



Universidad Técnica Federico Santa María

IP468 - FUNDS. DE DINÁMICA DE FLUIDOS COMPUTACIONAL
PHD ROMAIN GERS

TAREA DE DIFERENCIAS FINITAS 2023:

"Ecuación de Ondas en todos sus estados."

Autor: Javiera Espinoza M.¹

23 de Octubre, 2023

¹Correo: javieraespinozam@usm.cl - Rol: 201941036-9

Índice

Índice de Figuras	1
Índice de Tablas	2
1 Introducción	3
2 Ecuación de convección 1D	3
2.1 Perfil inicial	3
2.2 Solución analítica	4
2.3 Backward, Centered and Forward en el espacio, Euler explícito en el tiempo	5
2.3.1 Backward	5
2.3.2 Centered	6
2.3.3 Forward	7
2.3.4 Preguntas asociadas	8
2.4 Lax - Friedrichs	8
2.4.1 Preguntas asociadas	9
2.5 Lax - Wendroff	9
2.5.1 Preguntas asociadas	10
2.6 Leap - Frog	10
2.6.1 Preguntas asociadas	10
2.7 Centrado en el espacio y Euler implícito en el tiempo	11
2.8 Centrado en el espacio y Crank - Nicolson en el tiempo	11
2.9 Análisis	12
2.9.1 Errores métodos explícitos	12
2.9.2 Errores métodos implícitos	13
3 Ecuación de onda escalar 1D	14
3.0.1 Análisis	17
4 Ecuación de ondas escalar 1D de orden 4	17
4.1 Discretización	17
4.2 Análisis	20
4.3 Comparación entre métodos	20
Referencias	21
5 Anexos	21
5.1 Código computacional	21

Índice de Figuras

1 Perfil inicial bajo las condiciones dadas.	4
2 Solución analítica para distintos tiempos.	4
3 Modelación backward $Cr = 0.5$	5

4	Modelación centered Cr=0.5.	6
5	Modelación forward Cr = 0.5.	7
6	Modelación Lax - Friedrichs Cr = 0.5.	8
7	Modelación Lax - Wendroff C = 0.5.	9
8	Modelación Leap - Frog C = 0.5.	10
9	Modelación Centrada en el espacio y EI en el tiempo.	11
10	Modelación Centrada en el espacio y CN en el tiempo Cr = 0.5.	12
11	Errores métodos explícitos para distintos Cr.	13
12	Errores métodos implícitos para distintos Cr.	14
13	Esquema de Leap Frog de orden 2 para Cr = 0.01.	15
14	Esquema de Leap Frog de orden 2 para Cr = 0.5.	16
15	Esquema de Leap Frog de orden 2 para Cr = 1	16
16	Ecuación de onda escalar con Cr = 0.01.	19
17	Ecuación de onda escalar con Cr = 0.5.	19
18	Ecuación de onda escalar con Cr = 1.	20
19	Ecuación de onda escalar con Cr = 0.5.	21

Índice de Tablas

1	Tabla comparativa de errores.	14
---	---------------------------------------	----

1 Introducción

La ecuación de ondas permite describir la propagación de una onda. En esta tarea se trata de una onda de presión acústica P . A una dimensión se escribe en función de la velocidad del sonido c_0 :

$$\frac{\partial^2 P}{\partial t^2} - c_0^2 \frac{\partial^2 P}{\partial x^2} = 0 \quad (1)$$

Jean Le Rond d'Alembert (1717-1783) fue el primero a descubrir esta ecuación estudiando cuerdas vibrantes. Es el autor de "L'Encyclopédie o Dictionnaire raisonné des sciences, des arts et des métiers", una de las más grandes obras del siglo XVIII. Redactó cerca de 1700 artículos, la inmensa mayoría de los artículos sobre las matemáticas, la astronomía y la física. Esta ecuación se puede deducir de la linearización de las ecuaciones de Navier-Stokes:

$$\frac{\partial \rho}{\partial t} + \rho_0 \frac{\partial \nu}{\partial x} = 0 \quad (2)$$

$$\rho_0 \frac{\partial \nu}{\partial t} = - \frac{\partial P}{\partial x} \quad (3)$$

$$P = c_0^2 \rho \quad (4)$$

donde ρ_0 es la densidad, ρ la fluctuación de densidad y v la velocidad acústica. D'Alembert mostró que la ecuación se puede descomponer de la siguiente forma:

$$\left(\frac{\partial}{\partial t} + c_0 \frac{\partial}{\partial x} \right) \left(\frac{\partial}{\partial t} - c_0 \frac{\partial}{\partial x} \right) P = 0 \quad (5)$$

lo que indica que la solución representa dos ondas que se propagan en sentidos contrarios. Mirando entonces en un solo sentido, la ecuación de propagación de onda se escribe:

$$\frac{\partial P}{\partial t} + c_0 \frac{\partial P}{\partial x} = 0 \quad (6)$$

La solución exacta de esta ecuación de advección 1D con una condición inicial $P(x, t=0) = P_0(x)$ es: $P(x, t) = P_0(x - c_0 t)$.

2 Ecuación de convección 1D

2.1 Perfil inicial

Suponiendo un dominio de propagación:

$$x \in [-2\pi, 2\pi]$$

$$\Delta x = 0.1\pi$$

$$\Delta t = 0.1\pi$$

$$P_0(x) = \sin(x) \text{ si } -\pi \leq x \leq \pi, 0 \text{ si no}$$

Con ello, se obtiene el siguiente perfil:

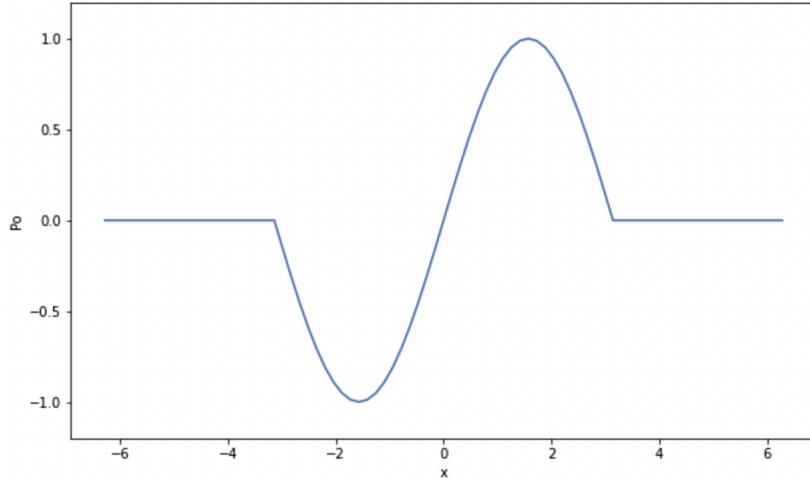


Figura 1: Perfil inicial bajo las condiciones dadas.

2.2 Solución analítica

Dado que se conoce el perfil inicial, es posible determinar la solución analítica en el tiempo. Aplicando los parámetros del perfil inicial, y utilizando 81 nodos en el tiempo, se tiene:

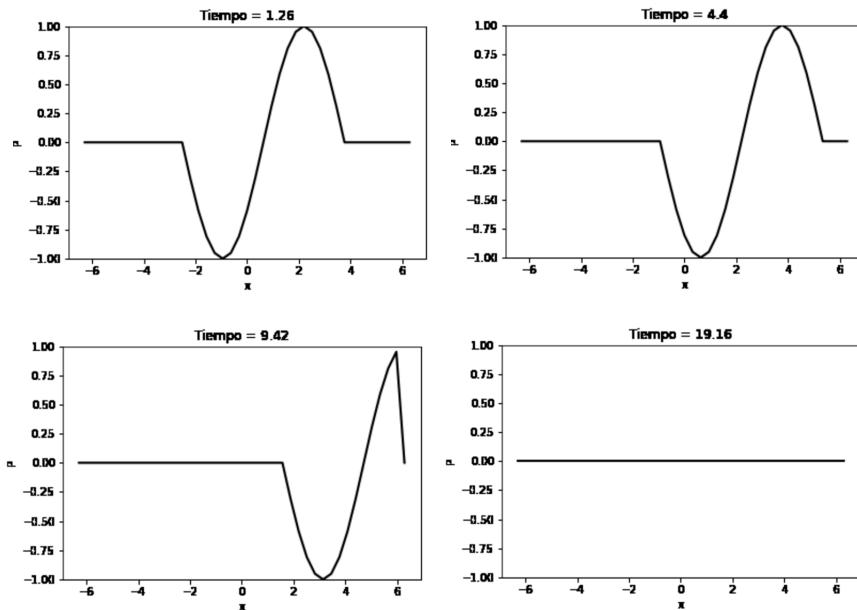


Figura 2: Solución analítica para distintos tiempos.

2.3 Backward, Centered and Forward en el espacio, Euler explícito en el tiempo

2.3.1 Backward

Para las diferencias retrasadas se considera el nodo actual y el nodo ubicado en la posición anterior en el espacio, ambos en el mismo instante de tiempo. En cuanto a la integración temporal, se emplea el método de Euler explícito. Esto significa que se utiliza toda la información del tiempo actual para calcular el estado en el tiempo siguiente. Así:

$$\frac{P_i^{n+1} - P_i^n}{\Delta t} = -c_0 \frac{P_i^n - P_{i-1}^n}{\Delta x} \quad (7)$$

$$P_i^{n+1} = \frac{-c_0 \Delta t}{\Delta x} (P_i^n - P_{i-1}^n) + P_i^n \quad (8)$$

$$P_i^{n+1} = \frac{-c_0 \Delta t}{\Delta x} (P_i^n) + \frac{c_0 \Delta t}{\Delta x} (P_{i-1}^n) + P_i^n \quad (9)$$

Con la discretización completada, se procede estudiar la estabilidad mediante el Courant-Friedrichs-Lowy:

$$CFL = \frac{c \cdot \Delta t}{\Delta x} \quad (10)$$

Reemplazando por los datos conocidos se tiene:

$$CFL = \frac{0.5[m/s] \cdot 0.1\pi}{0.1\pi} = 0.5 \quad (11)$$

Lo cual indica que el esquema es estable para las condiciones actuales.

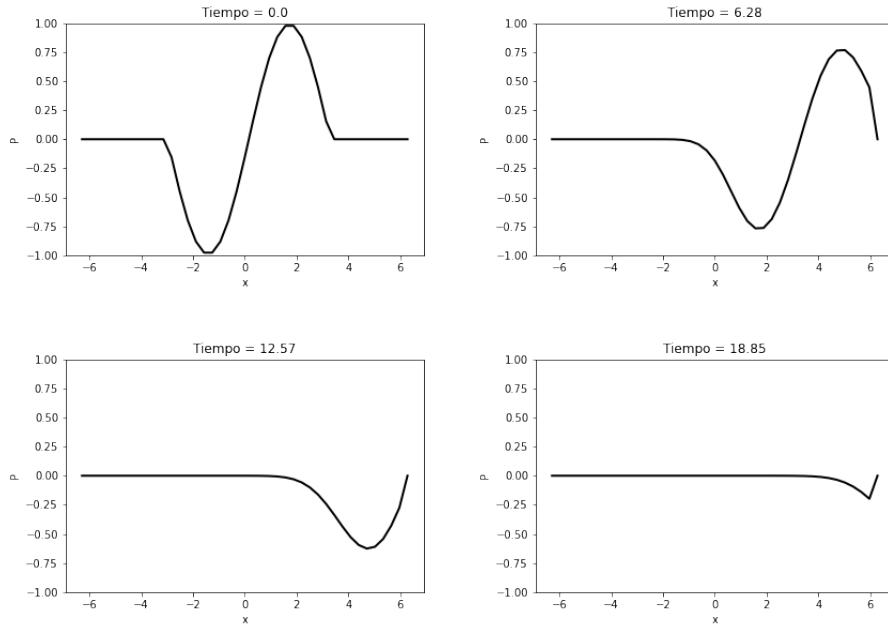


Figura 3: Modelación backward Cr = 0.5.

2.3.2 Centered

Para las diferencias centradas se trabaja con el nodo en la posición anterior y el nodo ubicado en la posición posterior en el espacio. Para la integración temporal, se emplea nuevamente el método de Euler explícito. Así:

$$\frac{P_i^{n+1} - P_i^n}{\Delta t} = -c_0 \frac{P_{i+1}^n - P_{i-1}^n}{2\Delta x} \quad (12)$$

$$P_i^{n+1} = \frac{-c_0 \Delta t}{2\Delta x} (P_{i+1}^n - P_{i-1}^n) + P_i^n \quad (13)$$

$$P_i^{n+1} = \frac{c_0 \Delta t}{2\Delta x} (P_{i-1}^n) + P_i^n - \frac{c_0 \Delta t}{2\Delta x} (P_{i+1}^n) \quad (14)$$

Utilizando la ecuación 13 y reescribiendo el factor $\frac{c_0 \Delta t}{2\Delta x}$ como el número de Courant, se reescribe la expresión de la forma:

$$P_i^{n+1} = -0.5Cr(P_{i+1}^n - P_{i-1}^n) + P_i^n \quad (15)$$

Aplicándole las condiciones de estabilidad de Von Neumann, reemplazando el término V_i^n por $g^n e^{ij\theta}$ y arreglando matemáticamente la expresión se tiene:

$$|g|^2 = 1 + Cr^2 \operatorname{sen}^2 \theta \quad (16)$$

Dado que $|g|^2 \geq 1$ para cualquier número en los reales, el esquema es incondicionalmente inestable. Los resultados de la respectiva simulación son:

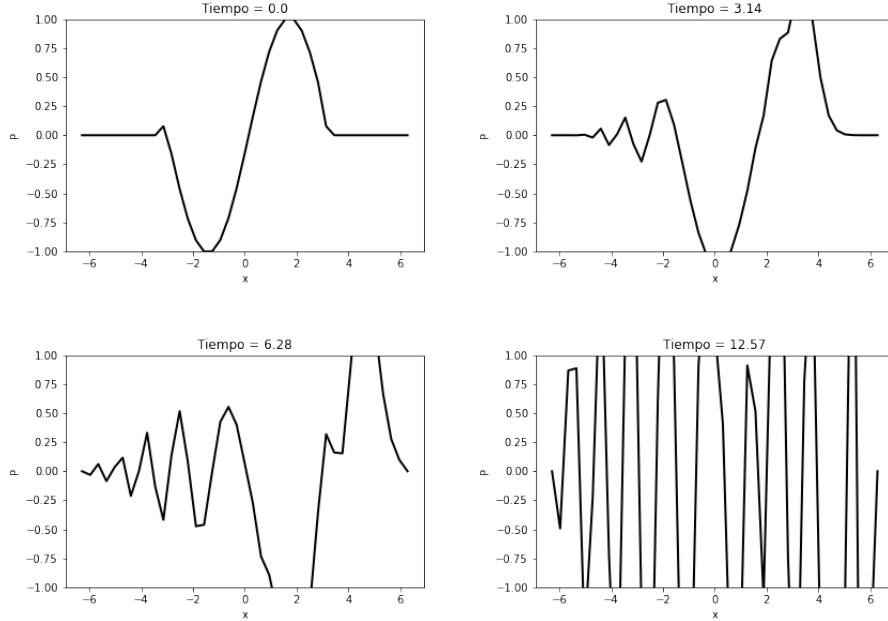


Figura 4: Modelación centered Cr=0.5.

2.3.3 Forward

Para las diferencias adelantadas se procede a trabajar con el nodo actual y el nodo ubicado en la posición posterior en el espacio. Para la integración temporal, se emplea nuevamente el método de Euler explícito. Así:

$$\frac{P_i^{n+1} - P_i^n}{\Delta t} = -c_0 \frac{P_{i+1}^n - P_i^n}{\Delta x} \quad (17)$$

$$P_i^{n+1} = \frac{-c_0 \Delta t}{\Delta x} (P_{i+1}^n - P_i^n) + P_i^n \quad (18)$$

$$P_i^{n+1} = \frac{c_0 \Delta t}{\Delta x} (P_i^n) + P_i^n - \frac{c_0 \Delta t}{\Delta x} (P_{i+1}^n) \quad (19)$$

Utilizando la ecuación 13 y reescribiendo el factor $\frac{c_0 \Delta t}{\Delta x}$ como el número de Courant, se reescribe la expresión de la forma:

$$P_i^{n+1} = -Cr(P_{i+1}^n - P_i^n) + P_i^n \quad (20)$$

Aplicándole las condiciones de estabilidad de Von Neumann, reemplazando el término V_i^n por $g^n e^{ij\theta}$ y arreglando matemáticamente la expresión se tiene:

$$|g|^2 = 1 + Cr^2 \operatorname{sen}^2 \theta \quad (21)$$

Dado que $|g|^2 \geq 1$ para cualquier número en los reales, el esquema es incondicionalmente inestable.

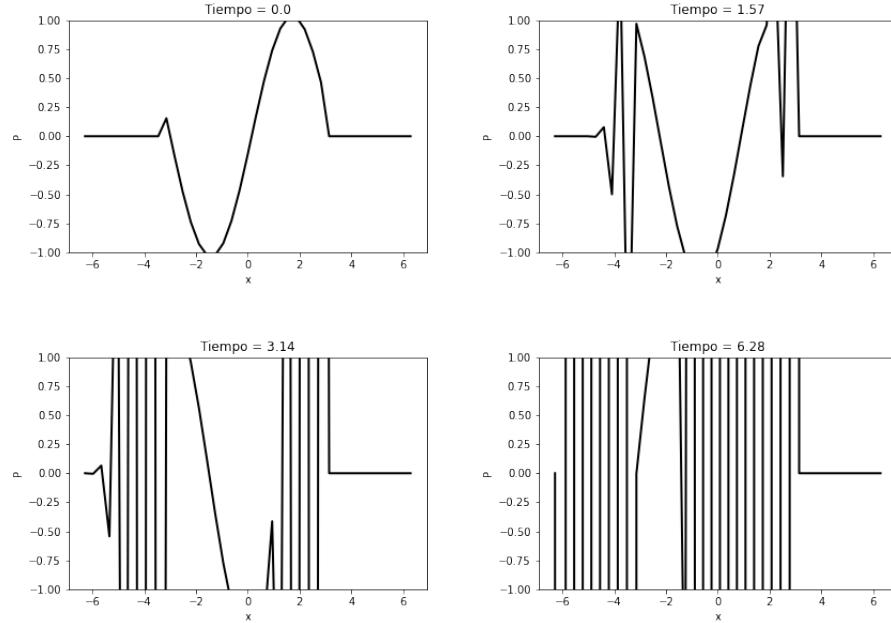


Figura 5: Modelación forward $Cr = 0.5$.

2.3.4 Preguntas asociadas

1. ¿Cuál es el más preciso? ¿Cuál es el más difusivo ? ¿Cuál utilizar y en qué caso?

Dados los esquemas presentados anteriormente, se puede decir que en base los criterios de estabilidad teóricos, el único estable es esquema backward en el tiempo, siempre y cuando se cumpla la condición de Lax - Friedrichs. Los otros dos esquemas, centered y forward, son incondicionalmente inestables para cualquier velocidad mayor a 0. Es por ello, que se recomienda utilizar un esquema atrasado, ya que este va avanzando en la misma dirección que la información, lo que entrega una mejor precisión y por lo tanto, confiabilidad en los resultados.

2.4 Lax - Friedrichs

El esquema de Lax - Friedrichs utiliza un nodo en la posición anterior y otro en la posición posterior en el espacio. Respecto al tiempo, se utiliza el promedio de la solución en dos pasos diferentes para aproximar la solución, lo cual busca evitar oscilaciones indeseadas. En base a ello, su discretización es la siguiente:

$$\frac{P_i^{n+1} - \frac{1}{2}(P_{i+1}^n - P_{i-1}^n)}{\Delta t} = -c_0 \frac{P_{i+1}^n - P_{i-1}^n}{2\Delta x} \quad (22)$$

$$P_i^{n+1} = \frac{-c_0 \Delta t}{2\Delta x} (P_{i+1}^n - P_{i-1}^n) + \frac{1}{2} (P_{i+1}^n - P_{i-1}^n) \quad (23)$$

$$P_i^{n+1} = \left(\frac{-c_0 \Delta t}{2\Delta x} + \frac{1}{2} \right) P_{i+1}^n + \left(\frac{c_0 \Delta t}{2\Delta x} - \frac{1}{2} \right) (P_{i-1}^n) \quad (24)$$

Utilizando la discretización presentada, la modelación computacional entrega los siguientes resultados:

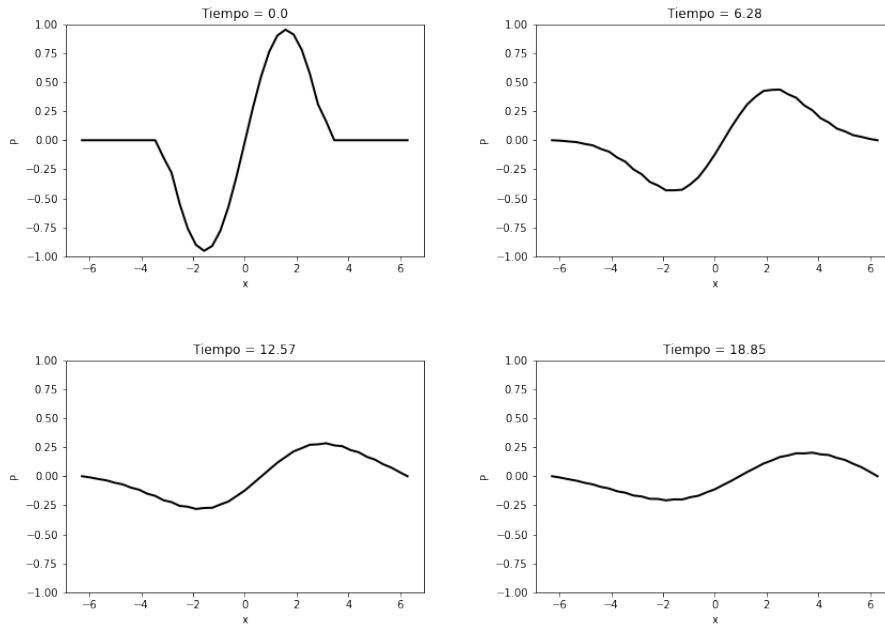


Figura 6: Modelación Lax - Friedrichs Cr = 0.5.

2.4.1 Preguntas asociadas

1. ¿Cuál es su ventaja con relación a los esquemas precedentes? ¿Es disipativo? ¿Cuándo?

A diferencia del esquema con diferencias adelantadas y atrasadas en el espacio y Euler explícito en el tiempo, el esquema de Lax - Friedrichs es estable. Para lograr esta estabilidad, la ecuación agrega un término de oscilación artificial, el cual provoca la pérdida de información que se evidencia en las simulación presentada.

2.5 Lax - Wendroff

Similar al esquema de Lax - Friedrichs, el método de Lax - Wendroff utiliza una discretización de segundo orden en el espacio, y el nodo actual y pasado en el tiempo para calcular el posterior. Así, la discretización resulta:

$$\frac{P_i^{n+1} - P_i^n}{\Delta t} + c_0 \frac{P_{i+1}^n - P_{i-1}^n}{2\Delta x} = c_0^2 \frac{\Delta t}{2} \frac{P_{i+1}^n - 2P_i^n + P_{i-1}^n}{2\Delta x^2} \quad (25)$$

$$P_i^{n+1} = -\frac{c_0 \Delta t}{2\Delta x} (P_{i+1}^n - P_{i-1}^n) + \frac{c_0^2 \Delta t^2}{4\Delta x} (P_{i+1}^n - 2P_i^n + P_{i-1}^n) + P_i^n \quad (26)$$

$$P_i^{n+1} = -\frac{c_0 \Delta t}{2\Delta x} (P_{i+1}^n) + \frac{c_0 \Delta t}{2\Delta x} (P_{i-1}^n) + \frac{c_0^2 \Delta t^2}{4\Delta x} (P_{i+1}^n) - \frac{c_0^2 \Delta t^2}{4\Delta x} (2P_i^n) + \frac{c_0^2 \Delta t^2}{4\Delta x} (P_{i-1}^n) + P_i^n \quad (27)$$

$$P_i^{n+1} = \left(-\frac{c_0 \Delta t}{2\Delta x} + \frac{c_0^2 \Delta t^2}{2\Delta x} \right) (P_{i+1}^n) + \frac{c_0 \Delta t}{2\Delta x} (P_{i-1}^n) - \left(\frac{c_0^2 \Delta t^2}{2\Delta x} - 1 \right) (P_i^n) \quad (28)$$

Utilizando la discretización presentada la modelación computacional entrega los siguientes resultados:

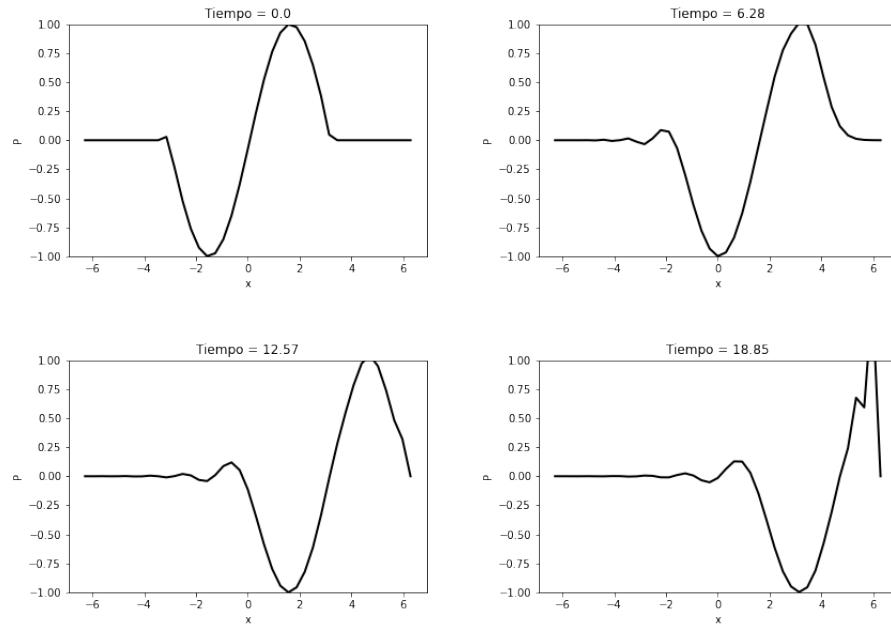


Figura 7: Modelación Lax - Wendroff C = 0.5.

2.5.1 Preguntas asociadas

1. ¿Cuál es su ventaja con relación a los esquemas precedentes?

Tal como se puede ver en la simulación presentada, el esquema de Lax - Wendroff es estable, lo cual lo pone en ventaja ante los esquemas de diferencias centradas y adelantadas. A pesar de ello, su solución no es precisa, ya que sufre de una gran dispersión, la cual comienza en el tiempo inicial y se amplifica en pasos posteriores.

2.6 Leap - Frog

Leap - Frog es un esquema que utiliza diferencias centradas en el espacio y el nodo anterior y actual para calcular el nodo posterior en el tiempo. Así,

$$\frac{P_i^{n+1} - P_i^{n-1}}{2\Delta t} = -c_0 \frac{P_{i+1}^n - P_{i-1}^n}{2\Delta x} \quad (29)$$

$$P_i^{n+1} = \frac{c_0 \Delta t}{2\Delta x} (P_{i+1}^n - P_{i-1}^n) + P_i^{n-1} \quad (30)$$

$$P_i^{n+1} = P_i^{n-1} + \frac{c_0 \Delta t}{2\Delta x} (P_{i+1}^n) - \frac{c_0 \Delta t}{2\Delta x} (P_{i-1}^n) \quad (31)$$

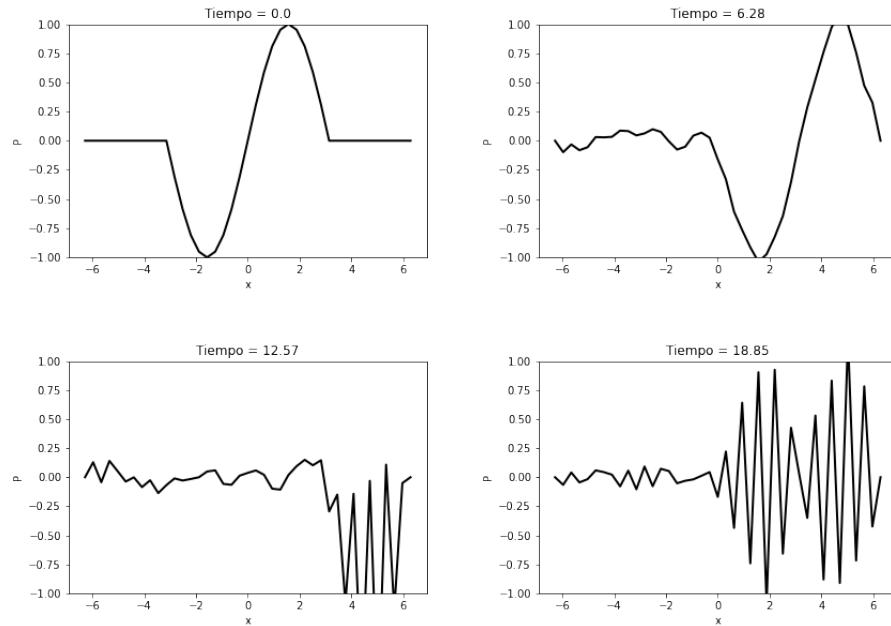


Figura 8: Modelación Leap - Frog $C = 0.5$.

2.6.1 Preguntas asociadas

1. ¿Es dispersivo? ¿Es disipativo? ¿Cuál es su desventaja?

El esquema de Leap - Frog utiliza diferencias finitas de segundo orden en el tiempo para aproximar las derivadas temporales. Estas diferencias provocan oscilaciones numéricas que se pueden

evidenciar en la modelación presentada, la cual sufre en alto grado de dispersión y en segundo lugar, de disipación. Su desventaja, por lo tanto, es que no transmite la información "enviada" como pulso en el tiempo inicial, sino que esta es deformada en gran dimensión.

2.7 Centrado en el espacio y Euler implícito en el tiempo

Utilizando la correspondiente ecuación:

$$\frac{P_i^{n+1} - P_i^n}{\Delta t} = \frac{-c_0}{2\Delta x} (P_{i+1}^{n+1} - P_{i-1}^{n+1}) \quad (32)$$

$$P_i^{n+1} = \frac{-c_0 \Delta t}{2\Delta x} (P_{i+1}^{n+1} - P_{i-1}^{n+1}) + P_i^n \quad (33)$$

$$P_i^{n+1} = \frac{c_0 \Delta t}{2\Delta x} (P_{i+1}^{n-1}) + P_i^n - \frac{c_0 \Delta t}{2\Delta x} (P_{i+1}^{n+1}) \quad (34)$$

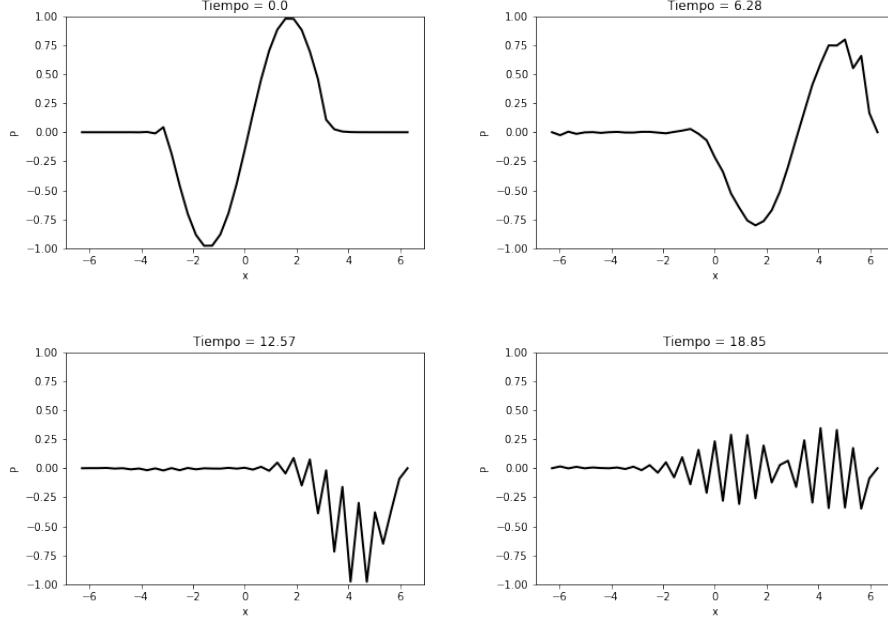


Figura 9: Modelación Centrada en el espacio y EI en el tiempo.

2.8 Centrado en el espacio y Crank - Nicolson en el tiempo

Utilizando la correspondiente ecuación:

$$\frac{P_i^{n+1}}{\Delta t} - \frac{P_i^n}{\Delta t} = \frac{c_0}{2} \frac{(P_{i+1}^{n+1} - P_{i-1}^{n+1})}{2\Delta x} + \frac{c_0}{2} \frac{(P_{i+1}^n - P_{i-1}^n)}{2\Delta x} \quad (35)$$

$$P_i^{n+1} = \frac{c_0 \Delta t}{2} \frac{(P_{i+1}^{n+1} - P_{i-1}^{n+1})}{2\Delta x} + \frac{c_0 \Delta t}{2} \frac{(P_{i+1}^n - P_{i-1}^n)}{2\Delta x} + P_i^n \quad (36)$$

$$P_i^{n+1} = \frac{c_0 \Delta t}{4\Delta x} (P_{i+1}^{n+1}) - \frac{c_0 \Delta t}{4\Delta x} (P_{i-1}^{n+1}) + \frac{c_0 \Delta t}{4\Delta x} (P_{i+1}^n) - \frac{c_0 \Delta t}{4\Delta x} (P_{i-1}^n) + P_i^n \quad (37)$$

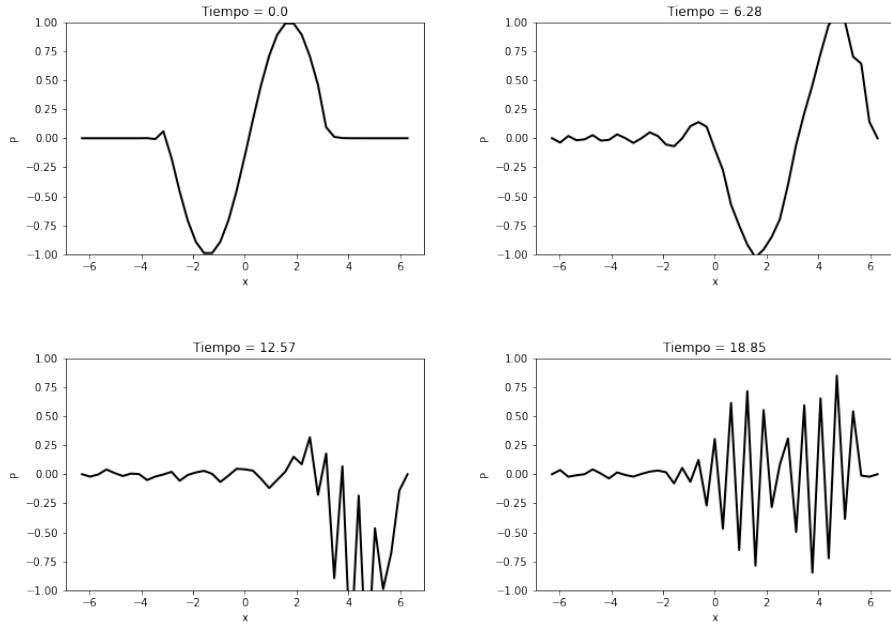


Figura 10: Modelación Centrada en el espacio y CN en el tiempo $Cr = 0.5$.

2.9 Análisis

Los esquemas presentados buscan resolver la misma ecuación, razón por la cual se pueden comparar entre sí. De este modo, se presentan dos diferentes grupos:

2.9.1 Errores métodos explícitos

Respecto a los métodos explícitos, se puede visualizar que la respuesta va de la mano con la intuición: A menos Cr , menor es el error asociado a la discretización. Esto tiene que ver con que el número de Courant disminuye, entregando una mayor estabilidad y precisión, dado que la separación entre cada fotograma es menor (menores pasos de tiempo, espacio o velocidades).

Se puede notar que los métodos menos precisos son aquellos que requieren información nodos espaciales posteriores para su cálculo. Esto tiene que ver con el sentido en el que viaja la información.

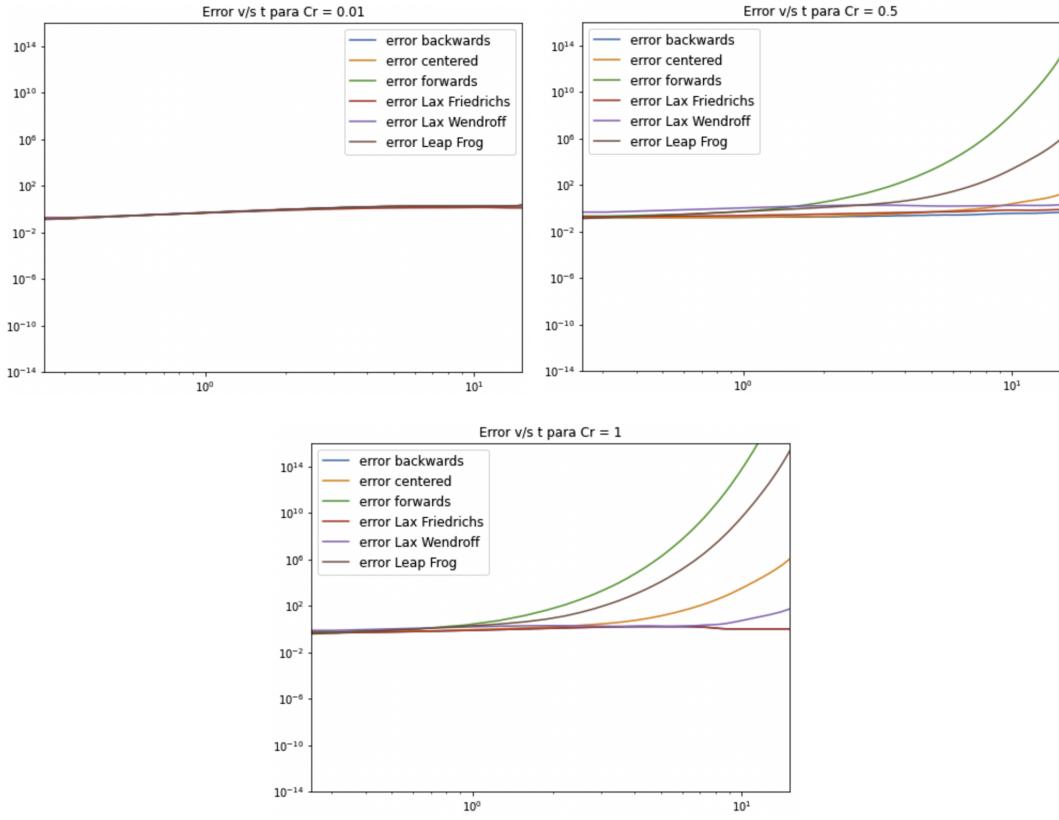


Figura 11: Errores métodos explícitos para distintos Cr.

2.9.2 Errores métodos implícitos

Al igual que los métodos explícitos, los esquemas implícitos presentan un menor error a menores números de Courant. Estos esquemas son más precisos en tiempos menores, ya que su construcción requiere de información de los nodos temporales anteriores y posteriores para los cálculos, lo que brinda una mayor estabilidad local.

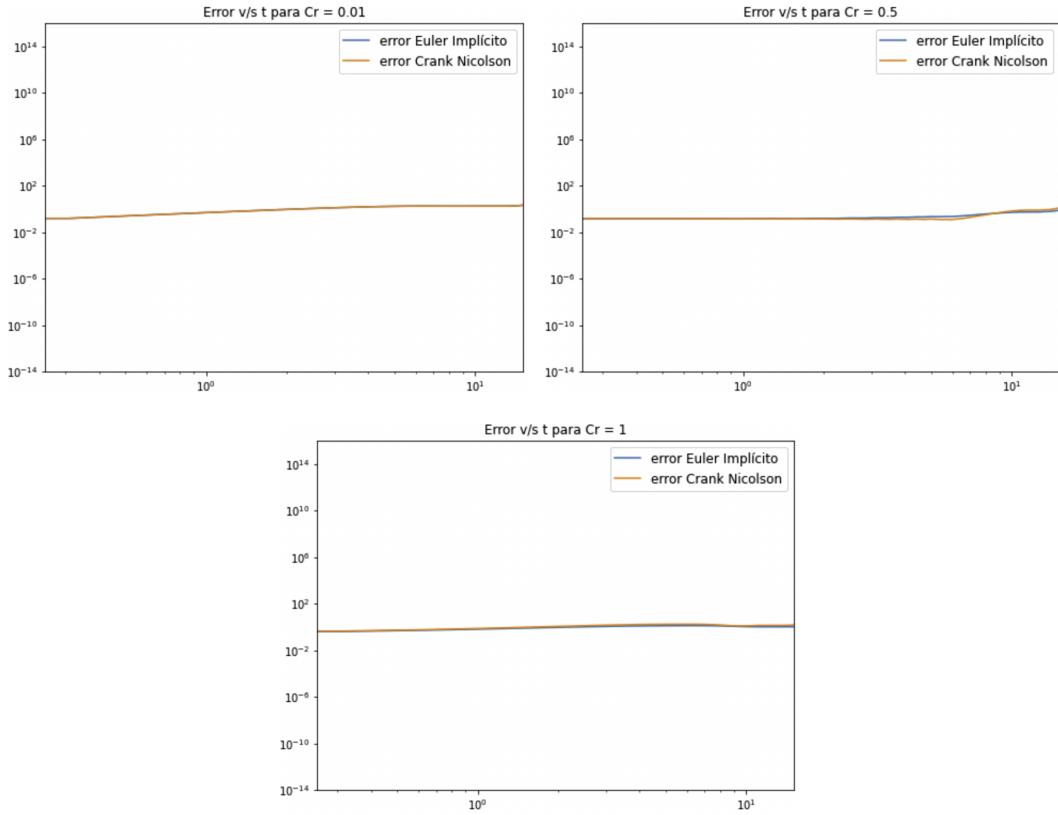


Figura 12: Errores métodos implícitos para distintos Cr.

Tabla 1: Tabla comparativa de errores.

Método	I.temporal	Precisión	Estabilidad	Propenso a
D.atrasadas	Explícito	$\sim 10^{-1}$	Condicionalmente estable	Disipación
D.centradadas	Explícito	$\sim 10^{-1} - 10^1$	Incondicionalmente inestable	Dispersión
D.adelantadas	Explícito	$\sim 10^{-1} - 10^{12}$	Incondicionalmente inestable	Dispersión
Lax Friedrichs	Explícito	$\sim 10^{-1}$	Condicionalmente inestable	Disipación
Lax Wendroff	Explícito	$\sim 10^0$	Condicionalmente inestable	Dispersión
Leap Frog	Explícito	$\sim 10^{-1} - 10^5$	Condicionalmente inestable	Dispersión
D.centradadas	Implícito	$\sim 10^{-1} - 10^0$	Incondicionalmente inestable	Dispersión y disipación
Crank Nicolson	Implícito	$\sim 10^{-1} - 10^0$	Incondicionalmente inestable	Dispersión y disipación

3 Ecuación de onda escalar 1D

Sea la ecuación de onda escalar de orden dos en una dimensión:

$$\frac{\partial^2 P}{\partial t^2} - c^2 \frac{\partial^2 P}{\partial x^2} = 0 \quad (38)$$

Utilizando la discretización para la segunda derivada se tiene:

$$\frac{P_i^{n+1} - 2P_i^n + P_i^{n-1}}{\Delta t^2} = c^2 \frac{P_{i+1}^n - 2P_i^n + P_{i-1}^n}{\Delta x^2} \quad (39)$$

$$P_i^{n+1} = \frac{c^2 \Delta t^2}{\Delta x^2} (P_{i+1}^n - 2P_i^n + P_{i-1}^n) + 2P_i^n - P_i^{n-1} \quad (40)$$

$$P_i^{n+1} = 2(1 - \frac{c^2 \Delta t^2}{\Delta x^2})(P_i^n) + \frac{c^2 \Delta t^2}{\Delta x^2} (P_{i+1}^n + P_{i-1}^n) - P_i^{n-1} \quad (41)$$

Con ello, se procede a comparar el esquema para distintos CFL, con el fin de comprender sus propiedades físicas:

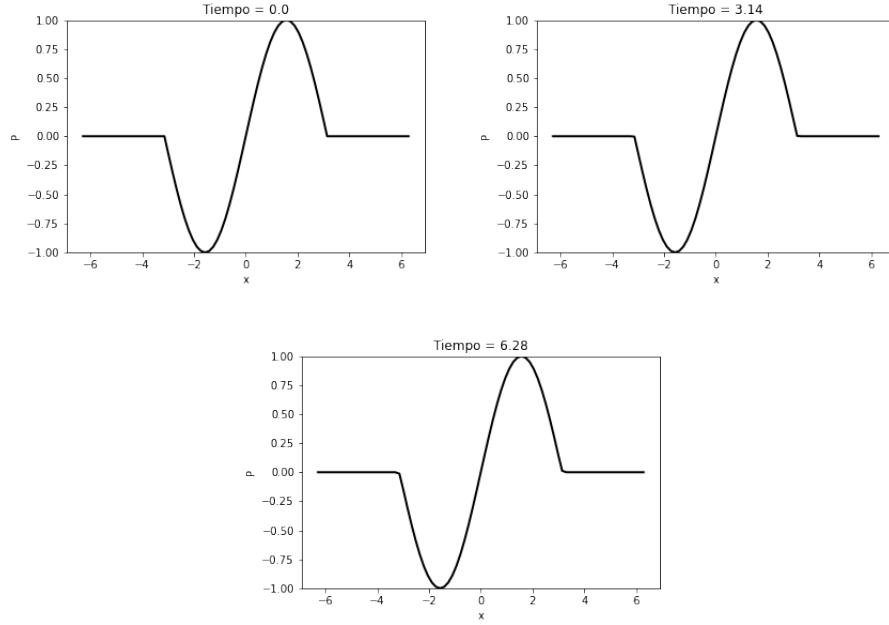


Figura 13: Esquema de Leap Frog de orden 2 para Cr = 0.01.

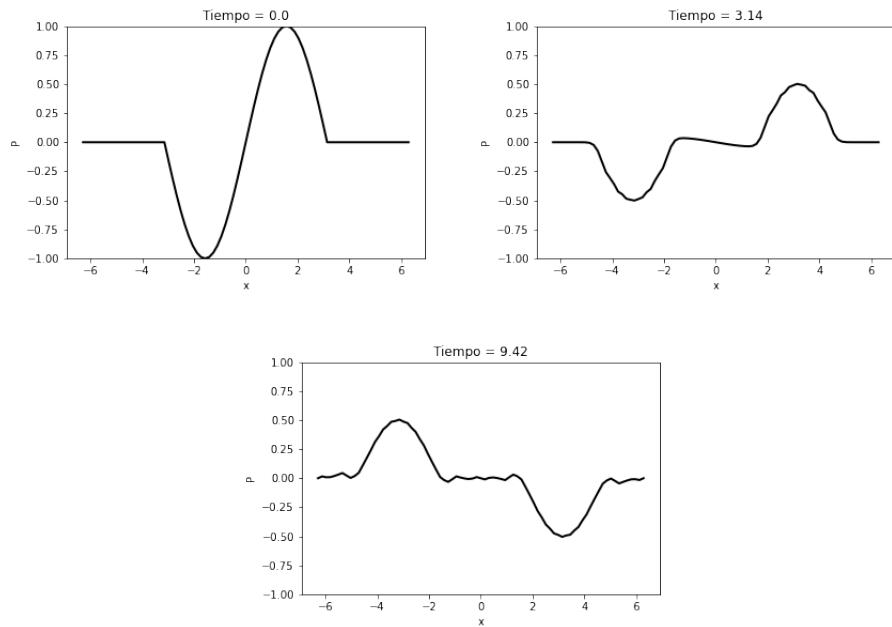


Figura 14: Esquema de Leap Frog de orden 2 para $Cr = 0.5$.

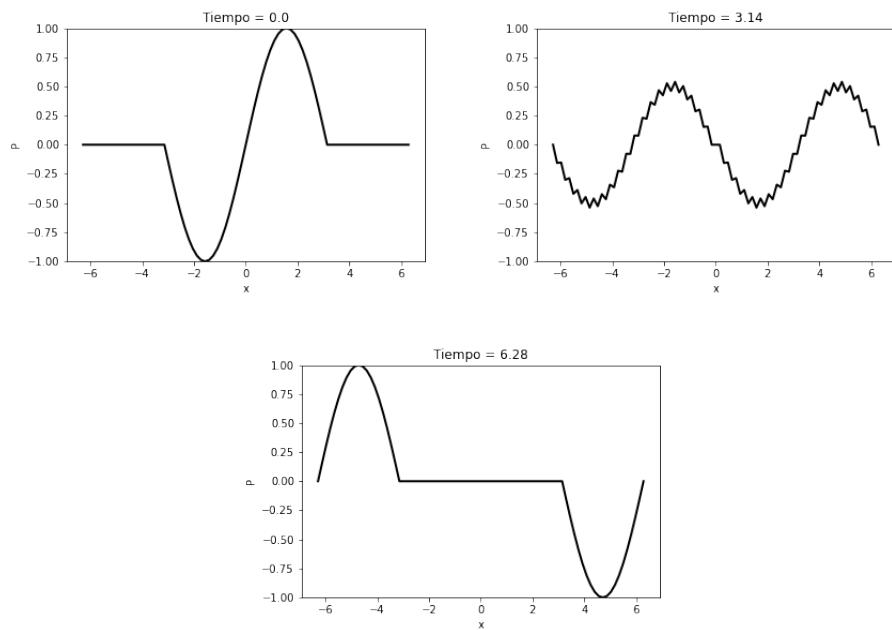


Figura 15: Esquema de Leap Frog de orden 2 para $Cr = 1$

3.0.1 Análisis

En base a las imágenes anteriormente presentadas, se puede notar que el esquema es condicionalmente inestable para distintos números de Courant, predominando esta en aquellos menores a 1, tal como lo indica la condición de estabilidad del número de Courant-Friedrichs-Lowy. Esta estabilidad se ve afectada por una considerable disipación y dispersión dada la forma del esquema, siendo esta última la cual se presenta en un tiempo que depende intrínsecamente del Cr.

El número de pasos de tiempo necesario depende de la velocidad de propagación en el medio, siendo suficiente 100 pasos cuando el número de Courant es 0.5. Respecto a la precisión, esta adquiere un orden dos dadas las propiedades del esquema.

Dado que el esquema de Leap Frog trabaja en el nodo posterior, anterior y actual respecto a el tiempo, este requiere de conocer $P[0]$ para inicializar. Para ello, se puede utilizar es euler en el tiempo con diferencias atrasadas en el espacio para el primer paso, pues este esquema mantiene el error asociado ya que la información viaja en el mismo sentido que la onda, además de ser generalmente estable. Es importante destacar que aplicar condición $P[0]=P[1]$ podría aumentar garrafalmente los errores asociados, ya que estos son de menor orden, y por consiguiente, afectan la precisión del método.

4 Ecuación de ondas escalar 1D de orden 4

4.1 Discretización

Sea el esquema:

$$\frac{\partial^2 P}{\partial x^2} = \frac{1}{\Delta x^2} [(AP_i^n + a(P_{i+1}^n + P_{i-1}^n) + b(P_{i+2}^n + P_{i-2}^n))] \quad (42)$$

Considerando que tenemos el perfil inicial P_i , realizamos la correspondiente expansión de Taylor en los nodos i-2, i-1, i+1 y i+2:

$$P_{i-2} = P_i - 2\Delta x P'_i + \frac{4\Delta x^2}{2} P''_i - \frac{8\Delta x^3}{6} P'''_i + \frac{16\Delta x^4}{24} P''''_i - O(h^5) \quad (43)$$

$$P_{i+1} = P_i + \Delta x P'_i + \frac{\Delta x^2}{2} P''_i + \frac{\Delta x^3}{6} P'''_i + \frac{\Delta x^4}{24} P''''_i + O(h^5) \quad (44)$$

$$P_{i-1} = P_i - \Delta x P'_i + \frac{\Delta x^2}{2} P''_i - \frac{\Delta x^3}{6} P'''_i + \frac{\Delta x^4}{24} P''''_i + O(h^5) \quad (45)$$

$$P_{i+2} = P_i + 2\Delta x P'_i + \frac{4\Delta x^2}{2} P''_i + \frac{8\Delta x^3}{6} P'''_i + \frac{16\Delta x^4}{24} P''''_i + O(h^5) \quad (46)$$

Generando el correspondiente sistema de ecuaciones se tiene:

$$\left. \begin{aligned} P_{i+1} + P_{i-1} &= 2P_i + \Delta x^2 P''_i + \frac{\Delta x^4}{12} P'''_i + O(h^5) \\ P_{i+2} + P_{i-2} &= 2P_i + 4\Delta x^2 P''_i + \frac{4\Delta x^4}{3} P'''_i + O(h^5) \end{aligned} \right\} \quad (47)$$

Con el fin de eliminar la cuarta derivada, se procede a restar estratégicamente ambas ecuaciones:

$$(P_{i+2} + P_{i-2}) - 16(P_{i+1} + P_{i-1}) = 2P_i + 4\Delta x^2 P''_i + \frac{4}{3}\Delta x^4 P'''_i \quad (48)$$

Así, luego de desarrollar y factorizar, se obtiene:

$$P''_i = \frac{1}{\Delta x^2} \left(\frac{-5}{2} P_i + \frac{4}{3} (P_{i+1} + P_{i-1}) - \frac{1}{12} (P_{i+2} + P_{i-2}) \right) \quad (49)$$

$$\boxed{A = \frac{5}{2}; a = \frac{-4}{3}; b = \frac{-1}{12}}$$

Con lo cual se logra obtener la expresión asociada a la segunda derivada. Realizando el mismo proceso para la cuarta derivada:

$$(P_{i+2} + P_{i-2}) - 4(P_{i+1} + P_{i-1}) = -6P_i + \Delta x^4 P'''_i \quad (50)$$

Así, luego de desarrollar y factorizar, se obtiene:

$$P''_i = \frac{1}{\Delta x^4} (6P_i - 4(P_{i+1} + P_{i-1}) + (P_{i+2} + P_{i-2})) \quad (51)$$

Por lo tanto:

$$\boxed{A = 6; a = -4; b = 1}$$

Así, el esquema global es:

$$\begin{aligned} \frac{P_i^{n+1} - 2P_i^n + P_i^{n-1}}{\Delta t^2} - \frac{c^4 \Delta t^2}{12 \Delta x^4} (6P_i^n - 4(P_{i-1}^n + P_{i+1}^n) + (P_{i-2}^n + P_{i+2}^n)) \\ = \frac{c^2}{\Delta x^2} \left(\frac{-5}{2} P_i^n + \frac{4}{3} (P_{i-1}^n + P_{i+1}^n) - \frac{1}{12} (P_{i-2}^n + P_{i+2}^n) \right) \end{aligned} \quad (52)$$

Reescribiendo:

$$\begin{aligned} P_i^{n+1} &= \frac{c^4 \Delta t^4}{12 \Delta x^4} (6P_i^n - 4(P_{i-1}^n + P_{i+1}^n) + (P_{i-2}^n + P_{i+2}^n)) \\ &\quad + \frac{c^2 \Delta t^2}{\Delta x^2} \left(\frac{-5}{2} P_i^n + \frac{4}{3} (P_{i-1}^n + P_{i+1}^n) - \frac{1}{12} (P_{i-2}^n + P_{i+2}^n) \right) + 2P_i^n - P_i^{n-1} \end{aligned} \quad (53)$$

Con ello, se pueden comparar diferentes CFL computacionalmente:

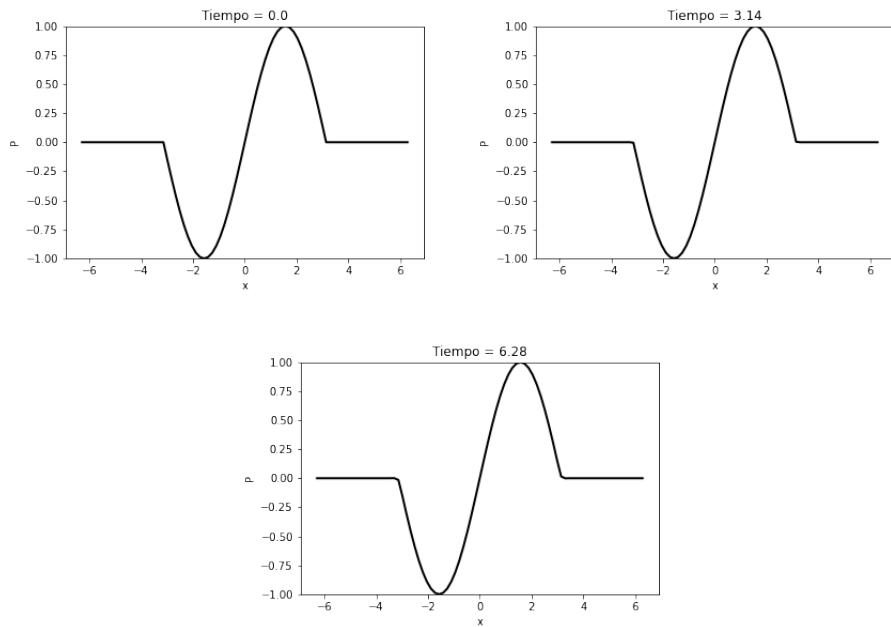


Figura 16: Ecuación de onda escalar con $Cr = 0.01$.

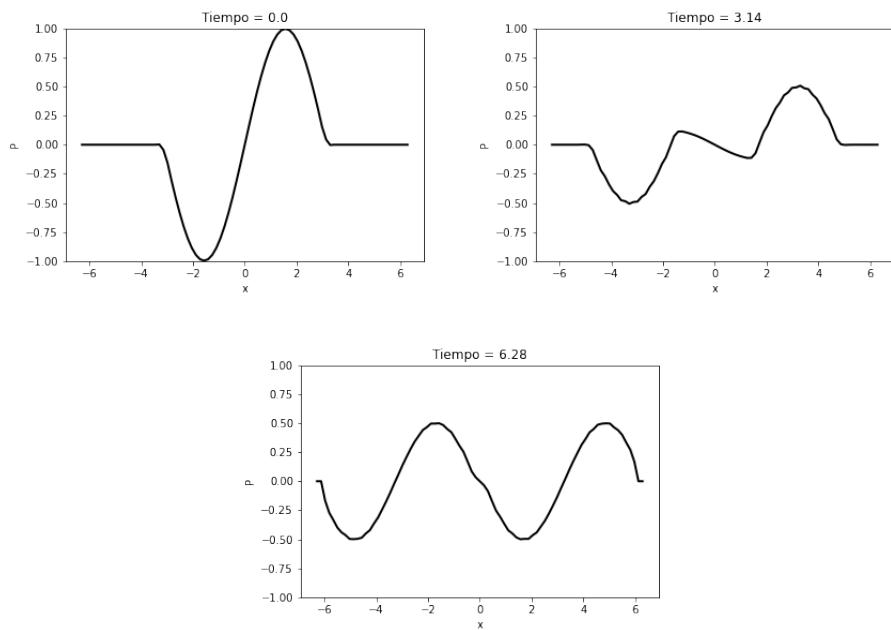


Figura 17: Ecuación de onda escalar con $Cr = 0.5$.

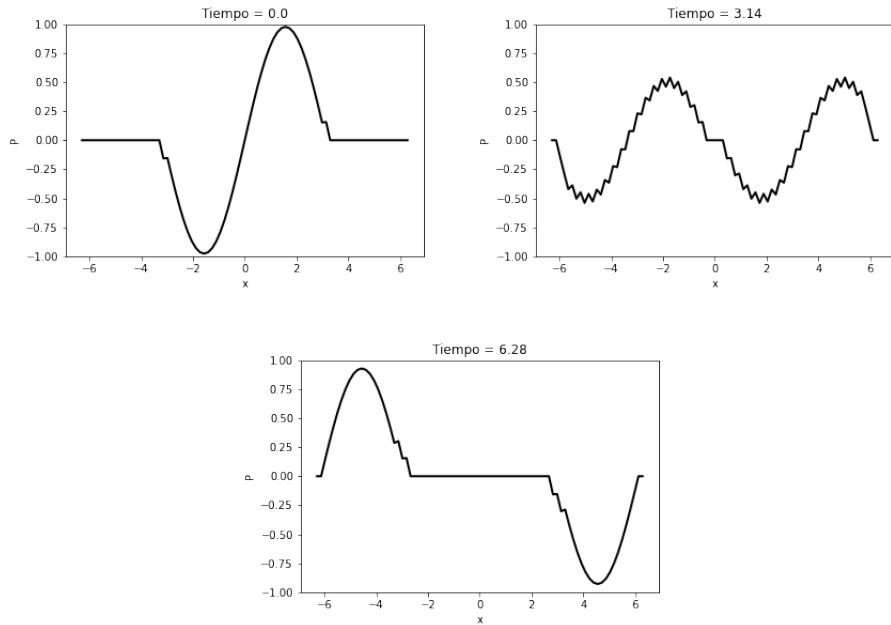


Figura 18: Ecuación de onda escalar con $\text{Cr} = 1$.

4.2 Análisis

En las modelaciones anteriormente presentadas se puede visualizar que el esquema es estable para los diferentes CFL. A pesar de esto, a medida que el Número de Courant aumenta el esquema sufre un mayor efecto de disipación, lo que quiere decir que la onda está perdiendo información. Esto se puede ver en la altura de las crestas y valles.

4.3 Comparación entre métodos

Es posible comparar las soluciones de onda escalar de ambos métodos: Leap Frog y de orden 4. Dado que mayor orden indica una mayor precisión, el método de orden 4 debiese entregar soluciones más verídicas, y por consiguiente, con menos error que el método de orden 2.

Las soluciones se pueden comparar en las siguientes imágenes:

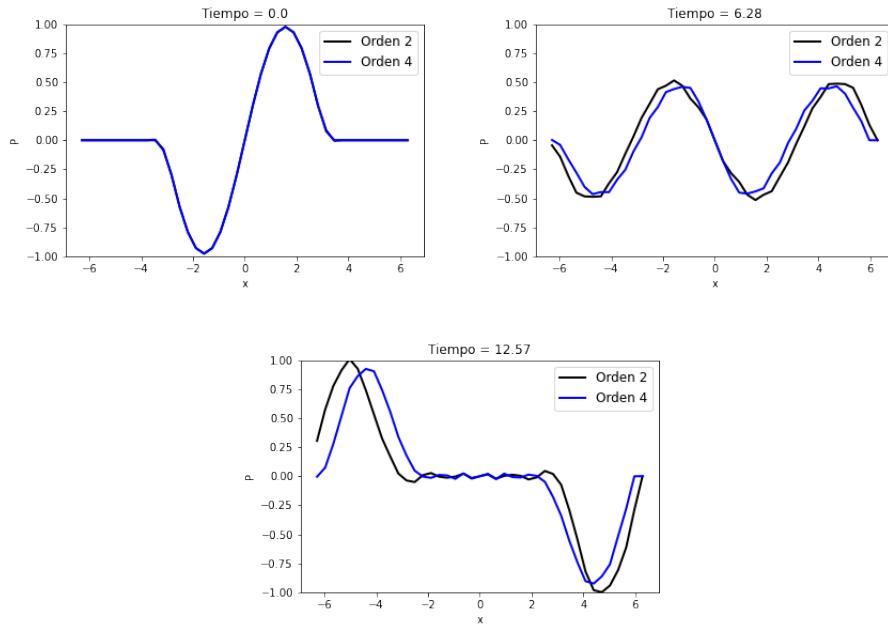


Figura 19: Ecuación de onda escalar con $\text{Cr} = 0.5$.

Referencias

[1] R.Luis, A.Zenner, V.José. "El método de diferencias finitas".

5 Anexos

5.1 Código computacional

Alumna: Javiera Espinoza Morales; ROL: 201941036-9

Importamos las librerías correspondientes:

```
In [2]: import numpy as np
from math import pi
from math import exp
from matplotlib import pyplot
import matplotlib.pyplot as plt
import imageio
```

Definimos la función para graficar:

```
In [3]: #Función para graficar
def graficar(x,P,t, images, dt):
    plt.plot(x, P, c='k', lw=2)
    plt.ylim([-1, 1])
    plt.title(f'Tiempo = {round(t*dt,2)}') # Título con el número de iteración
    plt.xlabel('x')
    plt.ylabel('P')

    # Guardar la figura actual como una imagen
    filename = f'Imagen_{t}.png'
    plt.savefig(filename)
    plt.close() # Cerrar la figura para no mostrarla en pantalla

    # Agregar la imagen a la lista
    images.append(imageio.imread(filename))
    return (images)
```

Cambiamos el directorio para poder guardar las imágenes posteriores:

```
In [4]: #cambio de directorio
import os

# Obtener el directorio de trabajo actual
directorio_actual = os.getcwd()
print("Directorio actual:", directorio_actual)

# Cambiar el directorio de trabajo actual a otro directorio
nuevo_directorio = '/Users/JavieraEspinosa/CFD'
os.chdir(nuevo_directorio)

# Verificar el cambio de directorio de trabajo
nuevo_directorio_actual = os.getcwd()
print("Nuevo directorio actual:", nuevo_directorio_actual)

Directorio actual: /Users/JavieraEspinosa
Nuevo directorio actual: /Users/JavieraEspinosa/CFD

In [5]: #Parámetros
dx = 0.1*pi
x = np.arange(-2*pi, 2*pi+dx, dx)
dt = 0.1*pi
Nt = 81
P = np.zeros(len(x))
c = 0.5
tp = []
for i in range(Nt):
    tp.append(round(i*dt,2))
```

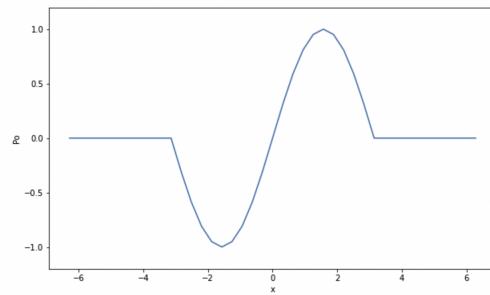
Pregunta N°1: Discretización de la ecuación de convección 1D

Paso inicial: determinar el perfil inicial

```
In [6]: #Sea el perfil inicial
def pinicial(x):
    return np.where((-np.pi <= x) & (x <= np.pi), np.sin(x), 0)

Pin = pinicial(x)
plt.figure(figsize = (10,6))
plt.plot(x,Pin)
plt.ylim(bottom = -1.2, top = 1.2)
plt.xlabel('x')
plt.ylabel('Po')

Text(0, 0.5, 'Po')
```



Solución analítica

```
In [7]: def PAu(x, dt, c,t):
    Pa = np.zeros(len(x))
    for i in range(len(x)-1):
        if x[i] >= (-np.pi + c*t*dt) and (np.pi + c*t*dt) >= x[i]:
            Pa[i] = np.sin(x[i] - c*t*dt)
    Pa[0] = Pa[-1] = 0
    return Pa
```

Diferencias en el tiempo: Backward, centradas, forward, Lax Friedrichs y Lax Wendroff

Discretización

Sea la ecuación de convección lineal en una dimensión:

$$\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = 0. \quad (1)$$

donde el subíndice i recorre la componente espacial de la malla (con espacio Δx) y el superíndice n la componente temporal (con pasos de tiempo Δt). En base a ello, procedemos a escribir las correspondientes funciones:

```
In [8]: def atrasadas(P,x, Nt, Cr):
    Pn = P.copy()
    for i in range(len(x)-1):
        Pi[i] = Cr*(Pn[i]-Pn[i-1]) + Pn[i]
    return P

In [9]: def centradas(P,x, Nt, Cr):
    Pn = P.copy()
    for i in range(len(x)-1):
        Pi[i] = (Cr/2)*(Pn[i+1]-Pn[i-1]) + Pn[i]
    return P

In [10]: def adelantadas(P,x, Nt, Cr):
    Pn = P.copy()
    for i in range(len(x)-1):
        Pi[i] = Cr*(Pn[i+1]-Pn[i]) + Pn[i]
    return P
```

¿Cuál es el más preciso? ¿Cuál es el más difusivo? ¿Cuál utilizar y en qué caso?

R// El método más preciso es el de diferencias atrasadas, ya que la información va avanzando en el mismo sentido que la onda. Tanto el método centrado como el adelantado son inestables, ya que no cumplen la condición del CFL. Entre ambos, el más difusivo es el de diferencias centradas.

```
In [11]: def lax_friedrichs(P,x,Nt,Cr):
    Pn = P.copy()
    for i in range(1, len(x)-1):
        Pi = ((Cr/2)*(Pn[i+1]-Pn[i-1]) + 0.5*(Pn[i+1] + Pn[i-1]))
    return P
```

¿Cuál es su ventaja con relación a los esquemas precedentes? ¿Es disipativo? ¿Cuándo?

R// Una de sus ventajas es que el método entrega una solución más suave que el resto, es decir, se ven menos esos trozos de línea "pegados", teniendo una solución más homogénea. Lamentablemente se puede ver que desde las primeras iteraciones existe disipación, por lo que a medida que la curva avanza en el tiempo la información se va perdiendo con rapidez.

```
In [12]: def lax_wendroff(P,x,Nt,Cr):
    Pn = P.copy()
    for i in range(1, len(x)-1):
        Pi = Pn[i] - (Cr/2)*(Pn[i+1] - Pn[i-1]) + ((Cr/2)**2) * (Pn[i+1] - 2*Pn[i] + Pn[i-1])
    return P
```

¿Cuál es su ventaja con relación a los esquemas precedentes?

Este esquema no pierde tanta información como el esquema de Lax - Friedrich, y representa mejor el fenómeno que los métodos de diferencias finitas, pero sufre dispersión dado que su orden es 2, lo cual podemos ver en el inicio de la onda y en su nodo máximo superior.

```
In [13]: def leap_frog(P,x,Nt,Cr,t):
    Pn1 = P.copy()
    Pn = P.copy()
    if t == 1:
        for i in range(1, len(x)-1):
            Pi1 = Pn1[i] + Cr * (Pn1[i+1] - Pn1[i-1])
        Pn1 = Pn.copy()
        Pn = P.copy()
    if t > 1:
        for i in range(1, len(x)-1):
            Pi1 = Pn1[i] + Cr * (Pn1[i+1] - Pn1[i-1])
        Pn1 = Pn.copy()
        Pn = P.copy()
    return P
```

Comparación de errores

```
In [14]: def L2_error(P, Pa):
    e = np.sqrt(np.sum((P-Pa)**2)/np.sum(Pa**2))
    return e

#Guarda las P de las distintas funciones

Cr = [-0.01, -0.5, -1]
for elementos in Cr:
    err = []
    for _ in range(7):
        P = [Pn.copy() for _ in range(7)]
        images = [im for _ in range(7)]

        for t in range(Nt):
            Backwards = atrasadas(P[1],x,Nt,elementos)
            Centered = centradas(P[2],x,Nt,elementos)
            Forwards = adelantadas(P[3],x,Nt,elementos)
            Lax_Friedrichs = lax_friedrichs(P[4],x,Nt,elementos)
            Lax_Wendroff = lax_wendroff(P[5],x,Nt,elementos)
            Leap_Frog = leap_frog(P[6],x,Nt,elementos,t)

            im1 = graficar(x,Backwards,t, images[1], dt)
            im2 = graficar(x,Centered,t, images[2], dt)
            im3 = graficar(x,Forwards,t, images[3], dt)
            im4 = graficar(x,Lax_Friedrichs,t, images[4], dt)
            im5 = graficar(x,Lax_Wendroff,t, images[5], dt)
            im6 = graficar(x,Leap_Frog,t, images[6], dt)

            Panalitica = Pn(x,dt,c,t)
            error1 = L2_error(Backwards,Panalitica)
            err[1].append(error1)
            error2 = L2_error(Centered,Panalitica)
            err[2].append(error2)
            error3 = L2_error(Forwards,Panalitica)
            err[3].append(error3)
            error4 = L2_error(Lax_Friedrichs,Panalitica)
            err[4].append(error4)
```

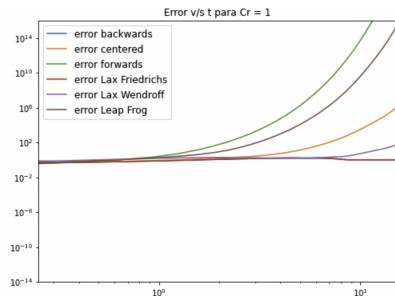
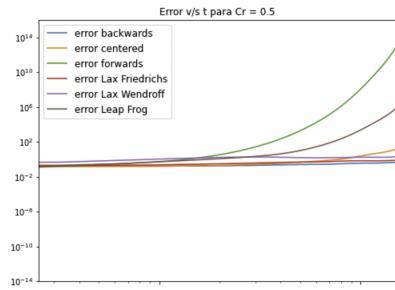
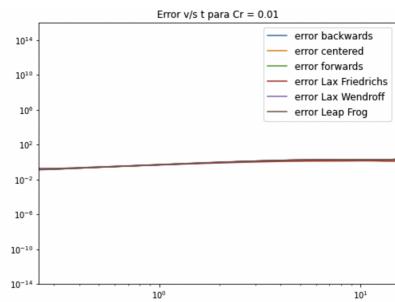
```
error5 = L2_error(Lax_Wendroff,Panalitica)
err[5].append(error5)
error6 = L2_error(Leap_Frog,Panalitica)
err[6].append(error6)

imageio.mimwrite('Backwards_C=' + str(elementos) + '.gif', images[1], duration=0.01)
imageio.mimwrite('Centered_C=' + str(elementos) + '.gif', images[2], duration=0.01)
imageio.mimwrite('Forwards_C=' + str(elementos) + '.gif', images[3], duration=0.01)
imageio.mimwrite('LaxFriedrichs_C=' + str(elementos) + '.gif', images[4], duration=0.01)
imageio.mimwrite('LaxWendroff_C=' + str(elementos) + '.gif', images[5], duration=0.01)
imageio.mimwrite('LeapFrog_C=' + str(elementos) + '.gif', images[6], duration=0.01)

fig = plt.figure(figsize=(8,6))
plt.title('Error v/s t para Cr = ' + str(abs(elementos)))

plt.loglog(tp,err[1], label='error backwards')
plt.loglog(tp,err[2], label='error centered')
plt.loglog(tp,err[3], label='error forwards')
plt.loglog(tp,err[4], label='error Lax Friedrichs')
plt.loglog(tp,err[5], label='error Lax Wendroff')
plt.loglog(tp,err[6], label='error Leap Frog')

plt.legend(loc='best', prop={'size':12})
plt.xlim([0,15]);
plt.ylim([10e-15,10e15]);
plt.show()
```



Métodos implícitos

Sean los métodos implícitos:

```
>>> from scipy.linalg import solve
#Genera la matriz de coeficientes
def generarMatriz(nx, alfa):
    matriz = np.zeros((nx-2,nx-2))
    for i in range (nx-2):
        for j in range (nx-2):
            if i == j and j!=(nx-3):
                matriz[i,j] = 2
            if i == 0:
                matriz[i+1,j] = -alfa
            else:
                matriz[i-1,j] = alfa
            else:
                if i == j:
                    matriz[i,j] = 2
                    matriz[i-1,j] = alfa
```

```

        matriz[i-1,j] = alfa
    return(matriz)

#Genera la matriz P
def lado_derechoe(P):
    d = np.zeros(len(P)-2)
    for n in range (len(P)-3):
        d[n] = 2*P[n+1]
    return(d)

def generarMatriz2(nx, a, b):
    matriz = np.zeros((nx-2,nx-2))
    for i in range (nx-2):
        for j in range (nx-2):
            if i == j and j !=(nx-3):
                matriz[i,j] = a
            if i == 0:
                matriz[i,0] = b
            else:
                matriz[i-1,j] = -b
            if i == j:
                matriz[i,j] = a
                matriz[i-1,j] = -b
    return(matriz)

#Genera la matriz P
def lado_derecho(P):
    d = np.zeros(len(P)-2)
    for n in range (len(P)-3):
        d[n] = P[n+1]
    return(d)

]: def euler_implicito(P, A, Nt):
    b = lado_derecho(P)
    P_interior = solve(A, b)
    P[1:-1] = P_interior
    P[0] = 0 # Dirichlet BC
    P[-1] = 0 # Dirichlet BC
    return P

def crank_nicolson(P, A, B, Nt):
    b = lado_derecho(P)
    B1 = np.dot(B,b)
    P_interior = solve(A, B1)
    P[1:-1] = P_interior
    P[0] = 0 # Dirichlet BC
    P[-1] = 0 # Dirichlet BC
    return P

Cr = [0.01, 0.5, 1]
for elementos in Cr:
    A = generarMatriz((len(x)),elementos)
    A1 = generarMatriz2((len(x)),4, (-1*elementos))
    B = generarMatriz2((len(x)), 4, (elementos))
    err7 = []
    err8 = []
    P7 = Pn.copy()
    P8 = Pn.copy()
    images7 = []
    images8 = []

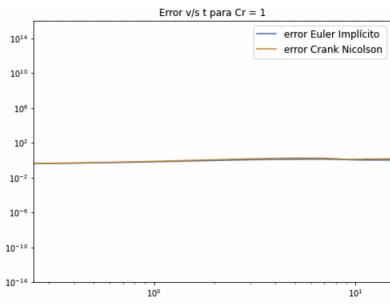
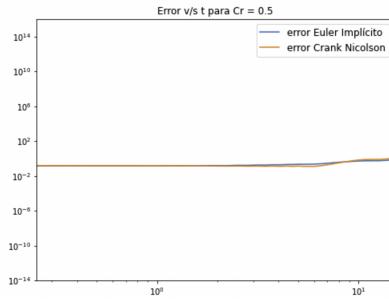
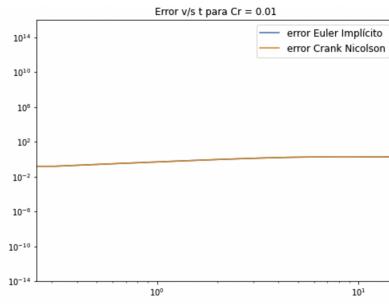
    for t in range (Nt):
        EI = euler_implicito(P7,A,Nt)
        CN = crank_nicolson(P8, A1, B, Nt)
        im7 = graficar(x,EI,t, images7, dt)
        im8 = graficar(x, CN, t, images8, dt)
        Panalitica = PAn(x,dt,c,t)
        error7 = L2_error(EI,Panalitica)
        error8 = L2_error(CN,Panalitica)
        err7.append(error7)
        err8.append(error8)

    imageio.mimwrite('Euler implicito ' + str(elementos) +'.gif', images7, duration=0.01)
    imageio.mimwrite('Crank Nicolsone ' + str(elementos) +'.gif', images8, duration=0.01)

fig = plt.figure(figsize=(8,6))
plt.title('Error v/s t para Cr = '+str(abs(elementos)))

plt.loglog(t,err7, label='error Euler Implicito')
plt.loglog(t,err8, label='error Crank Nicolson')
plt.legend(loc='best',prop={'size':12})
plt.xlim([0,15])
plt.ylim([10e-15,10e15]);
plt.show()

```



Pregunta N°2: Discretización de la ecuación de onda escalar 1D

```
#Parámetros
dx = 0.1*pi
x = np.arange(-2*pi, 2*pi*dx, dx)
dt = 0.1*pi
Nt = 81
P = np.zeros(len(x))
c = 0.5
tp = []
for i in range(Nt):
    tp.append(round(i*dt,2))

LL = []
```

```

: def leap_frog2(P,x, Nt, c, dt, dx):
    L = []
    Pn = P.copy()
    Pn1 = Pn.copy()
    images = []
    Cr = 0.5
    #paso en el tiempo
    for t in range(Nt):
        #paso en el espacio
        if t == 0:
            for i in range (len(x)-1):
                P[i] = 2*(1-(Cr**2))*Pn[i] + (Cr**2)*(Pn[i+1]+Pn[i-1]) - Pn1[i]
                Pn1 = Pn.copy()
                Pn = P.copy()
        if t > 1:
            for i in range(len(x)-1):
                P[i] = 2*(1-(Cr**2))*Pn[i] + (Cr**2)*(Pn[i+1]+Pn[i-1]) - Pn1[i]
                Pn1 = Pn.copy()
                Pn = P.copy()
        L.append(P.copy())
        P[0] = P[-1] = 0
        Panalitica = PAN(x,dt,c,t)
        im = graficar(x,P,t, images, dt)
        # Guardar las imágenes como un archivo GIF
        imageio.mimwrite('animacion.gif', images, duration=0.1)
    return P, L

P9 = Pn.copy()
P9, L = leap_frog2(P9,x, Nt,c,dt,dx)

```

Pregunta N°3: Ecuación de ondas escalar 1D de orden 4

$$P_i^{n+1} = \frac{c^4 \Delta t^4}{12 \Delta x^4} (6P_i^n - 4(P_{i-1}^n + P_{i+1}^n) + (P_{i-2}^n + P_{i+2}^n)) + \frac{c^2 \Delta t^2}{\Delta x^2} \left(\frac{-5}{2} P_i^n + \frac{4}{3} (P_{i-1}^n + P_{i+1}^n) - \frac{1}{12} (P_{i-2}^n + P_{i+2}^n) \right) + 2P_i^n - P_i^{n-1}. \quad (2)$$

```

def sol(P,x, Nt, Cr, c):
    Pn1 = P.copy()
    Pn = P.copy()
    images10 = []
    #paso en el tiempo
    for t in range(Nt):
        #paso en el espacio
        for i in range (len(x)-2):
            if t==1:
                if i-2 == 0:
                    P[i-1] == 0
                if i-1 == 0:
                    P[i-2] == 0
            else:
                a = ((Cr**4)/12)*(6*Pn[i] - 4*(Pn[i-1] + Pn[i+1]) + (Pn[i-2] + Pn[i+2]))
                b = (Cr**2)*((-5/2)*Pn[i] + (4/3)*(Pn[i-1] + Pn[i+1]) - (1/12)*(Pn[i-2] + Pn[i+2])) + 2*Pn[i] - Pn1[i]
                P[i] = a + b
        L.append(P.copy())
        P[0] = P[1] = 0
        Pn1 = Pn.copy()
        Pn = P.copy()
    # Guardar las imágenes como un archivo GIF
    return P, L

Cr = 0.5
P10 = Pn.copy()
P10 = sol(P10,x, Nt, Cr, c)

Imag = []
for t in range(Nt):
    #Función para graficar
    plt.plot(x, L[t], c='k', lw=2, label= 'Orden 2')
    plt.plot(x, LL[t], c='b', lw=2, label= 'Orden 4')
    plt.ylim([-1, 1])
    plt.title(f'Tiempo = {round(t*dt,2)}') # Título con el número de iteración
    plt.legend(loc='best', prop={'size':12})
    plt.xlabel('x')
    plt.ylabel('P')

    # Guardar la figura actual como una imagen
    filename = f'Imagen_{t}.png'
    plt.savefig(filename)
    plt.close() # Cerrar la figura para no mostrarla en pantalla

    # Agregar la imagen a la lista
    Imag.append(imageio.imread(filename))

```