + dx/2, ECg_n + k1/2)
        k3 = dx * f(x_n + dx/2, ECg_n + k2/2)
        k4 = dx * f(x_n + dx, ECg_n + k3)
        ECg_next = ECg_n + (k1 + 2*k2 + 2*k3 + k4) / 6
        ECg_vals.append(ECg_next)
    return np.array(ECg_vals)
```

```
def predictor_corrector_method(dx, N):
    x_vals = np.linspace(0, x_max, N)
    ECg_vals = [ECg_0]
    for i in range(1, len(x_vals)):
        x_n = x_vals[i-1]
        ECg_n = ECg_vals[-1]

        predictor = ECg_n + dx * f(x_n, ECg_n)
           corrector = ECg_n + (dx/2) * (f(x_n, ECg_n) + f(x_n + dx,
predictor))

        ECg_vals.append(corrector)
    return np.array(ECg_vals)


def main():
    dx_ini = 0.1
    dx_fin = 3.0
    dx_step = 0.1
    errores = {"dx": [], "Euler": [], "PC": [], "RK4": []}
    for dx in np.arange(dx_ini, dx_fin + dx_step, dx_step):
        x = np.linspace(0, x_max, N)
        H_analytical = []
        for i in range(len(x)):
            H = ((0.1225 + 0.5175*np.exp(-0.2*x[i]))**(1/2))
            H_analytical.append(H)

        ECg_euler = euler_method(dx, N)
        ECg_rk4 = runge_kutta_4_method(dx, N)
        ECg_pc = predictor_corrector_method(dx, N)

        H_euler = np.sqrt(ECg_euler / np.sqrt(DEPTH_HEIGHT))
        H_rk4 = np.sqrt(ECg_rk4 / np.sqrt(DEPTH_HEIGHT))
        H_pc = np.sqrt(ECg_pc / np.sqrt(DEPTH_HEIGHT))

        H_analytical = np.array(H_analytical)

        error_euler = np.abs(H_euler - H_analytical)
        error_pc = np.abs(H_pc - H_analytical)
        error_rk4 = np.abs(H_rk4 - H_analytical)
        errores["dx"].append(dx)
        errores["Euler"].append(error_euler[-1])
        errores["PC"].append(error_pc[-1])
```

```
        errores["RK4"].append(error_rk4[-1])

    plt.figure(figsize=(10, 6))
            plt.plot(errores["dx"],   errores["Euler"],   label="Euler",
marker='o')
     plt.plot(errores["dx"], errores["PC"], label="Predictor-Corrector",
marker='s')
    plt.plot(errores["dx"], errores["RK4"], label="RK4", marker='^')
    plt.xlabel("Paso espacial dx")
    plt.ylabel("Error absoluto en x = "+ str(x_max))
    plt.title("Error final en función del paso dx")
    plt.grid(True)
    plt.legend()
    plt.show()

if __name__ == "__main__":
    main()
```

```python
def show_methods():
    dx = 2
    N = int(x_max / dx) + 1
    x = np.linspace(0, x_max, N)

    H_inf = GAMMA * DEPTH_HEIGHT
    H_analytical = ((0.1225 + 0.5175 * np.exp(-0.2 * x))**0.5)

    ECg_euler = euler_method(dx, N)
    ECg_rk4 = runge_kutta_4_method(dx, N)
    ECg_pc = predictor_corrector_method(dx, N)

    H_euler = np.sqrt(ECg_euler / np.sqrt(DEPTH_HEIGHT))
    H_rk4 = np.sqrt(ECg_rk4 / np.sqrt(DEPTH_HEIGHT))
    H_pc = np.sqrt(ECg_pc / np.sqrt(DEPTH_HEIGHT))

    print("x (m) | H_analytical | H_euler | H_rk4 | H_pc")
    for i in range(0, N, N // 10):
        print(f"{x[i]:.2f} | {H_analytical[i]:.4f} | {H_euler[i]:.4f} |
{H_rk4[i]:.4f} | {H_pc[i]:.4f}")

    plt.plot(x, H_analytical, 'k--', label='Solución analítica')
    plt.plot(x, H_euler, label='Euler')
    plt.plot(x, H_rk4, label='Runge-Kutta 4')
    plt.plot(x, H_pc, label='Predictor-Corrector (Orden 2)')
        plt.axhline(y=H_inf,  color='r',  linestyle=':',  label='Altura
estable (H_inf)')

    plt.xlabel('Distancia x (m)')
    plt.ylabel('Altura de ola H (m)')
    plt.title('Comparación de decaimiento de altura de ola')
    plt.legend()
    plt.grid(True)
    plt.show()

    error_euler = np.abs(H_euler - H_analytical)
    error_pc = np.abs(H_pc - H_analytical)
    error_rk4 = np.abs(H_rk4 - H_analytical)

    print("\n x (m) | Error Euler | Error PC | Error RK4")
    for i in range(0, N, N // 10):
        print(f"{x[i]:.2f} | {error_euler[i]:.5f} | {error_pc[i]:.5f} |
{error_rk4[i]:.5f}")
```

```
    plt.figure(figsize=(10, 6))
    plt.plot(x, error_euler, '--', label='Error Euler')
    plt.plot(x, error_pc, '-.', label='Error Predictor-Corrector')
    plt.plot(x, error_rk4, ':', label='Error RK4')
    plt.xlabel('Distancia x (m)')
    plt.ylabel('Error absoluto |H_num - H_analytical|')
    plt.title('Errores de truncamiento de cada método')
    plt.legend()
    plt.grid(True)
    plt.show()

def show_inestability_threshold():
    dx_ini = 0.1
    dx_fin = 5.0
    dx_step = 0.1
    dx_values = []

    errores_mean = {"Euler": [], "PC": [], "RK4": []}
    errores_max = {"Euler": [], "PC": [], "RK4": []}
    errores_l2   = {"Euler": [], "PC": [], "RK4": []}

    for dx in np.arange(dx_ini, dx_fin + dx_step, dx_step):
        N = int(x_max / dx) + 1
        x = np.linspace(0, x_max, N)
        dx_values.append(dx)

        H_teorica = ((0.1225 + 0.5175 * np.exp(-0.2 * x))**0.5)

        ECg_euler = euler_method(dx, N)
        ECg_pc = predictor_corrector_method(dx, N)
        ECg_rk4 = runge_kutta_4_method(dx, N)

        H_euler = np.sqrt(ECg_euler / np.sqrt(DEPTH_HEIGHT))
        H_pc = np.sqrt(ECg_pc / np.sqrt(DEPTH_HEIGHT))
        H_rk4 = np.sqrt(ECg_rk4 / np.sqrt(DEPTH_HEIGHT))

        error_euler = np.abs(H_euler - H_teorica)
        error_pc = np.abs(H_pc - H_teorica)
        error_rk4 = np.abs(H_rk4 - H_teorica)

        errores_mean["Euler"].append(np.mean(error_euler))
        errores_mean["PC"].append(np.mean(error_pc))
```

```
        errores_mean["RK4"].append(np.mean(error_rk4))

        errores_max["Euler"].append(np.max(error_euler))
        errores_max["PC"].append(np.max(error_pc))
        errores_max["RK4"].append(np.max(error_rk4))

        errores_l2["Euler"].append(np.linalg.norm(error_euler))
        errores_l2["PC"].append(np.linalg.norm(error_pc))
        errores_l2["RK4"].append(np.linalg.norm(error_rk4))

    plt.figure(figsize=(10, 5))
            plt.plot(dx_values,   errores_mean["Euler"],   label="Euler",
marker='o')
                            plt.plot(dx_values,       errores_mean["PC"],
label="Predictor-Corrector", marker='s')
    plt.plot(dx_values, errores_mean["RK4"], label="RK4", marker='^')

    plt.title("Promedio del módulo del error")
    plt.xlabel("Paso espacial dx")
    plt.ylabel("Error promedio")
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
    plt.show()

    plt.figure(figsize=(10, 5))
            plt.plot(dx_values,   errores_max["Euler"],   label="Euler",
marker='o')
     plt.plot(dx_values, errores_max["PC"], label="Predictor-Corrector",
marker='s')
    plt.plot(dx_values, errores_max["RK4"], label="RK4", marker='^')


    plt.title("Máximo del módulo del error")
    plt.xlabel("Paso espacial dx")
    plt.ylabel("Error máximo")
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
    plt.show()

    plt.figure(figsize=(10, 5))
    plt.plot(dx_values, errores_l2["Euler"], label="Euler", marker='o')
```

```
     plt.plot(dx_values, errores_l2["PC"], label="Predictor-Corrector",
marker='s')
    plt.plot(dx_values, errores_l2["RK4"], label="RK4", marker='^')

    plt.title("Norma L2 del error")
    plt.xlabel("Paso espacial dx")
    plt.ylabel("Norma L2")
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
    plt.show()
```

Análisis del umbral de inestabilidad numérica

Con el objetivo de detectar el umbral de inestabilidad de los métodos numéricos implementados, se elaboró un script que permite variar sistemáticamente el paso espacial Δx y observar cómo esto afecta la precisión de los resultados obtenidos por los tres esquemas considerados: Euler, Predictor-Corrector de orden dos y Runge-Kutta cuatro. El umbral de inestabilidad se define como el valor máximo de Δx a partir del cual el método pierde precisión de forma significativa o se vuelve completamente inestable.El análisis se realiza en un bucle que recorre distintos valores de Δx, comenzando en 0.1 y aumentando de a 0.1 hasta un máximo de 5.0. Para cada uno de estos valores, se calcula la cantidad de puntos necesarios para cubrir el dominio de simulación (de 0 a 24 metros), se resuelve la ecuación con cada método y se traduce el flujo energético resultante en términos de altura de ola. A continuación, se calcula el error absoluto entre la altura de ola numérica y la solución teórica en cada punto, y se almacenan los errores en los puntos del dominio.Una vez recolectados los errores para todos los valores de Δx, se calcula el promedio, el máximo y la norma, y se genera un gráfico que muestra la evolución del error final en función del tamaño del paso.

Link al colab:
TP2.ipynb