93.75 - Métodos Numéricos Avanzados

Instituto Tecnológico de Buenos Aires

Trabajo Práctico Nro. 2

Spectral Splitting Methods

Grupo: MNAmigos

Alumnos: Ignacio Vidaurreta - 57250, Clara Guzzetti - 57100, Juan Bensadon - 57193, Gastón Lifschitz - 58225, Jimena Lozano - 58095, Marina Fuster - 57613

Profesores: Adrián Omar Álvarez, Pablo Esteban Schmidberg

Fecha: 22 de noviembre, 2020

Índice

1. Introducción

El presente informe desarrolla los detalles de la implementación y resultados de un sistema implementado para resolver el problema de la ecuación de *Korteweg-de Vries*[1] utilizando un método de *splitting* afín paralelo[2].

La motivación de este trabajo es la de realizar una implementación concurrente que logre resolver la ecuación diferencial de Korteweg-de Vries (a continuación referida como KdV la cual estimamos que, debido a la naturaleza concurrente de nuestra solución, resultará mucho maś eficiente que las implementaciones actualmente conocidas).

Como explica Xiang Tian[3] la ecuación de KdV describe cómo evolucionan las ondas frente a efectos de no-linearidades débiles y dispersiones también débilees. Podemos verlo en fenómenos físicos como las ondas navegantes entre otros.

En el contexto de nuestra materia, nos parece sumamente importante modelar dicha ecuación ya que es el principio de transmisión de datos que tienen las fibras ópticas.

2. Implementación

2.1. Características principales

- Programación realizada en MATLAB.
- Post procesamiento de datos y gráficos realizados en Python.

2.2. Metodología

La implementación de métodos afines consiste en los resultados obtenidos con los integradores de Lie Trotter de diferentes pasos. Esta técnica permite hacer los cálculos matemáticos en paralelo, reduciendo así los tiempos de procesamiento, que se aprovecha más aún sobre implementaciones modulares simples y escalables. Los métodos splitting aprovechan la capacidad de resolver fácilmente este tipo de problemas parciales, encontrando soluciones aproximadas al problema aplicando alternativamente los flujos parciales asociados a cada subproblema.

Para encontrar el escalar necesario para los integradores:

$$1/2 = \gamma_1 + \gamma_2 + ... + \gamma_n$$

$$0 = \gamma_1 + 2^{-2j}\gamma_2 + ... + n^{-2j}\gamma_n$$

Dado el flujo asociado a problemas parciales, definimos las aplicaciones para utilizar el método Lie Trotter:

$$\phi^+(h) = \phi_1(h) \cdot \phi_0(h), \phi(h) = \phi_0(h) \cdot \phi_1(h)$$

$$\phi_m^{\pm}(h) = \phi^{\pm}(h) \cdot \phi_{m-1}^{\pm}(h)$$

La implementación en pseudo código del método Lie Trotter es la siguiente:

```
for n=1:order
    if is_reverse
        U = nonlinear(U, k, delta_t/order);
        U = linear(U, k, delta_t/order);
    else
        U = linear(U, k, delta_t/order);
        U = nonlinear(U, k, delta_t/order);
    end
end
```

- Línea 2: Aplicamos el step reverso (no lineal o lineal)
- Línea 5: Aplicamos el step no reverso (lineal o no lineal)

Luego, aplicando el método afín simétrico, el integrador a utilizar es:

$$\phi(h) = \sum_{m=1}^{s} \gamma_m(\phi_m^+(h/m) + \phi_m^-(h/m))$$

Finalmente, la implementación en pseudo código del método afín para el caso paralelizable es la siguiente:

```
1 X = gammas(order);
2 S = zeros(size(U));
3 parfor n=1:2*order
4     if mod(n, 2) == 0
5         S = S + X(n/2)*LieTrotter(U, k, delta_t, n/2, true);
6     else
7         S = S + X((n+1)/2)*LieTrotter(U, k, delta_t, (n+1)/2, false);
8     end
9 end
```

- Línea 1: obtenemos los coeficientes gammas para los integradores.
- Línea 2: declaramos la variable donde se guardará el resultado.
- Línea 5: realizamos el Lie Trotter reverso.
- Línea 7: realizamos el Lie Trotter no reverso.

2.3. Exactitud

Para analizar la exactitud del método, utilizamos la medición del error descrita en uno de los papers provistos por la cátedra[6]. Esta medición es llevada a cabo por la ejecución del método en un paso temporal h y en un paso temporal h/2, para luego analizar la diferencia entre las aproximaciones a ambos pasos temporales. Esto nos permite obtener el error punto a punto, durante la duración de la aproximación. Por otro lado, es posible tomar norma infinita del vector de las diferencias para obtener un error del método. Nótese que la elección de norma infinita es nuestra, ya que uno podría haber utilizado, por ejemplo, la norma 2 para realizar esta medición.

3. Resultados

En base a las pruebas realizadas, se obtuvieron los siguientes resultados.

3.1. Aproximación

En esta sección buscamos mostrar brevemente la ejecución de nuestro programa. Inicializa con dos solitones y vemos luego que el más grande pasa por encima del más pequeño. En la Fig. 2, vemos el resultado final de realizar aproximaciones con un tiempo máximo de 1.5 segundos.

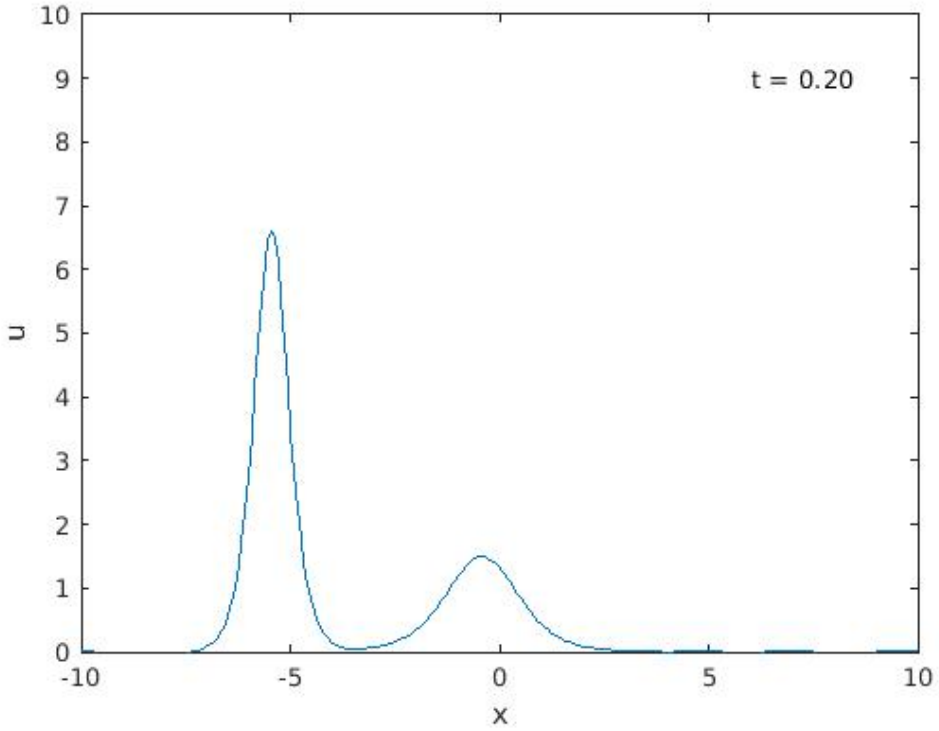

Figura 1: Aproximación en t=0.20

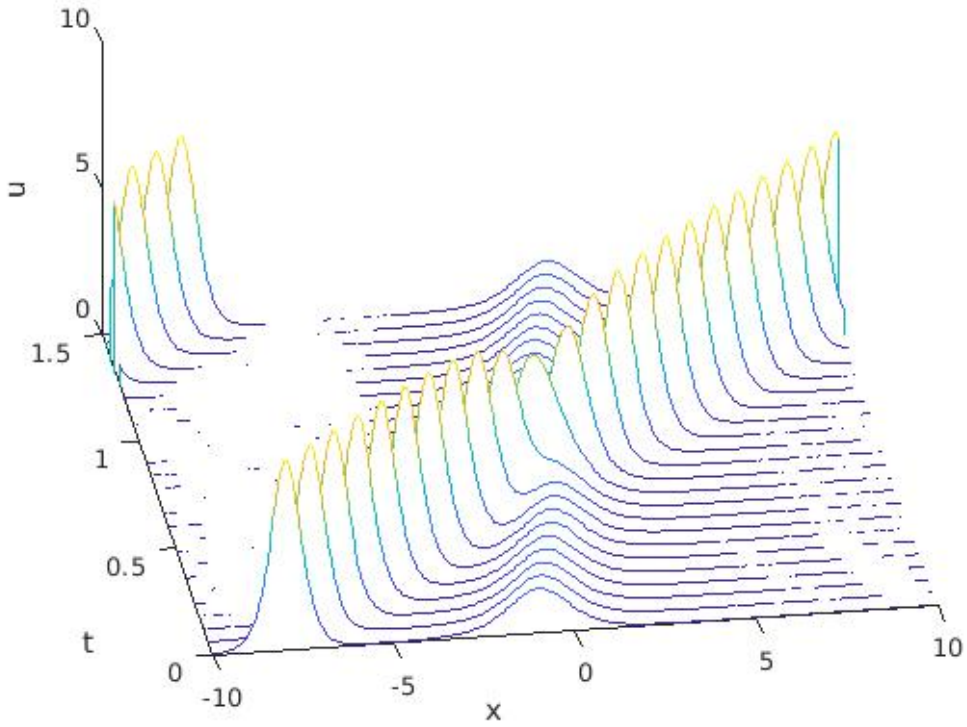

Figura 2: Aproximaciones realizadas entre 0 y 1.5 segundos

3.2. Métricas calcualdas

Para obtener la información necesaria para realizar los análisis pertinentes fue necesario recabar información sobre las ejecuciones. A continuación mostramos las métricsa obtenidas para determinadas corridas del programa. Además de estas métricas, puede obtenerse el error. Esto se explicará con mayor detalle en la sección de exactitud.

Orden	Delta$_T$	Tiempo [s] (Paralelo)	Tiempo [s] (Serie)
2	0,0005	61,07348	4,647198
2	0,0005	58,24946	4,672958
2	0,0005	57,29200	4,649283
2	0,0005	57,62675	4,754210
2	0,0005	59,24001	4,658284
4	0,0005	63,47466	6,760054
4	0,0005	63,98020	6,664634
4	0,0005	63,80706	6,659773
4	0,0005	63,29659	6,667074
4	0,0005	63,54526	6,765790
6	0,0005	74,86527	10,61734
6	0,0005	75,95905	10,45760
6	0,0005	75,19689	10,74926
6	0,0005	74,82743	10,06732
6	0,0005	74,93699	9,861968

Podemos observar que el tiempo de ejecución de nuestro método en paralelo es mucho mayor al tiempo en serie. Esto es muy interesante porque resulta casi anti intuitivo. Nosotros partimos de la premisa de que la paralelización va a mejorar la performance de nuestro algoritmo. Esta situación se explicará con mayor detalle en la sección de Speed Up.

3.3. Speed up

Se midió el Speed Up a partir de las ejecuciones realizadas en la sección 3.1 de Métricas Calculadas. Nótese que fueron realizadas 5 ejecuciones por valor de interés para lograr valores más reales al conocer la desviación estándar que el programa puede presentar. Se realizó para órdenes 2, 4 y 6, con un dt = 0.0005.

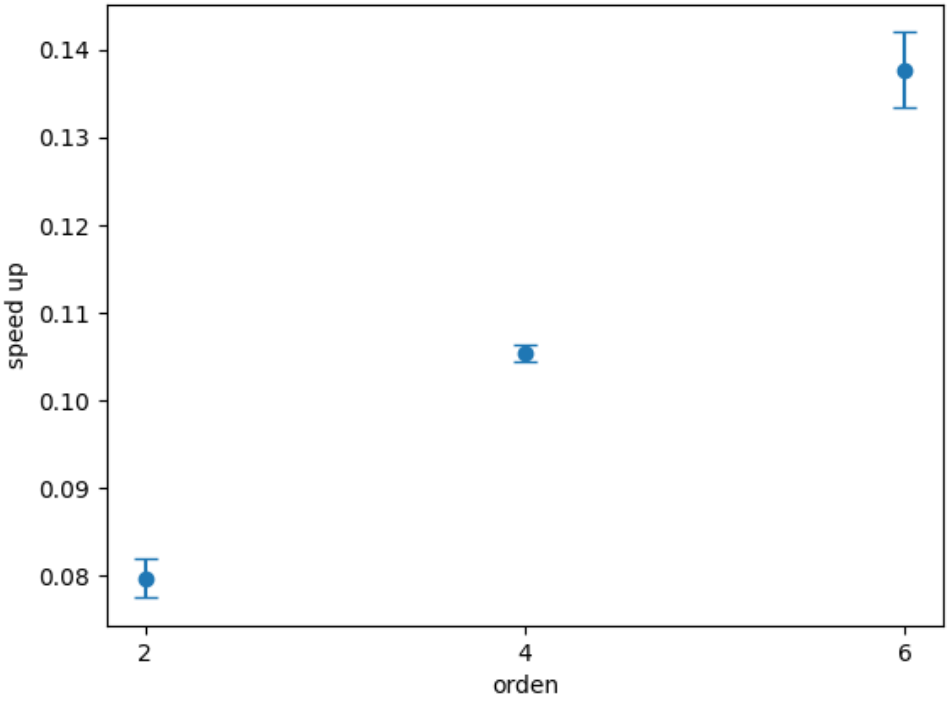

Figura 3: Speed up.

Esta figura es importante para notar varias cosas. En primer lugar vemos una tendencia creciente, lo cual es esperado: a mayor paralelización, mayor es el Speed Up.

Por otra parte, llaman la atención los valores menores a uno, pues uno esperaría que la ejecución en paralelo sea más veloz que la ejecución en serie, a pesar de que esto no sería una tendencia creciente infinita como se ha demostrado con la Ley de Amdahl, la cual nos indica que existe un techo para la velocidad que podemos ganar paralelizando. Lo que puede estar provocando que el Speed Up sea menor a uno es el contexto en el cual ejecutamos nuestro programa.

Una razón por la cual puede suceder esto es porque usamos órdenes bajos y delta ts inadecuados para mostrar los beneficios de la paralelización, ya que las máquinas de las cuales disponemos son de propósitos generales.

Otra razón por la que no está sucediendo esto es proque estamos trabajando con un hardware hogareño que no está especializado para el trabajo distribuido.

Entonces, existe un tiempo no trivial que se va acumulando de comunciación entre los distintos nodos de nuestro çluster.[el] cuál le agregará cierto *overhead* a nuestro código y es este el que hace que performe peor el paralelo para nuestro ejemplo. Sin embargo, creemos que en el caso de utilizar el hardware apropiado como por ejemplo el servicio de Amazon Web Services de clústers de alta performance[5] la mejora sería evidente.

3.4. Tiempo ideal vs. tiempo real

En esta sección llamamos tiempo real al promedio que el programa tarda en correr en paralelo. Por otro lado, el tiempo ideal (cuya definición fue dada en la teórica) refiere al tiempo que tarda el programa en correr en serie, sobre la cantidad de núcleos de los cuales se dispone. Se promediaron los tiempos de 5 corridas y se comparó el tiempo ideal con el tiempo real.

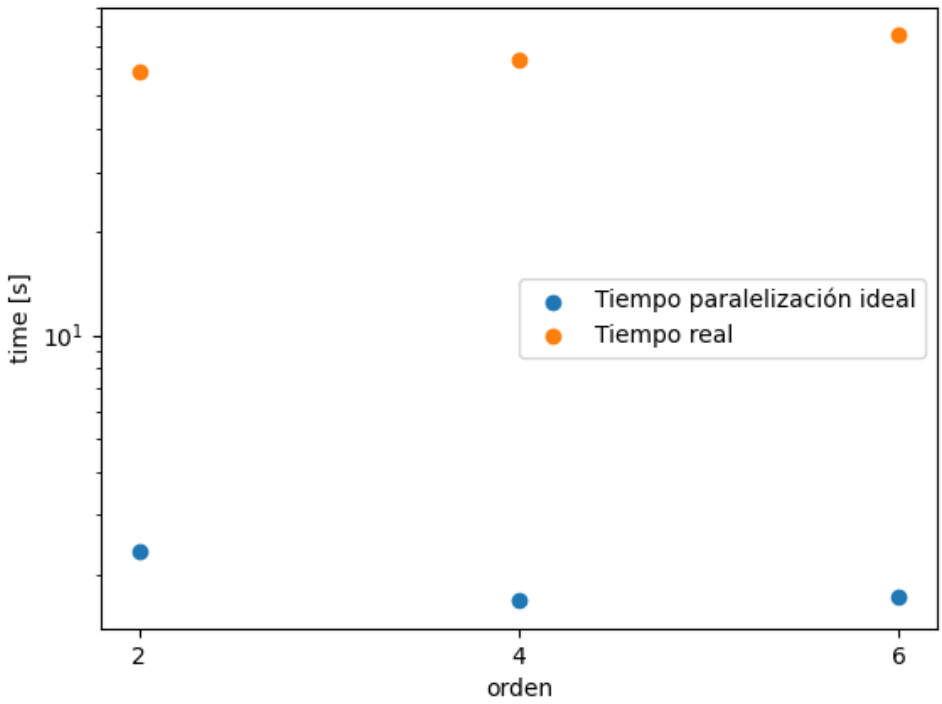

Figura 4: Tiempo real vs. tiempo ideal.

El tiempo real es mayor que el tiempo ideal por dos razones. En primer lugar, la paralelización no es gratuita. Hay cierto costo referente a la distribución y comunicación entre distitos flujos de ejecución que provocarán que no pueda alcanzarse el tiempo ideal. Por otro lado, como fue explicado en la sección anterior, a órdenes tan bajos se tarda más en serie que en paralelo, lo cual provoca que la brecha se acentúe aún más de lo que debería.

3.5. Exactitud

El análisis de resultados en lo referente a la exactitud es sumamente importante a la hora de construir un método. Es importante tener conocimiento de cuánto estamos perdiendo al realizar aproximaciones, en la medida que ello sea posible. Para la ecuación que se analiza en este informe, existe una solución analítica, pero la misma es extensa para ser eficiente en la comparación. Es por ello que, para conocer la exactitud, nos guiaremos por el error correspondiente a la fórmula presente en la sección 2.3. Se analizaron varios escenarios: se buscó comparar la variación en la exactitud a diferentes órdenes, a diferentes deltas de tiempo y con diferentes métodos. En esta sección nos dedicaremos a los primeros dos escenarios y luego será dedicada una sección a la comparación de dicho método con el método de Strang.

En primer lugar, analicemos qué ocurre con la exactitud en función del orden. Se realizaron ejecuciones con un $dt = 0.0001$, para órdenes 2, 4 y 6.

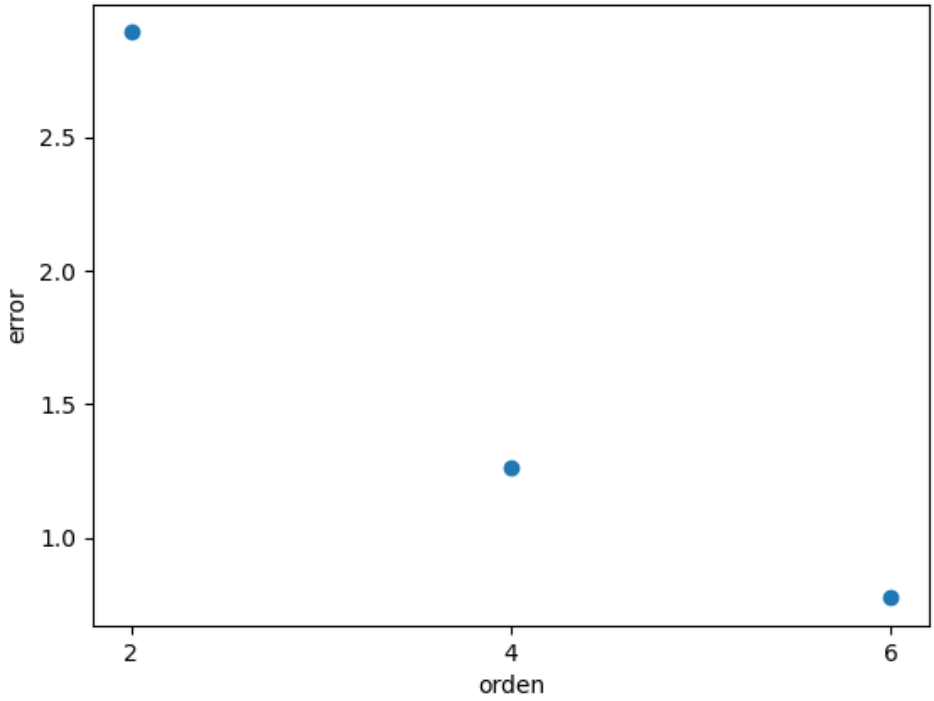

Figura 5: Error vs. Orden

Estos resultados nos muestran que, al menos para los órdenes con los cuales nos estuvimos manejando se muestra una tendencia decreciente del error al aumentar el orden. Para afirmar categóricamente que el error siempre disminuye al aumentar el orden, se debería realizar un análisis más extenso que encuentre la asíntota para la cual el orden ya no disminuye significativamente.

Por otro lado, nos pareció interesante analizar más al detalle, qué ocurre con la evolución de dicho error a medida que avanza el tiempo. Para representar dichas situaciones, se tomaron las mismas ejecuciones que se mencionaron para Fig. 5.

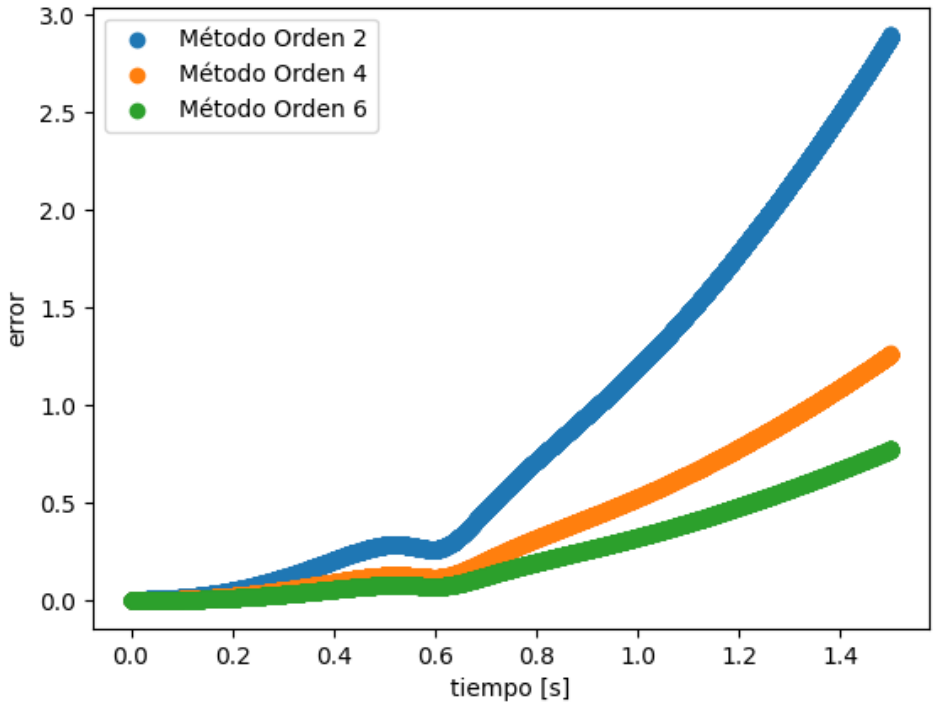

Figura 6: Error en función del tiempo, según orden

Aquí vemos que la tendencia pareciera indicar que el error aumenta a medida que pasa el tiempo. Esto es esperable ya que todos los métodos comienzan en la misma situación inicial, en cuyo momento el error es cero. Luego las aproximaciones van acumulando el error, lo que provoca que éste incremente cuanto mayor es el período de aproximación utilizado.

Como segundo escenario y como fue mencionado más arriba, se analizó la evolución del error en el tiempo para distintos pasos temporales.

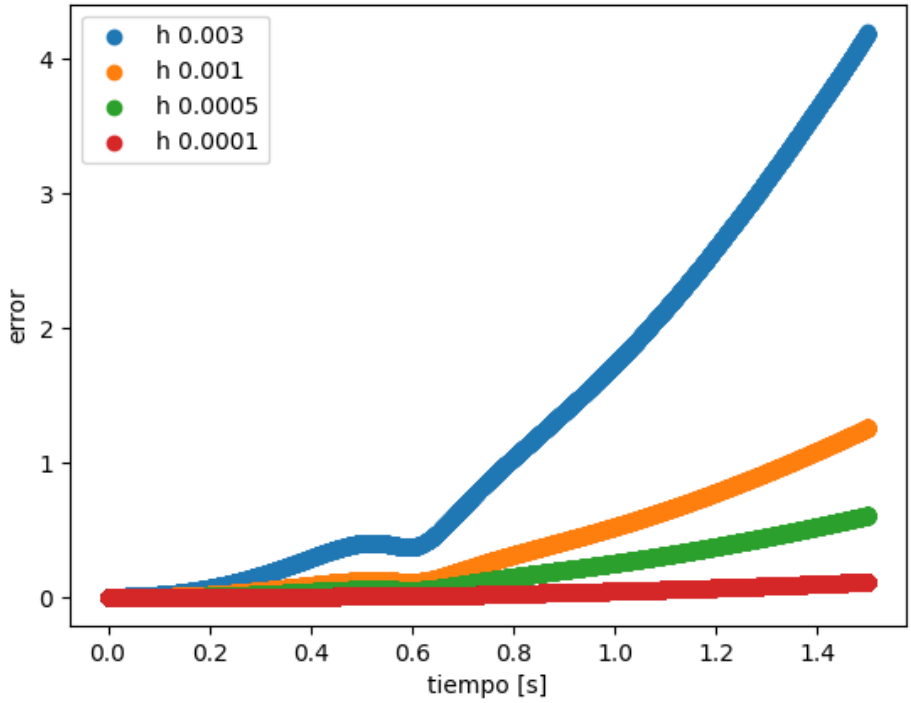

Figura 7: Error en función del tiempo, según el paso temporal

En Fig. 7. podemos apreciar que, a menor paso temporal, el error se vuelve menor.

3.6. Comparación con el método de Strang

Ya hablamos bastante sobre el método Afín como método en sí y obtuvimos distintas métricas que nos hacen ver la utilidad de este método.

Sin embargo, creemos que el análisis no estará completo hasta que lo comparemos con el actual estado del arte. Uno de los métodos más famosos de *splitting* es el método de Strang[4].

El método de Strang es un método de orden 2 el cuál tiene las ventajas de ser bastante rápido y en relación a la velocidad bastante preciso. Por otro lado, nos fue de especial interés realizar la comparación con este método ya que en este grupo nos resulta muy interesante el profesor Strang y sus videos del MIT Opencourseware de

Algebra Lineal.

Para comparar ambos métodos, lo primero que hacemos es graficar la exactitud del método a lo largo del tiempo. La forma en la que realizamos el calculo de dicha exactitud es el mismo que fue mencionado en la sección 3.2. En el Fig. 8 observamos que la exactitud del método de Strang pareciera ser menor en todo momento que la dada por el método método afín en orden 2.

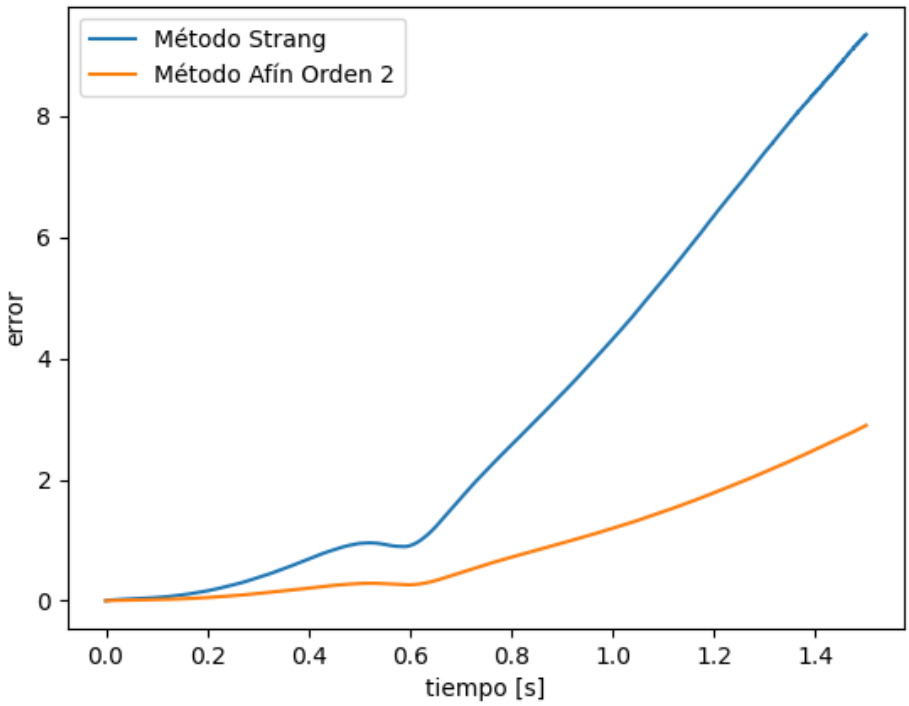

Figura 8: Strang vs Afin accuracy

Podemos verlo en forma de tabla con la norma infinito.

Método	Error
strang	9,349504
afín (Ord. 2)	2,893575

Otro punto de comparación que podemos hacer es sobre el tiempo de ejecución. Para esto en la siguiente tabla observamos el tiempo que tarda el método de Strang

en comparativa con nuestro método en orden 2 (lo hacemos con el cálculo en serie por los problemas que trae la paralelización sobre hardware hogareño)

Método	Tiempo de Ejecución
strang	2,7124 seg
afín (Ord. 2)	8,2685354 seg

Podemos observar entonces, que si bien el método de Strang resulta en una menor exactitud, el tiempo que tarda en obtener la aproximación es mucho menor.

4. Conclusiones

Para finalizar, queríamos comentar sobre ciertos puntos que nos parecieron de gran interés. En primer lugar, es necesario resaltar que el método es parametrizable y que permite seleccionar ciertos parámetros de manera tal que podamos mejorarlo en exactitud o velocidad. Por ejemplo, los resultados han mostrado que a menor paso temporal y a mayor orden, la exactitud del método mejora. Esto puede ser interesante para controlar la exactitud con un método adaptativo que se concentre en la precisión más que en la velocidad: puede analizar el error en todo momento y ajustar el paso temporal para mantenerlo por debajo de un epsilon deseado.

Por otra parte, sería de gran interés realizar este análisis en el contexto de procesamiento distribuido, con los parámetros de paso temporal y orden que se realizan en la industria, con cluster de computadoras. Esto nos daría mejores resultados sobre todas las métricas analizadas e, inclusive, nos permitiría validar la ley de Amdahl de tener el poder de cómputo necesario para ello.

En tercer lugar, es importante entender los límites presentes en la ciencia de la computación dados por leyes como la Ley de Amdahl. Esta ley es esencial a la hora de construir soluciones porque nos permite entender hasta dónde podemos llegar con la paralelización. Inclusive es importante para estar atentos y reconocer validez en las soluciones provistas por otros.

Otro punto descatado a mencionar, que se desprende de las aproximaciones que realizamos, es que nosotros tenemos dos solitones que avanzan en el tiempo, donde el de mayor altura pasa al de menor altura. Esto, en el contexto de aplicación de transmisión de datos, quiere decir que uno puede mandar una señal con una velocidad de datos y, a la vez, mandar otra señal con otra velocidad de datos en el mismo cable. Nos resultó muy interesante este resultado.

Por úlimo, el trabajo nos muestra la importancia de rodearse de contexto a la hora de presentar un método y compararlo con aquellos de su clase. Se resalta la necesidad de investigar y tener presente cuál es el estado del arte, de manera tal que uno pueda tomar las métricas necesarias que le permitan entender cómo se posiciona el propio método respecto de aquellos existentes.

Referencias

[1] https://en.wikipedia.org/wiki/Korteweg

[2] https://www.researchgate.net/publication/339014976_Paralleling_Affine_methods_with_spect

[3] Xiang, Tian. (2015). A Summary of the Korteweg-de Vries Equation. 10.13140/RG.2.1.1579.4083.

[4] https://pdfs.semanticscholar.org/29c1/80990c1d744170fd25011c95f5b48990bfca.pdf

[5] https://aws.amazon.com/hpc/

[6] https://people.maths.ox.ac.uk/trefethen/publication/PDF/2005_111.pdf