Introducción

En todos los casos consideramos métodos no adaptativos, por lo que el paso de evolución será fijo e igual a $h = {}^{(b-a)}/_{(n-1)}$, donde n es el número de puntos y $(n-1)$ el número de intervalos.

Método de Euler

En este método se toma la pendiente en el punto actual como una aproximación de la pendiente promedio sobre todo el intervalo, esta pendiente se utiliza para extrapolar linealmente la solución desde punto actual hacia el punto siguiente (puntos separados por una distancia h ó paso).

El error de truncamiento local del método se calcula conociendo los términos remanentes del desarrollo en serie de taylor de la solución que no son considerados. De forma aproximada, si los h son pequeños, el error de truncamiento local será aproximadamente del orden $O\big(h^2\big)$ entonces,

$$\epsilon^{euler}_{trunc,loc} = \frac{f'(x_i, y_i)}{2!}h^2 \tag{1}$$

Métodos de Runge Kutta de 2do orden

Se llama de 2do orden debido a que el método utiliza dos puntos dentro de cada intervalo para estimar la pendiente sobre todo el intervalo.

$$y_{(i+1)} = y_i + (a_1k_1 + a_2k_2)h \tag{2}$$

donde $k_1 = f(x_i, y_i)$ y $k_2 = f(x_i + p_1h, y_i + q_{11}k_1h)$ y los valores de a_1, a_2, p_1 y q_{11} se evalúan al igualar la ecuación para $y_{(i+1)}$ con su expansión en serie de Taylor hasta el término de segundo orden, de esta forma se obtiene el siguiente sistema de ecuaciones

$$a_1 + a_2 = 1;\ a_2p_1 = {}^1/_2;\ a_2q_{11} = {}^1/_2 \tag{3}$$

Como tenemos tres ecuaciones para cuatro incógnitas hay un número infinito de métodos de RK de 2do orden qué arrojan exactamente la misma solución si la solución exacta es de hasta 2do orden, caso contrario tendrán ciertas diferencias.

<u>Método de Heun con un solo corrector</u> $\left(a_2 = {}^1/_2\right)$ entonces obtenemos

$$y_{(i+1)} = y_i + \frac{h}{2}(k_1 + k_2);\ k_1 = f(x_i, y_i);\ k_2 = f(x_i + h, y_i + k_1h) \tag{4}$$

<u>Método del punto medio</u> $(a_2 = 1)$ entonces obtenemos

$$y_{(i+1)} = y_i + k_2h;\ k_1 = f(x_i, y_i);\ k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1h\right) \tag{5}$$

<u>Método de Raltson</u> $\left(a_2 = {}^2/_3\right)$ entonces obtenemos

$$y_{(i+1)} = y_i + \frac{h}{3}(k_1 + 2k_2);\ k_1 = f(x_i, y_i);\ k_2 = f\left(x_i + \frac{3}{4}h, y_i + \frac{3}{4}k_1h\right) \tag{6}$$

Raltson(1962) y Rabinowitz(1978) demostraron que para $a_2 = {}^2/_3$ se obtiene un mínimo en el error de truncamiento para los métodos de RK de 2do orden.

Los errores locales de los métodos de RK de 2do orden son del orden de $\epsilon^{RK2}_{trunc,loc} \approx O\big(h^3\big)$ y los globales serán $\epsilon^{RK2}_{trunc,glob} = n \cdot \epsilon^{RK2}_{trunc,loc} \approx O\big(h^2\big)$ pues $n \propto {}^1/_h$.

Método de Runge Kutta de 4to orden

Se llama de 4to orden debido a que el método utiliza cuatro puntos dentro de cada intervalo para estimar la pendiente sobre todo el intervalo. Además, también existen infinitos métodos de Runge Kutta y el más común (método clásico) es el siguiente

$$\begin{gathered} y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 = f(x_i, y_i);\ k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h\right) \\ k_3 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2 h\right);\ k_4 = f(x_i + h, y_i + k_3 h) \end{gathered} \tag{7}$$

Los errores locales de los métodos de RK de 4to orden son del orden de $\epsilon_{trunc,loc}^{RK4} \approx O\big(h^5\big)$ y los globales serán $\epsilon_{trunc,glob}^{RK4} = n \cdot \epsilon_{trunc,loc}^{RK4} \approx O\big(h^4\big)$.

Resultados y Discusiones

Para calcular los números de puntos óptimos a optimizar deberemos minimizar el error total, esto se hace de la siguiente manera,

$$\epsilon_{tot} = \epsilon_{trunc,glob}^{method} + \epsilon_{glob}^{machine} \approx \frac{\alpha}{n^{\beta}} + \sqrt{n}\epsilon_M \tag{8}$$

$$\Rightarrow \begin{cases} \epsilon_{tot}^{euler} \approx \epsilon_{tot}^{RK2} \approx \dfrac{1}{n^2} + \sqrt{n}\epsilon_M & \Rightarrow \dfrac{d\epsilon_{tot}^{euler}}{dn} = 0 \Rightarrow n_{op}^{euler} = n_{op}^{RK2} = \left(\dfrac{4}{\epsilon_M}\right)^{2/5} \approx O\big(10^6\big) \\ \epsilon_{tot}^{RK4} \approx \dfrac{1}{n^4} + \sqrt{n}\epsilon_M & \Rightarrow \dfrac{d\epsilon_{tot}^{RK4}}{dn} = 0 \Rightarrow n_{op}^{RK4} = \left(\dfrac{8}{\epsilon_M}\right)^{2/5} \approx O\big(10^3\big) \end{cases} \tag{9}$$

donde los resultados de n del orden de $O\big(10^6\big)$ para Euler y RK2 y del orden de $O\big(10^3\big)$ para RK4 se obtuvieron bajo la hipótesis de trabajar todo el algoritmo con precisión doble $\epsilon_M \approx O\big(10^{-15}\big)$. Esto nos muestra que el método de RK4 requiere puntos que tienen un orden de mangintud tres veces menor que los otros métodos para alcanzar su error mínimo.

Los resultados obtenidos, simulando para un número de puntos igual a $n = 2^{10} = 1024$ fueron

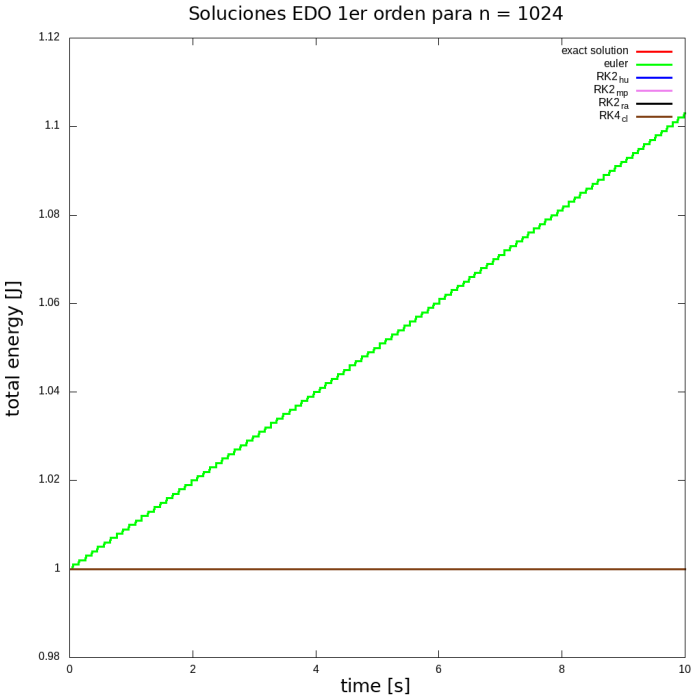


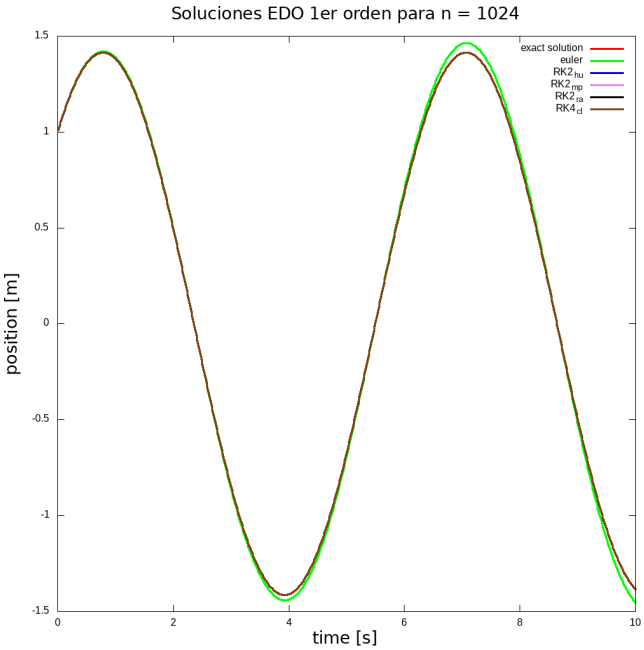
Figure 1: Energía total $= (E_{cin} + E_{pot})$ vs tiempo (real time)(link)

Notemos que para el método de Euler la energía no se conserva, mientras que para los métodos de RK si, esto es debido a que estos últimos incluyes términos de orden superior del desarrollo en serie de Taylor de la solución y como estos términos de orden superior son fundamentales para el cálculo de la energía, no incluirlos ocasionan una no conservación de la energía. Esto nos muestra que el método de Euler, para este caso de soluciones armónicas no son buenos métodos numéricos, pues al no conservar la energía, la física del problema no se puede explicar correctamente.

El sistema de ecuaciones a resolver fue (considerando masa y constante de rigidez unitarias)

$$\frac{dy_1}{dx} = -y_2; \quad \frac{dy_2}{dx} = y_1 \tag{10}$$

donde en nuestro caso y_1 es la velocidad de la partícula e y_2 la posición de la partícula.



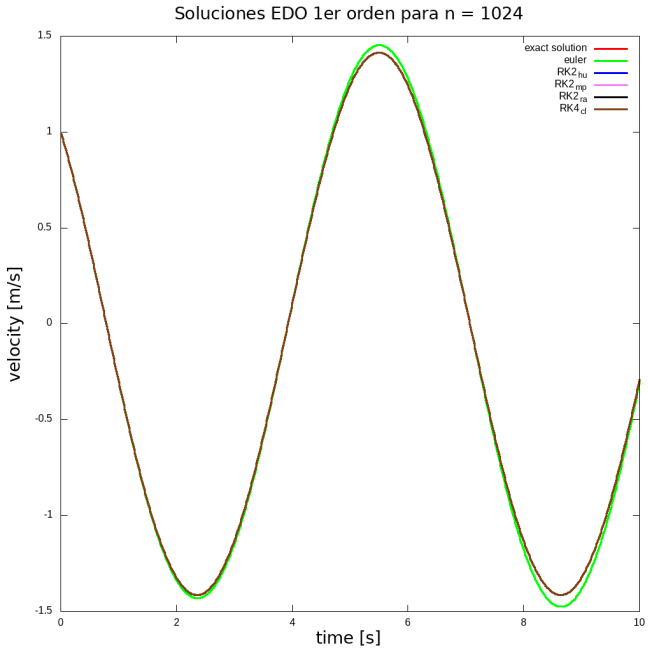


Figure 2: Posición vs tiempo (real time)(link) **Figure 3:** Velocidad vs tiempo (real time)(link)

Aquí se puede notar para una, relativamente, poca cantidad de puntos cómo el método de

Euler es menos eficiente que los otros, pues presenta ciertas diferencias significativas respecto de la solución exacta.

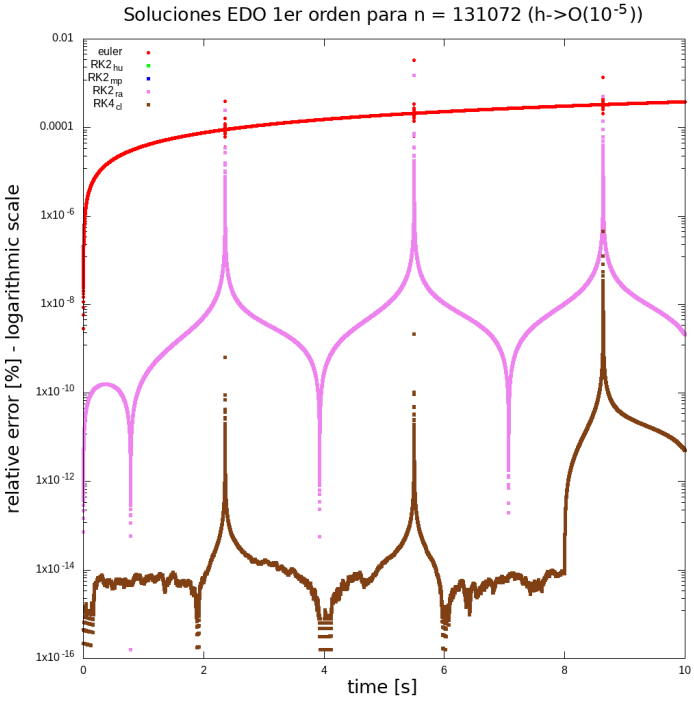
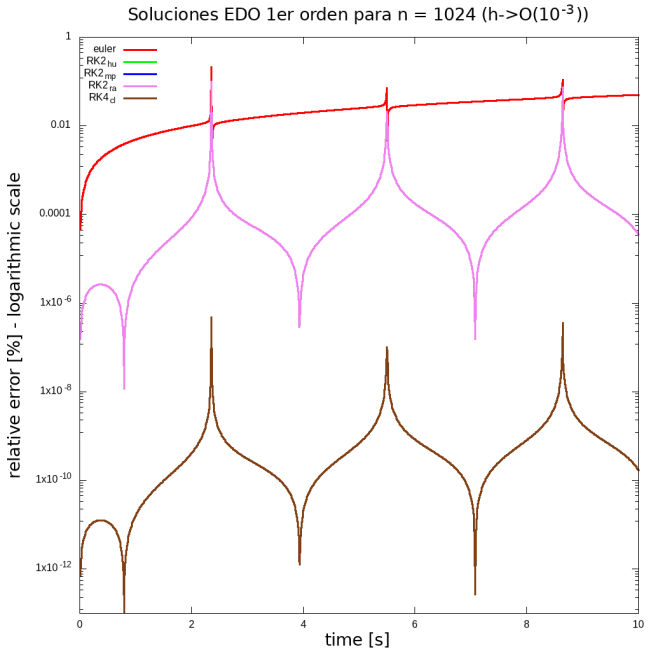


Figure 4: error relativo vs tiempo (real time)(link)



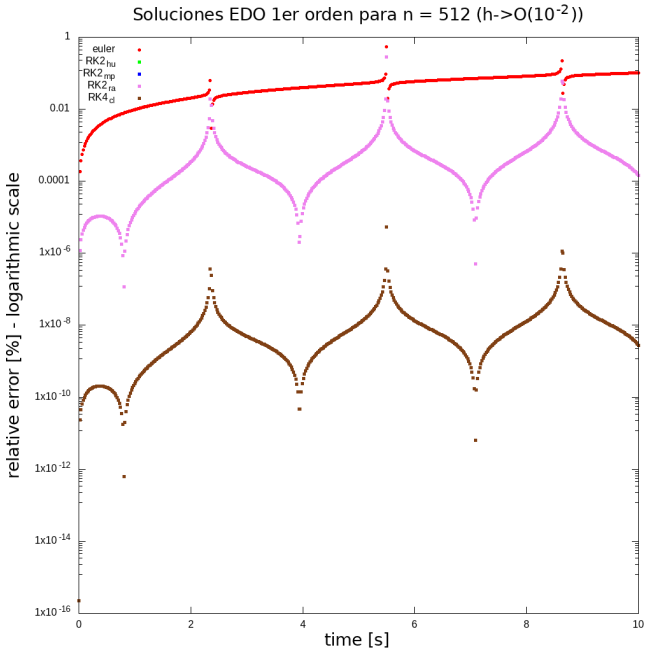


Figure 5: error relativo vs tiempo (real time)(link)**Figure 6:** error relativo vs tiempo (real time)(link)

Aquí vemos que el error relativo presenta picos, los picos hacia arriba muestran una maximización del error que se producen cuando la solución exacta se anula, entonces la diferencia entre la solución exacta y la aproximada y la numérica se hace muy grande. En cambio, los picos hacia abajo muestran una minimización del error que se producen cuando la solución exacta y numérica se igualan y el patrón se repite debido al carácter armónico de las soluciones. Observamos también que el método de RK4 tiene menor error, luego le siguen los métodos de RK2 y finalmente el método de Euler. Es necesario recalcar que, si bien algunos valores del error nos pueden dar muy pequeños estos no pueden ser menores que el error de redondeo de la máquina del orden de $O\left(10^{-15}\right)$.

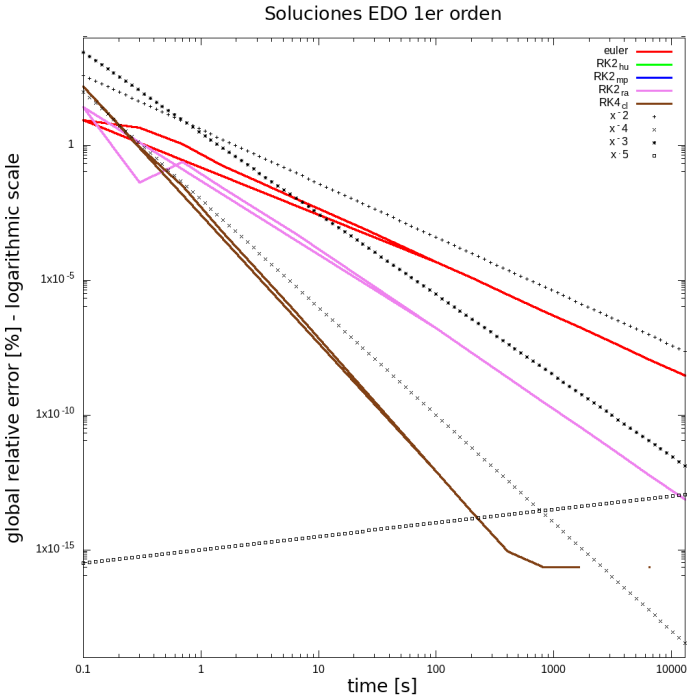


Figure 7: error relativo global vs tiempo (real time)(link)

Aquí vemos que el error relativo (al tiempo $t = t_{final} = 10$) varia según una ley de potencias, que coincide con la ley de potencia de los errores de truncamiento local, es decir, el método de Euler varía según una ley $1/h^2$, los métodos de RK2 varían según una ley $1/h^3$ y el método de RK4 varía según una ley $1/h^4$. Es necesario aclarar que las simulaciones sólo se lograr hacer hasta una cantidad de puntos igual a $n = 2^{17}$ que nos daría un paso de $h \approx O(10^{-5})$, esto es suficiente para corroborar el n_{opt} del método de RK4 que nos daría del orden de $O(10^3)$ pero no así para ver el n_{opt} del método de Euler y RK2, además, se graficó la ley de potencia de $1/\sqrt{n}$ que corresponde a como crecen los errores de redondeo, luego de alcanzada la precisión de la máquina. No se simuló para un n nos permita llegar a un paso del orden de $h \approx O(10^{-6})$ debido a un aumento apreciable en el tiempo de cpu, esto puede deberse a varios factores, los más relevantes podrían ser que se están guardando todos los puntos de la grilla y la otra es al diseño del algoritmo que puede tener varios loops que aumentan el tiempo de computo.

Códigos

Repositorio GitHub

https://github.com/mendzmartin/fiscomp2022.git

Programa principal

https://github.com/mendzmartin/fiscomp2022/blob/main/lab01/prob06/ODE_second_order.f90

Bash script para correr el código

https://github.com/mendzmartin/fiscomp2022/blob/main/lab01/prob06/script_run.sh

Módulos necesarios

https://github.com/mendzmartin/fiscomp2022/blob/main/modules/module_presition.f90

https://github.com/mendzmartin/fiscomp2022/blob/main/modules/module_functions_1D.f90

https://github.com/mendzmartin/fiscomp2022/blob/main/modules/module_functions_2D.f90

https://github.com/mendzmartin/fiscomp2022/blob/main/modules/module_EDO_segundo_orden.f90

https://github.com/mendzmartin/fiscomp2022/blob/main/modules/module_numerical_error.f90

Referencias

Chapman, S. (2007). *Fortran 95/2003 for Scientists & Engineers*. McGraw-Hill Education. https://books.google.com/books/about/Fortran_95_2003_for_Scientists_Engineers.html?hl=&id=c8cLDQEACAAJ

Chapra, S. C., & Canale, R. P. (2007). *Métodos numéricos para ingenieros*. https://books.google.com/books/about/M%C3%A9todos_num%C3%A9ricos_para_ingenieros.html?hl=&id=hoH0MAAACAAJ

Landau, R. H., Mejía, M. J. P., PÃ¡ez, M. J., Kowallik, H., & Jansen, H. (1997). *Computational Physics*. Wiley-VCH. https://books.google.com/books/about/Computational_Physics.html?hl=&id=MJ3vAAAAMAAJ