Trabajo Práctico 2

Métodos Numéricos Avanzados

Integrantes: Augusto Henestrosa, Guido Barbieri, Agustín Roca, Nicolás Britos, Catalina Varela, Francisco Choi.

Profesor: Adrián Omar Álvarez.

Introducción

El objetivo de este trabajo práctico es resolver la ecuación diferencial parcial de Kuramoto–Sivashinsky (KS) mediante el uso de métodos llamados Spectral Splitting Methods (SSM) y comparar los resultados de los mismos.

Marco teórico

La ecuación de Kuramoto-Sivashinsky (KS):

La ecuación KS es la siguiente:

$$u_t = -uu_x - u_{xx} - u_{xxxx},\ x \in [0, 32\pi] \tag{1}$$

Es una ecuación diferencial parcial no lineal de cuarto orden. La ecuación fue derivada en 1970 por Yoshiki Kuramoto y Gregory Sivashinsky. Se resolverá la ecuación de la misma manera que el paper que la consigna recomendó tomar como referencia.

$$u(x, t = 0) = cos(\tfrac{x}{16})(1 + sin(\tfrac{x}{16})) \tag{2}$$

Transformando la ecuación (1) aplicando la transformada de Fourier se obtiene:

$$\widehat{u_t} = -\tfrac{ik}{2}\widehat{u^2} + (k^2 - k^4)\widehat{u} \tag{3}$$

De acá separamos la parte lineal de la no lineal, por lo que:

$$(L\widehat{u})(k) = (k^2 - k^4)\widehat{u} \tag{4}$$

es la parte lineal y,

$$N(\widehat{u},\ t) = N(\widehat{u}) = -\tfrac{ik}{2}\,(F((F^{-1}(\widehat{u}))^2)) \tag{5}$$

es la parte no lineal.

Resolvemos la ecuación (4) analiticamente,

$$\widehat{u} = e^{(k^2-k^4)h} \tag{6}$$

donde h = $t-t_0$

La ecuación (5) no puede resolverse analiticamente por lo que obtenemos una solución aproximada utilizando la transformada rápida de Fourier (FFT) y Runge-Kutta de orden 4.

Una vez obtenidas las soluciones de ambas partes, se les aplica las composiciones de los integradores. Estos integradores son:

- Asimétrico:

$$\Phi(h) = \sum_{j=1}^{s} \gamma_j \Phi_j^{\pm}(h/j) \tag{7}$$

- Simétrico:

$$\Phi(h) = \sum_{j=1}^{s} \gamma_j (\Phi_j^{+}(h/j) + \Phi_j^{-}(h/j)) \tag{8}$$

Donde,

$$\Phi^{+}(h) = \phi_1(h) \circ \phi_0(h) \tag{9}$$

$$\Phi^{-}(h) = \phi_0(h) \circ \phi_1(h) \tag{10}$$

$$\Phi_j^{\pm}(h) = \begin{cases} \Phi^{\pm}(h) \circ \Phi_{j-1}^{\pm}(h) & \text{si j > 1} \\ \Phi^{\pm}(h) & \text{si j = 1} \end{cases} \tag{11}$$

y para las constantes γ_j se deben cumplir las siguientes condiciones en cada caso:

- Asimétrico:

$$1 = \sum_{j=1}^{s} \gamma_j \tag{12}$$

$$0 = \sum_{j=1}^{s} j^{-k} \gamma_j, \ 1 \le k \le q-1 \tag{13}$$

- Simétrico:

$$\frac{1}{2} = \sum_{j=1}^{s} \gamma_j \tag{14}$$

$$0 = \sum_{j=1}^{s} j^{-2k} \gamma_j, \ 1 \le k \le n-1 \tag{15}$$

En el caso de los integradores simetricos n = q/2 y el q solo puede ser par.

Implementación

El trabajo fue realizado en MATLAB utilizando la librería Parpool para la paralelización de los integradores. Los gráficos fueron hechos en Google Sheets y en Python.

La idea de la implementación fue poder modularizar el código lo suficiente como para que se pueda utilizar cualquier método de integración que se desee para aproximar la ecuación de KS de acuerdo con la condición inicial que se establezca.

Es importante destacar que para la implementación en paralelo se paralelizaron los términos de las sumatorias de las ecuaciones (7) y (8).

Análisis de los resultados

Precisión

Para medir la precisión de los métodos propuestos utilizaremos las siguientes fórmulas de error:

- Para medir el error variando el paso Δt usaremos la siguiente fórmula:

$$E = || f(\Delta t) - f(\Delta t/2) ||_\infty \quad (16)$$

- Para medir el error variando el orden n usaremos la siguiente fórmula:

$$E = || f_n(\Delta t) - f_{n+1}(\Delta t) ||_\infty \quad (17)$$

Comparación de los métodos de integración no afines

En la figura 1, analizamos cuál método funciona mejor como integrador de acuerdo al error dado. Para Δt muy chicos, Neri es el más preciso entre todos pero pierde esta precisión rápidamente a medida que aumenta el Δt. Los otros métodos crecen más lentamente a medida que aumenta el paso. Asimismo, podemos decir que el método que mejor funciona para Δt más altos es el de Strang.

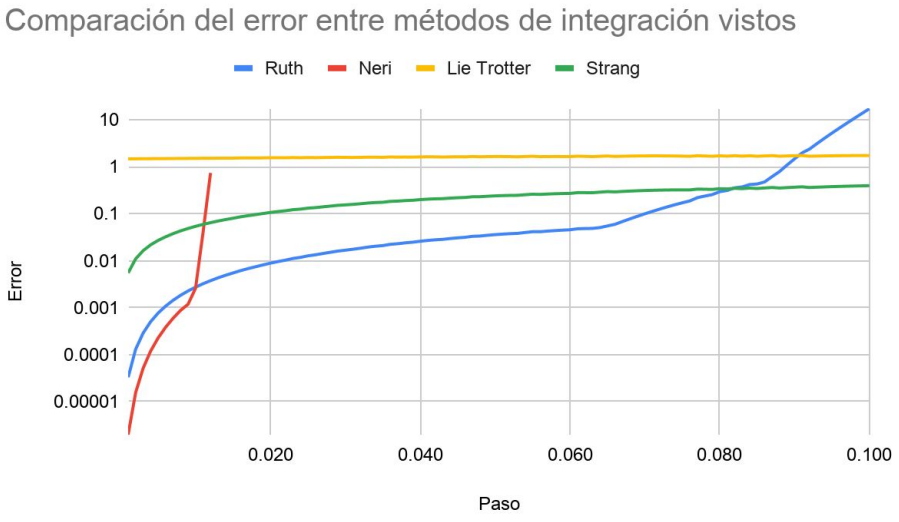

Figura 1. Errores de los distintos métodos de integración usando como paso Δt=0.001 a Δt=0.1

Comparación de los errores según el orden: métodos afines

En el caso del método afín asimétrico nos encontramos con lo esperado que es que a mayor orden menor es el error. La diferencia entre errores de órdenes consecutivos decrece a mayores órdenes. Sin embargo, para nuestra sorpresa, en el caso del método afín simétrico ocurrió lo contrario, el orden con mayor error resultó ser el de orden 6 y el de menor error el de orden 2.

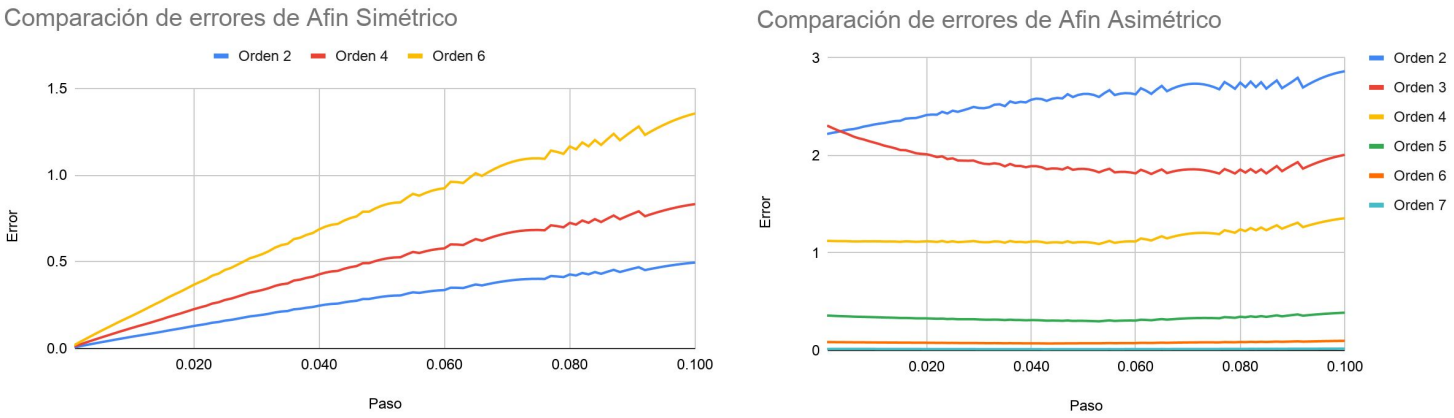

Figura 2. Errores de los métodos de afin de distintos órdenes usando como paso Δt=0.001 a Δt=0.1

Comparación de métodos con métodos de su mismo orden

Podemos ver que a medida que aumenta Δt, el error tiende a aumentar. Para el caso de Neri, crece demasiado rápido el error una vez superado el paso 0.01 mientras que los métodos afines el error crece más suavemente. A pesar de que el afin simétrico crece más rápidamente que el asimétrico, el afin simétrico se mantiene con menor error para todos los Δt estudiados.

Figura 3. Errores de los métodos de orden 4 usando como paso Δt=0.001 a Δt=0.1

En general se puede ver que Δt chicos los métodos simplécticos suelen ser más efectivos mientras que los otros métodos tienden a ser más precisos a Δt mayores. Para visualizar mejor esto se decidió comparar el método de Ruth contra el afin asimétrico en el tercer orden como se puede ver en la figura 4.

Figura 4. Errores del método de Ruth y el método afín asimétrico de orden 3 usando como paso $\Delta t=0.001$ a $\Delta t=0.1$

Aumento de velocidad por paralelización

Para analizar el aumento de velocidad en el tiempo de respuesta debido a la paralelización vamos a utilizar el concepto de *speedup*. El speedup permite medir la performance relativa de dos sistemas en ejecutar la misma tarea. En nuestro caso, los dos sistemas son la misma computadora corriendo el programa en serie y en paralelo. La fórmula de speedup que utilizaremos será la siguiente:

$$S = \frac{t_{serie}}{t_{paralelo}} \tag{18}$$

donde t es el tiempo de ejecución en cada caso.

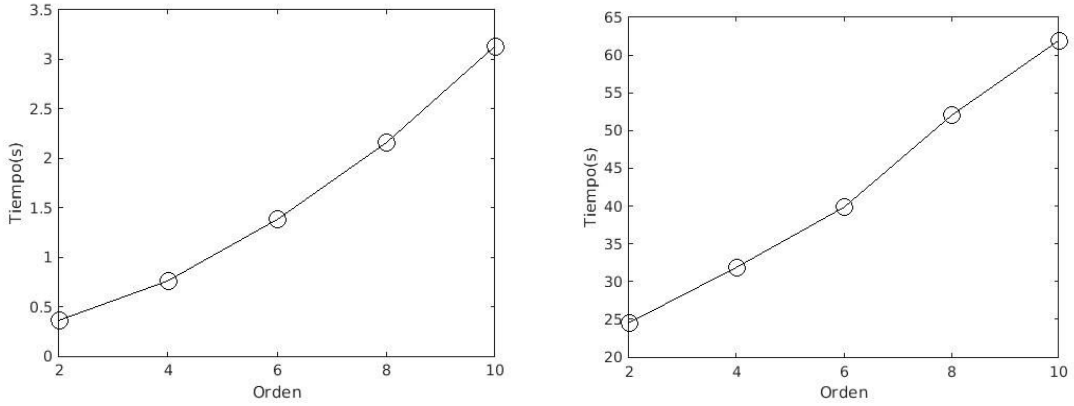

Figura 5: Tiempos de ejecución con el método afín simétrico en serie (izquierda) y en paralelo (derecha) a distintos órdenes

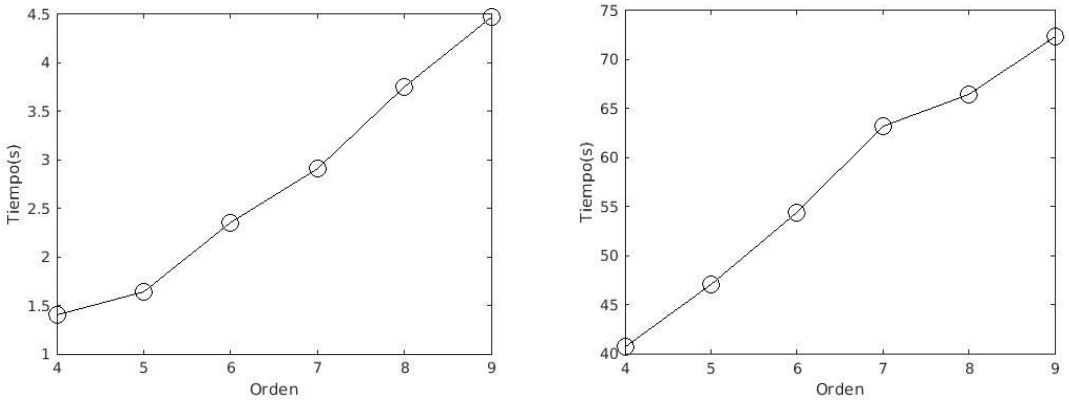

Figura 6: Tiempos de ejecución con el método afín asimétrico en serie (izquierda) y en paralelo (derecha) a distintos órdenes

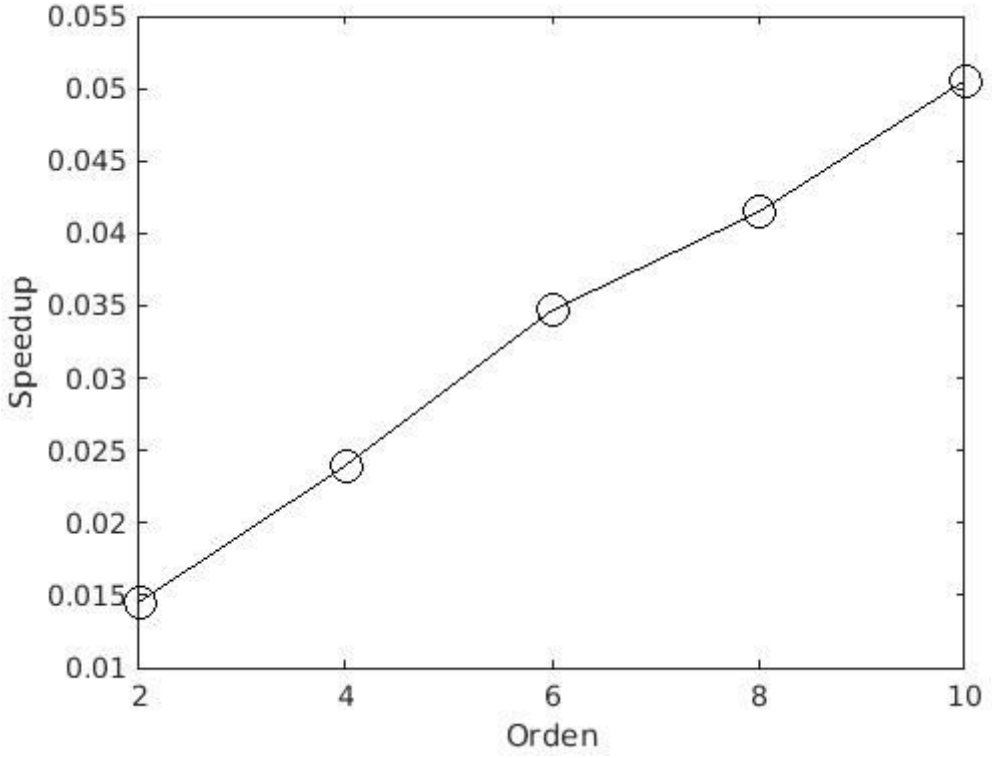

Figura 7: Resultado de Speedup con el método afín simétrico a distintos órdenes

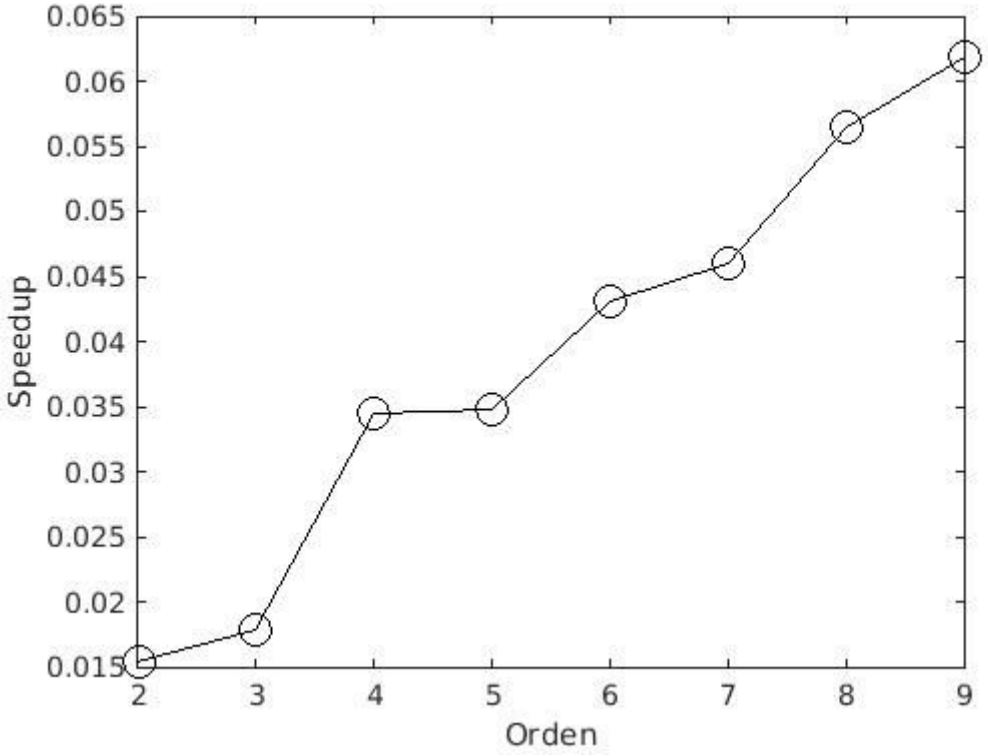

Figura 8: Resultado de Speedup con el método afín asimétrico a distintos órdenes

Como era de esperarse los tiempos de ejecución a mayores órdenes resultaron mayores como se puede apreciar en la figura 5 para el método afín simétrico y en la figura 6 para el método afín asimétrico. Sin embargo, sorprendentemente, en este caso resultó que paralelizar el proceso no resultó beneficioso sino al contrario. El proceso en paralelo llegó a ser más de 10 veces más lento que en serie tanto en el método simétrico (figura 7) como en el asimétrico (figura 8). Nosotros creemos que esto puede ser porque al paralelizar se pueden perder optimizaciones entre los pasos del procesos, además de tener que sumarle el *overhead* que trae de por sí el paralelismo al tener que crear distintos procesos y distribuir el código y recolectar los resultados al final.

Algo que es importante destacar también es que a mayores órdenes el speedup es mayor. Esto quiere decir que se aprovecha mejor el paralelismo a mayores órdenes. Sin embargo, sigue siendo más rápido correr el programa en serie que en paralelo en todos los órdenes probados.

Efecto de perturbaciones espaciales aleatorias

Para analizar el efecto que tienen perturbaciones espaciales aleatorias en la condición inicial de la ecuación graficamos los resultados obtenidos en una animación.

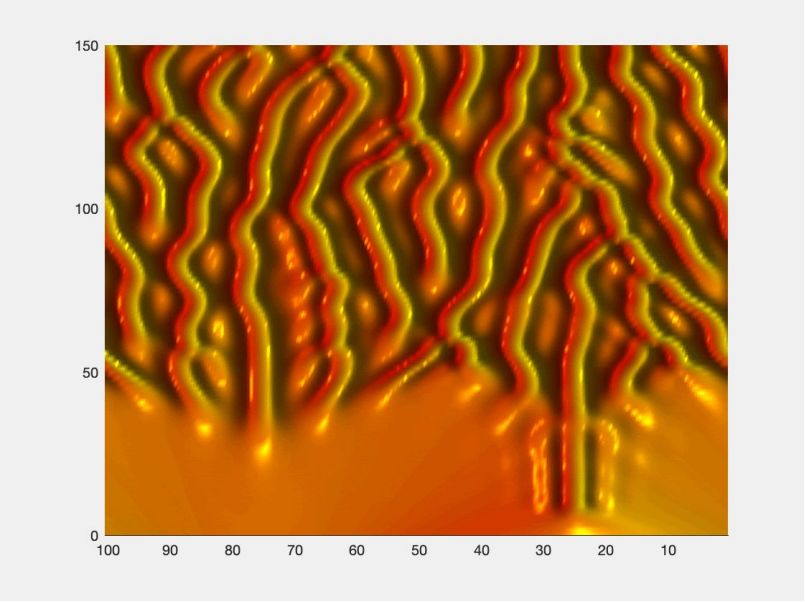

Figura 5. Animación de la solución de la ecuación KS utilizando el método de Lie Trotter con una perturbación aleatoria del 1%.

Conclusiones

Según los resultados obtenidos en este trabajo, podemos concluir que los integradores no simplécticos que se definen en el paper que presenta la consigna son muy efectivos cuando se trate de resolver ecuaciones diferenciales de órdenes altos con pasos no muy pequeños comparado con los integradores simplécticos. Esto es porque en los integradores simplecticos el error crece demasiado a medida que se aumenta el paso. En cambio, los otros integradores a pesar de ser menos precisos a pasos chicos, no pierden precisión tan drásticamente como los simplecticos.

Por otro lado, si el problema llegase a necesitar un poder de cómputo muy grande, se podría resolver paralelizando. Pero no en todos los casos conviene hacerlo, ya que el paralelismo viene con un costo intrínseco que es el de la creación de procesos o threads y la comunicación entre ellos que puede llegar a ser mayor que el beneficio que se pueda lograr paralelizando.

Bibliografía

1. https://www.researchgate.net/publication/294715506_Affine_Combination_of_Splitting_Type_Integrators_Implemented_with_Parallel_Computing_Methods
2. https://people.maths.ox.ac.uk/trefethen/publication/PDF/2005_111.pdf
3. https://en.wikipedia.org/wiki/Speedup
4. https://pdfs.semanticscholar.org/fb83/3754f58e5488739b27a40e6d597d75d8763b.pdf
5. https://people.maths.ox.ac.uk/trefethen/publication/PDF/2005_111.pdf3c©
6. Cooley J. W. Tukey J. W. An algorithm for the machine calculation ofcomplex fourier series.Math. Comput., 19:297–301, 1965
7. Trotter H.F. On the product of semigroups of operators. 10:545–551, 1959.Proc. Amer. Math. Soc
8. Alvarez A. Rial D. Affine combination of splitting type integrators imple-mented with parallel computing methods.International Journal of Mat-hematical, Computational, Physical, Electrical and Computer Engineering,9(2):146–149, 2015.
9. Merson R. H. An operational method for the study of integration processes.Proc. Symp. Data Processing , Weapons Res. Establ. Salisbury, I:110–125,1957.