Métodos Numéricos Avanzados
Trabajo Práctico Nº2

"KDV"

21 de Noviembre de 2019

Comisión: S

Grupo 5

Ferrer Tomás	*57207*
Lo Coco Juan Pablo	*57313*
Lund Marcos	*57159*
Princ Guido	*57334*
Zuberbuhler Ximena	*57287*

Índice

Resumen 2

Palabras Clave 2

Introducción 2

Metodología 3

- **Modelo matemático** 3
- **Pseudocódigo KdV.** 5
- **Desarrollo** 5
- **Pruebas** 6

Resultados 7

Conclusiones 12

Bibliografía 13

Resumen

El objetivo de este trabajo es desarrollar un sistema informático capaz de obtener una solución aproximada a las ecuaciones Korteweg-de Vries a partir el uso de SSM (Spectral Splitting Methods) [1].

Palabras Clave

SSM: Spectral Splitting Methods
KdV: Korteweg-de Vries
RK4: Runge Kutta de Orden 4
FFT: Transformada Rápida de Fourier
iFFT: Anti-transformada Rápida de Fourier

Introducción

A mediados del siglo XIX, en las aguas poco profundas del canal que va de Falkirk a Edimburgo, el ingeniero escocés John Scott Russell observó una onda creada por un bote. Al detenerse el bote, la onda chocó con este, agitándose durante el choque pero recuperando su forma original tras sobrepasar, y transmitiendo como una elevación solitaria en el agua. Russell bautizo entre sus notas a este fenómenos Onda de Translación.

En 1895, el matemático Diederik Korteweg y su discípulo Gustav de Vries modelaron la elevación del agua con la ecuación en derivadas parciales a la cual bautizaron ecuación KdV. Sin embargo no fue hasta 1960 cuando se empezaron a emplear ordenadores para estudiar la propagación no lineal de ondas y se empezó a dar importancia al fenómeno.

De este estudio de ondas solitarias partieron numerosas aplicaciones a la ciencia moderna a través del estudio del comportamiento dinámico de los sistemas, en campos como la hidrodinámica, la óptica no lineal o la fisica de particulas elementales.

Metodología

Modelo matemático

Para la realización del proyecto se trabajó sobre la ecuación KdV detallada en la Figura 1, utilizando dos solitones u ondas con velocidades y amplitudes distintas.

$$u_t + 6uu_x + u_{xxx} = 0$$

Figura 1: Ecuación KdV.

Al utilizar SSM, se dividió el problema en una parte lineal y una parte no lineal, como puede observarse de las Figuras 2 y 3.

$$u = u \,.\, e^{(i \,.\, k^3 \,.\, h)}$$

Figura 2: Solución Lineal

$$f = -i \,.\, k^3 \,.\, h$$
$$a = f \,.\, FFT(Re(iFFT(u))^2)$$
$$b = f \,.\, FFT(Re(iFFT(u + \tfrac{a}{2}))^2)$$
$$c = f \,.\, FFT(Re(iFFT(u + \tfrac{b}{2}))^2)$$
$$d = f \,.\, FFT(Re(iFFT(u + c))^2)$$
$$u = u + \frac{(a + 2(b + c) + d)}{6}$$

Figura 3: Solución no lineal

Se optó por nombrar $\Phi_1(h)$ a la operación lineal, y $\Phi_0(h)$ a la operación no lineal, según la notación dada en [4]. Luego, para cada SSM implementado se utilizaron composiciones de dichas operaciones como se detalla en las Figuras 4 y 5.

$$\Phi_L^j = \Phi_{1-j}(h) \circ \Phi_j(h)$$

Figura 4: Implementación Lie-Trotter [2][4]

$$\Phi_S^j(h) = \Phi_j(h/2) \circ \Phi_{1-j}(h) \circ \Phi_j(h/2)$$

Figura 5: Implementación Strang [3][4]

Finalmente, se optó por implementar un integrador simétrico que itere por órdenes pares según lo establecido en las Figuras 6, 7, 8 y 9 utilizando valores de γ_m previamente establecidos [4]. En concreto, aquellos enunciados en la Figura 10.

$$\Phi(h) = \sum_{m=1}^{s} \gamma_m \Phi_m^{\pm}(h/m)$$

Figura 6: Integrador simétrico

$$\Phi_m^{\pm}(h) = \Phi^{\pm}(h) \circ \Phi_{m-1}^{\pm}(h)$$

Figura 7: Ecuación de $\Phi_m^{\pm}$

$$\Phi^{+}(h) = \Phi_1(h) \circ \Phi_0(h)$$

Figura 8: Ecuación de Φ^{+}

$$\Phi^{-}(h) = \Phi_0(h) \circ \Phi_1(h)$$

Figura 9: Ecuación de Φ^{-}

$$If\ q = 2,\ \gamma_1 = 1/2$$
$$If\ q = 4,\ \gamma_1 = -1/6,\ \gamma_2 = 2/3$$
$$If\ q = 6,\ \gamma_1 = 1/144,\ \gamma_2 = -8/63,\ \gamma_3 = \gamma_4 = 0,\ \gamma_5 = 625/1008$$
$$If\ q = 8,\ \gamma_1 = -\frac{1}{2304},\ \gamma_2 = \frac{32}{675},\ \gamma_3 = -\frac{729}{3200},\ \gamma_4 = \gamma_5 = \gamma_6 = 0,\ \gamma_7 = \frac{117649}{172800}$$

Figura 10: Valores de γ_m para ordenes q pares

Para la solución inicial del problema se utilizó la ecuación provista en la Figura 11, y se iteró a partir de los resultados obtenidos de la misma.

$$u(x) = \frac{1}{2} v_1 (senh^2(\frac{\sqrt{v_1}(x+8)}{2})) + \frac{1}{2} v_2 (senh^2(\frac{\sqrt{v_2}(x+8)}{2}))$$

Figura 11: Solución inicial del problema

Pseudocódigo KdV.

A continuación se detalla el proceso que se ejecuta al correr el programa. Quedan implícitas las asignaciones a variables como el paso espacial y velocidades para cada solitón, así como también la obtención de gráficas del movimiento de los mismos.

Choose splitting method (Lie Trotter / Strang)
Choose method order q
s ← q / 2
for t = 0 : tmax
    U ← 0
    for m = 1 : s
        Calculate $\Phi_m^+(\frac{dt}{m})$
        Calculate $\Phi_m^-(\frac{dt}{m})$
        U ← U + $\gamma_m(\Phi_m^+(\frac{dt}{m}) + \Phi_m^-(\frac{dt}{m}))$
    end
end

Desarrollo

Para el desarrollo se utilizó MATLAB R2019b con la extensión *Parallel Computing Toolbox* para distribuir el procesamiento entre múltiples "workers".

Se implementaron los SSM de Lie-Trotter y Strang, encontrándose cada uno de estos en su archivo correspondiente. Para obtener la transformada de Fourier y la transformada inversa se desarrollaron implementaciones propias de las funciones *fft* y *ifft*.

Se desarrolló, asimismo, una versión secuencial del script para comparar el rendimiento de ambas implementaciones. Ésta no hace uso de ningún pool de workers, por lo que no distribuye el procesamiento.

Pruebas

Las pruebas fueron inicialmente realizadas utilizando el método de Euler [5], y luego se repitieron reemplazando éste método por el método RK4 [6].

Para calcular el error global, se mantuvo constante el orden y se varió el paso (h) temporal (probando los valores 1×10^{-4}, 1×10^{-5} y 1×10^{-6}), comparando en cada caso el resultado obtenido con el paso h y el resultado obtenido con el paso $2h$. Para obtener el error, se restaron los resultados de ambas corridas y se aplicó la norma infinito al vector resultante.

Luego, para calcular el error local se realizaron numerosas ejecuciones manteniendo constante el paso temporal y variando el orden (se probaron los valores 2, 4, 6 y 8), utilizando en todos los casos el método propuesto por Strang. Como en el caso anterior, para calcular el error se restaron los resultados obtenidos y se aplicó la norma infinito.

Finalmente, se realizaron múltiples corridas con parámetros idénticos (midiendo la duración de cada corrida), tanto en serie como en paralelo, con el fin de obtener una métrica que permita comparar la eficiencia temporal de ambos tipos de ejecución.

Resultados

Los resultados de las pruebas fueron, en todos los casos, más precisos utilizando el método RK4, y consistentes con los resultados obtenidos con el método de Euler, por lo que sólo se analizarán los resultados de las pruebas que hubieran utilizado el primer método, es decir, RK4.

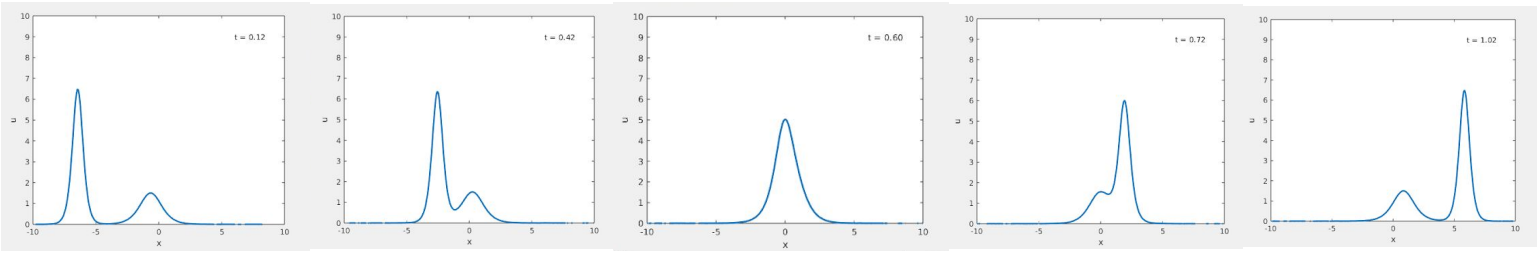


Figura 12: Desplazamiento de los solitones al transcurrir el tiempo durante una ejecución del programa.

Como puede observarse de la Figura 12, se verificó que la implementación de KdV propuesta funcionara correctamente a simple vista. Efectivamente, al correr el tiempo los solitones se desplazan a la derecha y al momento de superponerse forman una única onda de amplitud intermedia entre ambas. Luego, las ondas logran separarse para seguir cada una su recorrido normal. Asimismo, se verificó que existieran condiciones periódicas al volver a aparecer un solitón del lado izquierdo luego de alcanzar el extremo opuesto.

Los primeros resultados arrojaron un error global decreciente según decrece el paso temporal, alcanzando magnitudes insignificantes (del orden de 1×10^{-6}) cuando el paso es 1×10^{-5}. Se observó una pendiente de carácter lineal para el caso de Lie-Trotter, mientras que con Strang se adquirieron errores "exponencialmente" mayores al aumentar el paso temporal.

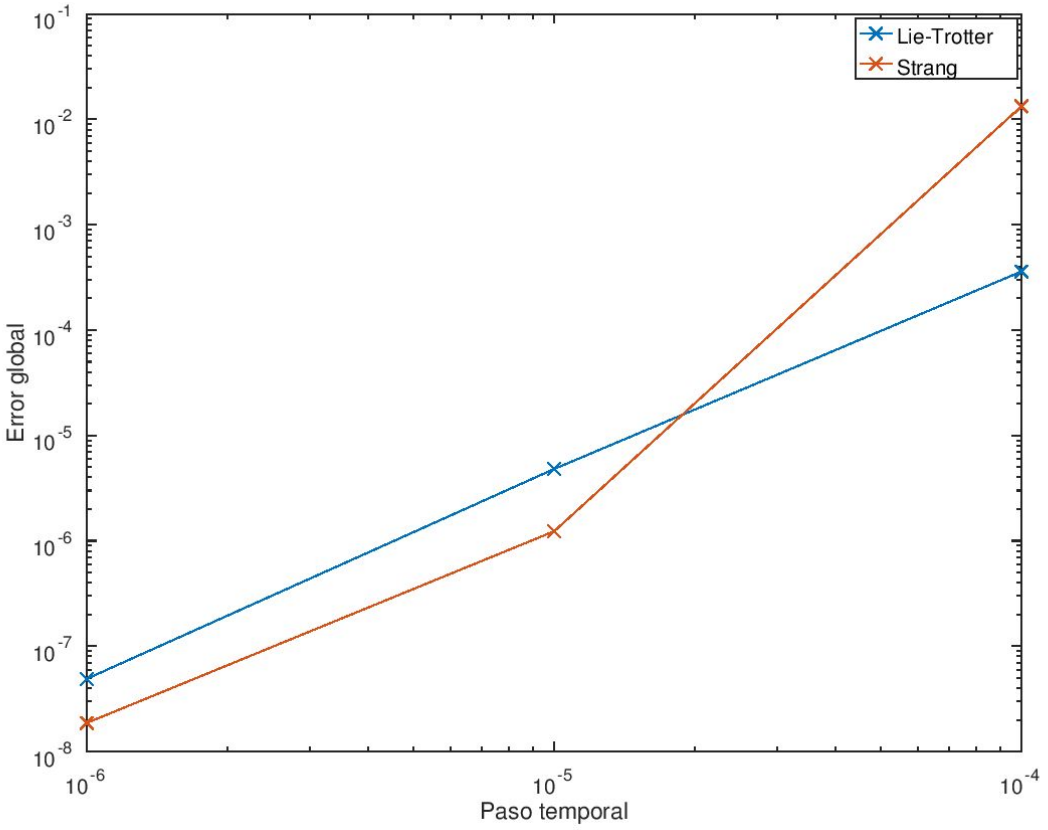


Figura 13: Error global en función del paso temporal, utilizando métodos de orden 8.

En base a dichos resultados, se eligió un paso temporal de 1×10^{-5} para utilizar en las pruebas subsiguientes al considerar innecesario utilizar un paso menor que, si bien obtendría errores menores, dicha ganancia con respecto al mayor tiempo requerido para ejecutar el programa no estaría justificada, a nuestra consideración.

Con respecto al error local, se observó que decrecía a medida que aumentaba el orden del método utilizado (en este caso, Strang), siendo la diferencia entre el orden 2 y el orden 4 mucho mayor a la diferencia entre el orden 4 y el orden 6. Se compararon los resultados obtenidos con aquellos arrojados al ejecutar el algoritmo con orden 8.

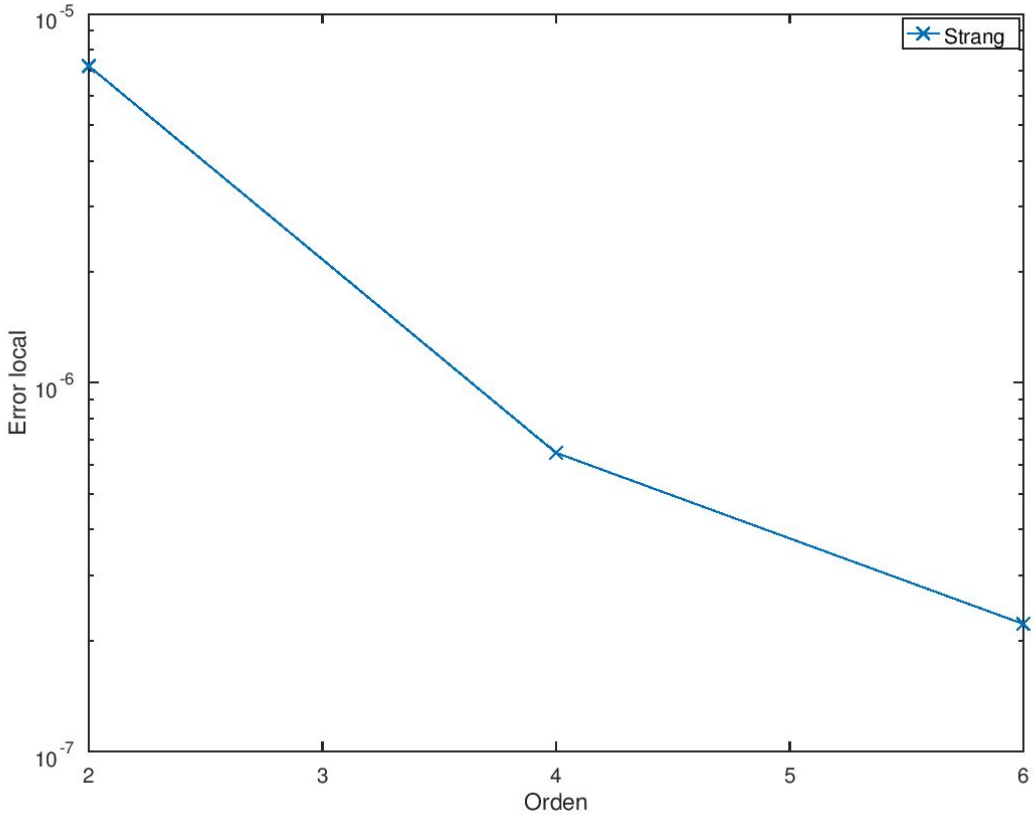


Figura 14: Error local en función del orden, utilizando el método de Strang y un paso temporal de 1×10^{-5}.

Para comparar las ejecuciones en serie con las ejecuciones en paralelo se realizaron diez ejecuciones en serie para cada orden, y cinco ejecuciones en paralelo para cada combinación de parámetros. Luego se promediaron los tiempos de ejecución en serie y en paralelo (por separado), y finalmente se calculó el "speedup", esto es, la mejora en la velocidad de ejecución de una tarea ejecutada en dos arquitecturas similares con diferentes recursos [8]. El cálculo se realizó basándose en la Ley de Amdahl, mediante la siguiente fórmula:

$$A = \frac{T\ serie}{T\ paralelo}$$

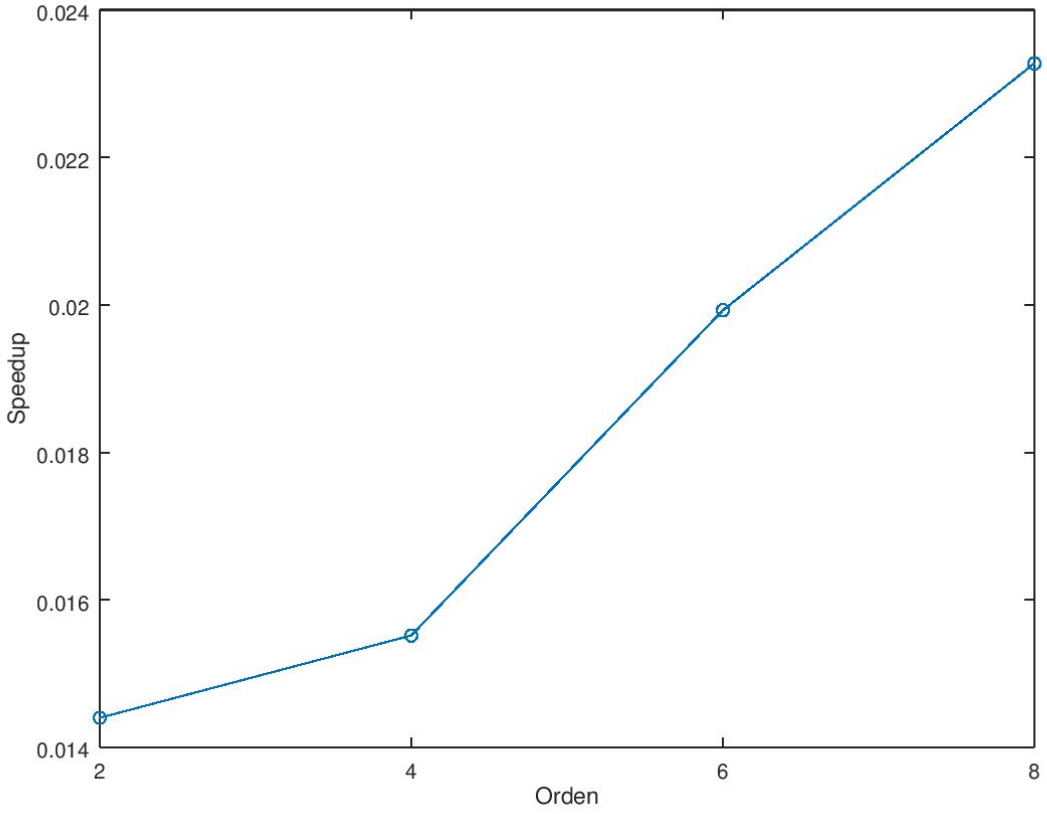
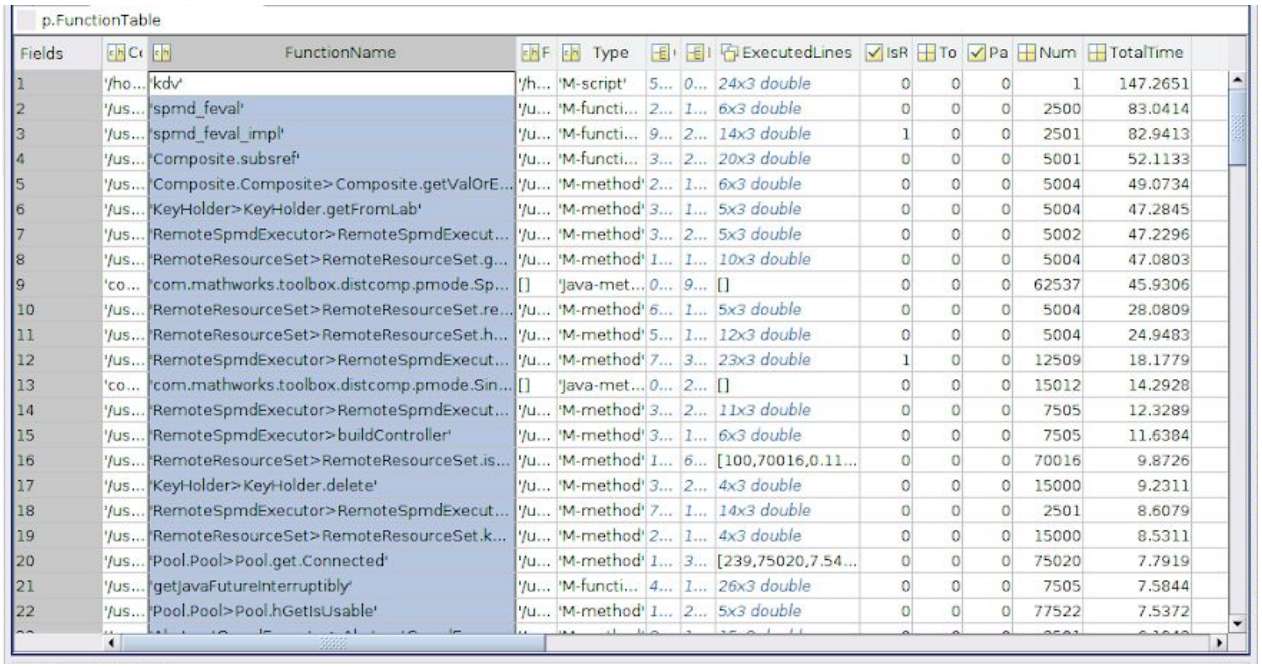


Figura 15: Speedup para ejecuciones de órdenes 2, 4, 6 y 8 respectivamente, utilizando el método de Strang, un paso de 1×10^{-5} y $t_{max} = 0.1\ s$.

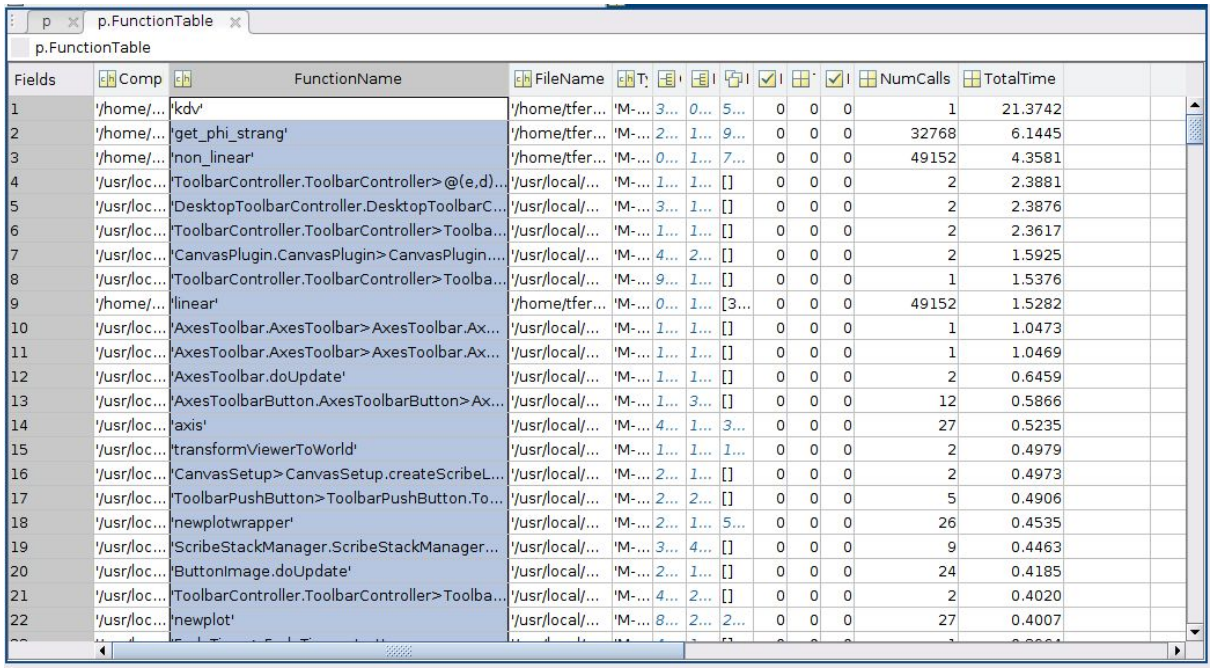
Los tiempos de ejecución en paralelo fueron más de diez veces mayores a los tiempos de ejecución en serie, causando que el speedup sea mucho menor a 1. La amplia diferencia en los tiempos de ejecución se debe al elevado throughput temporal resultante de utilizar la librería de ejecución en paralelo de MATLAB. Ejecutando corridas con un profiler resultó evidente que las funciones que más veces fueron ejecutadas y que más tiempo estuvieron ejecutándose (a excepción del script kdv.m, que por obvios motivos fue el que mayor tiempo de ejecución tuvo) fueron funciones internas relacionadas con spmd. Ejecutando una corrida similar en serie resultó notable la ausencia de llamados a estas funciones.



p.FunctionTable

Fields	FunctionName	Type	ExecutedLines	IsR	To	Pa	Num	TotalTime
1	'kdv'	'M-script'	24x3 double	0	0	0	1	147.2651
2	'spmd_feval'	'M-functi...	6x3 double	0	0	0	2500	83.0414
3	'spmd_feval_impl'	'M-functi...	14x3 double	1	0	0	2501	82.9413
4	'Composite.subsref'	'M-functi...	20x3 double	0	0	0	5001	52.1133
5	'Composite.Composite>Composite.getValOrE...	'M-method'	6x3 double	0	0	0	5004	49.0734
6	'KeyHolder>KeyHolder.getFromLab'	'M-method'	5x3 double	0	0	0	5004	47.2845
7	'RemoteSpmdExecutor>RemoteSpmdExecut...	'M-method'	5x3 double	0	0	0	5002	47.2296
8	'RemoteResourceSet>RemoteResourceSet.g...	'M-method'	10x3 double	0	0	0	5004	47.0803
9	'com.mathworks.toolbox.distcomp.pmode.Sp...	'Java-met...	[]	0	0	0	62537	45.9306
10	'RemoteResourceSet>RemoteResourceSet.re...	'M-method'	5x3 double	0	0	0	5004	28.0809
11	'RemoteResourceSet>RemoteResourceSet.h...	'M-method'	12x3 double	0	0	0	5004	24.9483
12	'RemoteSpmdExecutor>RemoteSpmdExecut...	'M-method'	23x3 double	1	0	0	12509	18.1779
13	'com.mathworks.toolbox.distcomp.pmode.Sin...	'Java-met...	[]	0	0	0	15012	14.2928
14	'RemoteSpmdExecutor>RemoteSpmdExecut...	'M-method'	11x3 double	0	0	0	7505	12.3289
15	'RemoteSpmdExecutor>buildController'	'M-method'	6x3 double	0	0	0	7505	11.6384
16	'RemoteResourceSet>RemoteResourceSet.is...	'M-method'	[100,70016,0.11...	0	0	0	70016	9.8726
17	'KeyHolder>KeyHolder.delete'	'M-method'	4x3 double	0	0	0	15000	9.2311
18	'RemoteSpmdExecutor>RemoteSpmdExecut...	'M-method'	14x3 double	0	0	0	2501	8.6079
19	'RemoteResourceSet>RemoteResourceSet.k...	'M-method'	4x3 double	0	0	0	15000	8.5311
20	'Pool.Pool>Pool.get.Connected'	'M-method'	[239,75020,7.54...	0	0	0	75020	7.7919
21	'getJavaFutureInterruptibly'	'M-functi...	26x3 double	0	0	0	7505	7.5844
22	'Pool.Pool>Pool.hGetIsUsable'	'M-method'	5x3 double	0	0	0	77522	7.5372

Figura 16: Resultados de profiler (ejecución en paralelo)
Columna seleccionada: nombres de funciones ejecutadas. Primera columna de la izquierda: Tiempo total de ejecución por función
Segunda columna de la izquierda: Número de ejecuciones por función



p p.FunctionTable

p.FunctionTable

Fields	Comp	FunctionName	FileName	NumCalls	TotalTime
1	'/home/...	'kdv'	'/home/tfer...	1	21.3742
2	'/home/...	'get_phi_strang'	'/home/tfer...	32768	6.1445
3	'/home/...	'non_linear'	'/home/tfer...	49152	4.3581
4	'/usr/loc...	'ToolbarController.ToolbarController>@(e,d)...	'/usr/local/...	2	2.3881
5	'/usr/loc...	'DesktopToolbarController.DesktopToolbarC...	'/usr/local/...	2	2.3876
6	'/usr/loc...	'ToolbarController.ToolbarController>Toolba...	'/usr/local/...	2	2.3617
7	'/usr/loc...	'CanvasPlugin.CanvasPlugin>CanvasPlugin....	'/usr/local/...	2	1.5925
8	'/usr/loc...	'ToolbarController.ToolbarController>Toolba...	'/usr/local/...	1	1.5376
9	'/home/...	'linear'	'/home/tfer...	49152	1.5282
10	'/usr/loc...	'AxesToolbar.AxesToolbar>AxesToolbar.Ax...	'/usr/local/...	1	1.0473
11	'/usr/loc...	'AxesToolbar.AxesToolbar>AxesToolbar.Ax...	'/usr/local/...	1	1.0469
12	'/usr/loc...	'AxesToolbar.doUpdate'	'/usr/local/...	2	0.6459
13	'/usr/loc...	'AxesToolbarButton.AxesToolbarButton>Ax...	'/usr/local/...	12	0.5866
14	'/usr/loc...	'axis'	'/usr/local/...	27	0.5235
15	'/usr/loc...	'transformViewerToWorld'	'/usr/local/...	2	0.4979
16	'/usr/loc...	'CanvasSetup>CanvasSetup.createScribeL...	'/usr/local/...	2	0.4973
17	'/usr/loc...	'ToolbarPushButton>ToolbarPushButton.To...	'/usr/local/...	5	0.4906
18	'/usr/loc...	'newplotwrapper'	'/usr/local/...	26	0.4535
19	'/usr/loc...	'ScribeStackManager.ScribeStackManager...	'/usr/local/...	9	0.4463
20	'/usr/loc...	'ButtonImage.doUpdate'	'/usr/local/...	24	0.4185
21	'/usr/loc...	'ToolbarController.ToolbarController>Toolba...	'/usr/local/...	2	0.4020
22	'/usr/loc...	'newplot'	'/usr/local/...	27	0.4007

Figura 17: Resultados de profiler (ejecución en serie)
Columna seleccionada: nombres de funciones ejecutadas. Primera columna de la izquierda: Tiempo total de ejecución por función
Segunda columna de la izquierda: Número de ejecuciones por función

Conclusiones

Resultó sumamente interesante el análisis de la ecuación KdV y el comportamiento obtenido de aplicar SSM junto a paralelización en el desarrollo del programa. De las pruebas relacionadas al error global, se concluyó que pasos temporales elevados derivaron en menor precisión, siendo el método de Strang preferible por encima de Lie-Trotter en pasos muy pequeños, lo cual indica una mejora obtenida por un método de mayor orden. Resultaría de interés observar si dicha afirmación se cumple al utilizar otros métodos de órdenes aún mayores, como ser las implementaciones de Ruth y Neri, entre otros, para futuras investigaciones. Asimismo, se logró concluir que resultados obtenidos de utilizar órdenes elevados tuvieron diferencias menores que al utilizar órdenes inferiores, señalando cierta convergencia para las soluciones alcanzadas al aumentar el orden de los SSM utilizados. Finalmente, en cuanto al procesamiento en paralelo se determinó que a mayor cantidad de hilos se obtuvieron mayores ganancias. Si bien no se vieron reflejadas en las pruebas realizadas al comparar con los tiempos de ejecuciones en serie, debería resultar evidente utilizando órdenes mucho más elevados que los utilizados en el presente informe, así como también equipos de procesamiento más potentes y avanzados que puedan llevar a cabo dichas tareas. Sin embargo, cabe aclarar que no siempre la paralelización es deseada, y debería observarse que el speedup alcanza un límite, reflejado asintóticamente en el gráfico correspondiente, y en cumplimiento con la Ley de Amdahl.

Bibliografía

[1] Cooley J. W. Tukey J. W. An algorithm for the machine calculation of complex fourier series. Math. Comput., 19:297–301, 1965.
[2] Trotter H.F. On the product of semigroups of operators. 10:545–551, 1959. Proc. Amer. Math. Soc.
[3] Strang G. Accurate partial difference methods ii: Non linear problems. Numerische Math., 6:37–46, 1963.
[4] Alvarez A. Rial D. Affine combination of splitting type integrators implemented with parallel computing methods. International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering, 9(2):146–149, 2015
[5] https://es.wikipedia.org/wiki/Método_de_Euler (19-11-2019)
[6] https://es.wikipedia.org/wiki/Método_de_Runge-Kutta (19-11-2019)
[7] S. Lopez-Aguayo, M. Esparza-Echevarría, G. Lem-Carrillo y J. C. Gutierrez-Vega, (2014) “Ondas solitarias no lineales: una introducción a los solitones ópticos espaciales”, http://www.scielo.org.mx/pdf/rmfe/v60n1/v60n1a4.pdf (19-11-2019)
[8] https://es.wikipedia.org/wiki/Speedup (19-11-2019)