Primer informe de Física Computacional: Integración numérica y ecuaciones diferenciales

Francisco Fernández

Marzo del 2020

Resumen

En este informe se resolvieron dos problemas de la guía. El primero de ellos se basa en la integración numérica, comparando la regla del trapecio, la de Simpson y la cuadratura de Gauss–Legendre. El segundo se centra en la resolución de ecuaciones diferenciales ordinarias de primer orden usando los métodos de Euler, Runge–Kutta 2 y 4 para un oscilador armónico. En los dos casos se realizó un análisis de los errores de algoritmo y de redondeo.

1. Introducción

1.1. Integración numérica

La integración numérica se basa en la forma tradicional de sumar áreas de cuadriláteros debajo de una curva. Esta definición está estrechamente relacionada con la de Riemann, que relaciona la integral con el límite de la suma de los cuadriláteros cuando el ancho de los intervalos tiende a cero. Teniendo esto en cuenta se puede aproximar la integral de una función $f(x)$ por

$$\int_a^b f(x) \simeq \sum_{i=1}^{N} f(x_i) w_i, \tag{1}$$

esto quiere decir que debemos tener nuestra función evaluada en N puntos dentro del intervalo $[a, b]$ y sumar cada uno de ellos multiplicados por un peso w_i.

1.1.1. Regla del Trapecio

En la regla del trapecio los puntos en x, incluyendo a y b, en los que está evaluada la función están equiespaciados por una distancia h,

$$h = \frac{b-a}{N-1}, \tag{2}$$

si empezamos contando $x_1 = a$, y se quiere llegar a $x_N = b$, entonces el i-ésimo punto se obtiene como

$$x_i = a + (i-1)h. \tag{3}$$

En cada intervalo de integración se construye un trapezoide de base h y altura dada por el punto medio, $(f_i + f_{i+1})/2$, de una linea recta que une $f(x_i)$ con $f(x_{i+1})$. Como los puntos internos al intervalo se cuentan dos veces, se tiene que

$$\int_a^b f(x)dx \simeq \frac{h}{2}f_a + hf_2 + ... + hf_{N-1} + \frac{h}{2}f_b,$$

entonces los pesos w_i son iguales a $h/2$ si estoy en los extremos e iguales a h en los puntos internos.

1.1.2. Regla de Simpson

En la regla de Simpson también se cumple que los puntos están equiespaciados y se incluyen los extremos a y b. Las ecuaciones (2) y (3) siguen siendo validas, lo que cambia ahora es que la función $f(x)$ en el intervalo se aproxima por una parábola, por lo que se necesitan tres puntos. Para hacer esto se utilizan dos intervalos adyacentes por paso, lo cual exige que la cantidad de intervalos sea par o, lo que es lo mismo, que el número de puntos N sea impar. Desarrollando lo explicado arriba se puede obtener que

$$\int_a^b f(x)dx \simeq \frac{h}{3}f_a + \frac{4h}{3}f_2 + \frac{2h}{3}f_3 + \frac{4h}{3}f_4 + ... + \frac{4h}{3}f_{N-1} + \frac{h}{3}f_b,$$

de esta expresión se pueden extraer los distintos pesos w_i.

1.1.3. Cuadratura de Gauss–Legendre

En este método de integración los x_i no están uniformemente distribuidos dentro del intervalo y no se utilizan los extremos a y b. Esta variación de los x_i está acompañada por una variación en los pesos w_i, siempre positivos, y dependen de la función $f(x)$ que se quiera integrar. Los puntos N se eligen de manera tal que el error de la aproximación (1) se anule.

1.2. Ecuaciones diferenciales

Mientras que analíticamente solo pueden resolverse, de manera fácil, ecuaciones diferenciales en regiones en las cuales se cumple alguna aproximación lineal, computacionalmente se pueden resolver en cualquier región. A partir de una ecuación diferencial de primer orden,

$$\frac{dy}{dt} = f(t, y),$$

podemos obtener la dinámica de un problema, es decir, ver como evoluciona en función del tiempo; para ello es necesario dar una condición inicial, $y_0 \equiv y(t = 0)$. Como la única variable independiente, en este caso, es el tiempo t, la ecuación diferencial de primer orden es *ordinaria*.

1.2.1. Regla de Euler

La regla de Euler es el algoritmo más simple para integrar ecuaciones diferenciales. Utiliza el algoritmo de la diferencia finita hacia adelante para obtener la derivada

$$\frac{dy(t)}{dt} \simeq \frac{y(t_{n+1}) - y(t_n)}{h} = f(t, y),$$

esta resta en el numerador puede llevar a un gran error de redondeo. Si definimos $y_n \equiv y(t_n)$, entonces el algoritmo queda

$$y_{n+1} = y_n + hf(t_n, y_n);$$

el error del mismo es $\mathcal{O}(h^2)$.

1.2.2. Algoritmos de Runge–Kutta

Para el desarrollo del algoritmo de Runge–Kutta 2 (RK2) se expande $f(t, y)$ en una serie de Taylor al rededor del punto medio del intervalo de integración y se retienen dos términos. El algoritmo se basa en las siguientes ecuaciones

$$y_{n+1} \simeq y_n + k_2,$$

donde

$$k_2 = hf\left(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right), \qquad k_1 = hf(t_n, y_n),$$

y el error es $\mathcal{O}(h^3)$.

El método de Runge–Kutta de orden cuatro (RK4) baja el error a $\mathcal{O}(h^4)$ al quedarse con el termino h^2 en la expansión de Taylor en el punto medio del intervalo. Este algoritmo necesita evaluar cuatro pendientes intermedias, esto lo hace computacionalmente más caro que Euler, pero con una precisión mucho mejor con menos puntos. Se puede obtener la solución a la ecuación diferencial de la siguiente manera

$$y_{n+1} = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

donde

$$k_1 = hf(t_n, y_n), \qquad k_2 = hf\left(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right),$$

$$k_3 = hf\left(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right), \qquad k_4 = hf(t_n + h, y_n + k_3).$$

2. Resultados y discusiones

2.1. Comparación entre métodos de integración numérica

Para la resolución del problema 4 de la guía se escribió un módulo, `integrales.f90`, que contiene el programa disponible en la página de la materia, `gaussmod.f90`, para calcular integrales utilizando la cuadratura de Gauss–Legendre. Al mismo se le agregó la regla de Simpson, la del Trapecio y la subrutina que calcula la función $f(x) = \exp(-x)$. También se utilizó el módulo `precision.f90` para poder cambiar rápidamente de doble a simple precisión. Luego se escribió el `main.f90`, dónde se llama dentro de un loop a cada una de las subrutinas que devuelve el valor de la integral numérica con dichos métodos, teniendo cuidado de tomar solo número de puntos impares donde la regla de Simpson es válida; a estos valores se los compara con la integral exacta entre 0 y 1,

$$\int_0^1 \exp(-x)dx = 1 - e^{-1}.$$

En un archivo de datos se guardaron las cantidades N y ε (error relativo definido como $|numérico - exacto|/exacto$) y se graficaron luego utilizando `gnuplot`. Todos estos archivos están adjuntos con el informe.

En la figura (1) se observa el gráfico obtenido en escala logarítmica para ambos ejes del error relativo en función del número de puntos de integración N para los tres métodos desarrollados. Podemos observar que con la cuadratura de Gauss–Legendre se puede obtener un error relativo de 10^{-14} en tan solo 10 pasos de integración y que luego se empieza a observar contribuciones del error de redondeo de la maquina. En el caso de la regla de Simpson, tenemos que con 1000

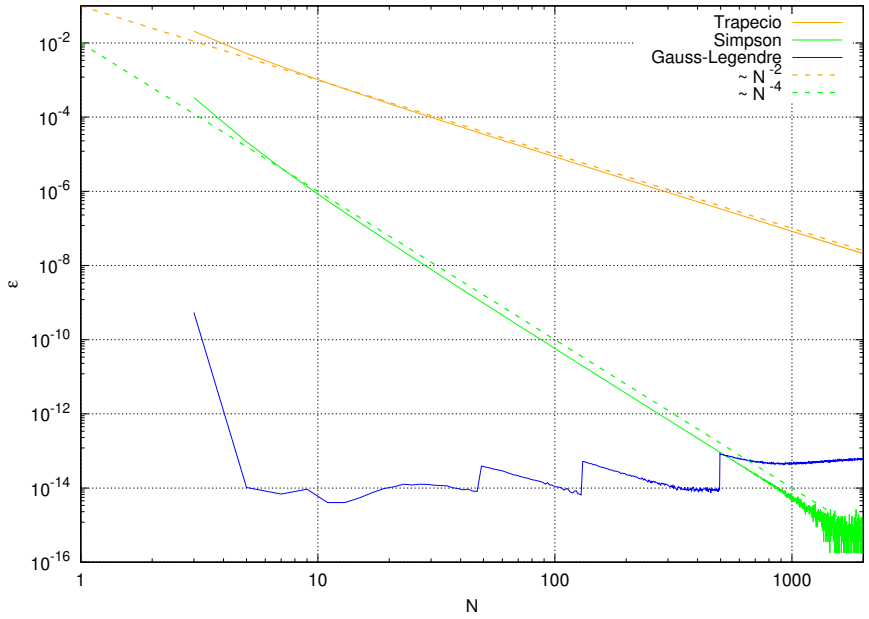

Figura 1: Error relativo en función de N para cálculos en doble precisión

pasos recién se llega a la misma precisión que con Gauss–Legendre, por encima de ese valor de N se tiene que empieza a influir el error de redondeo. Por último, para la regla del Trapecio vemos que en 2000 pasos de integración el error de algoritmo todavía no alcanzó su mínimo, sino que se llegó a una precisión cercana a 10^{-8}. Podemos observar que para N mayor a 100 tanto la regla de Simpson como la del Trapecio se comportan como $\epsilon \approx CN^{\alpha}$; esto se corresponde con las líneas rectas de punto para α igual a -2 y -4.

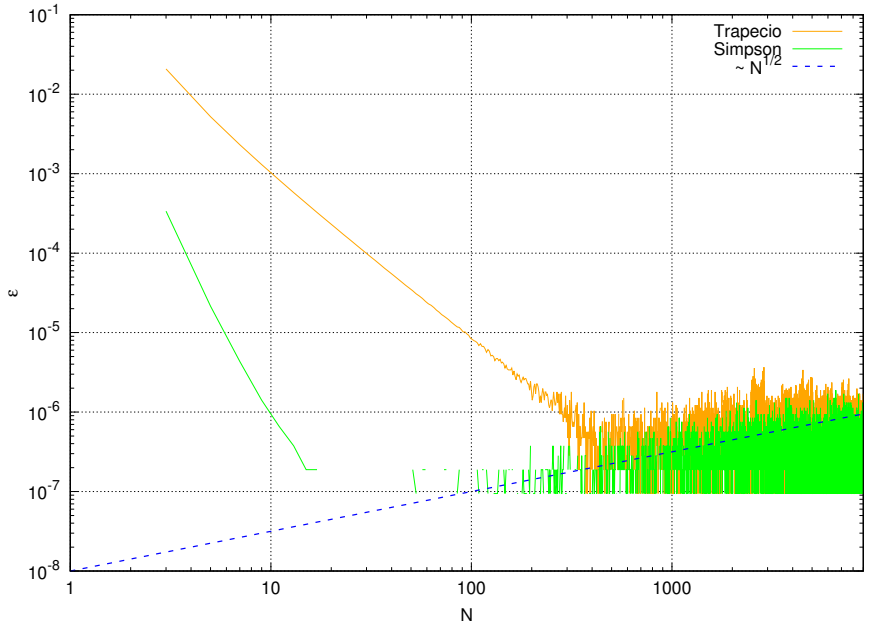

Figura 2: Error relativo en función de N para cálculos en simple precisión

Ahora, si quisiera observar en el mismo gráfico el error de redondeo para la regla de Trapecio tendría que irme a valores de N muy grandes ($\approx 10^6$), en los que la precisión del cálculo llegue a 10^{-14}. Para observar esto se modificó ligeramente el código para que compute en simple precisión y obtenga la figura (2). La línea de puntos que se observa es la correspondiente a $N^{1/2}$ y es la que está relacionada con el error de redondeo. Es curioso que para valores de N

entre 10 y 100 el error relativo para la regla de Simpson sea igual a cero, por lo cual no se observan puntos en la gráfica, esto se debe a que, en la precisión en la que estoy trabajando, el truncamiento del valor exacto coincide con el que devuelve la subrutina.

Para concluir este problema puede decirse que la regla de Simpson es más precisa que la del trapecio y que no hace falta tomar valores de N mayores a 1000, alcanza con menos para tener una buena aproximación cercana al error de la maquina ($\varepsilon_m \approx 10^{-16}$, en mi caso). Por otro lado, si decidiera usar Gauss–Legendre, en este caso con 10 puntos de integración alcanza, por lo menos para esta función.

2.2. Oscilador armónico

La ecuación de un oscilador armónico es de segundo orden,

$$\ddot{x} = -\frac{k}{m}x, \tag{4}$$

donde k es la constante del resorte y m la masa unida al mismo, ambos parámetros son igual a 1 en el problema que se resuelve para t entre 0 y 10, usando la condición inicial $x(0) = \dot{x}(0) = 1$. La ecuación (4) la podemos reescribir como un sistema de dos ecuaciones, si tomamos $v = \dot{x}$, de la siguiente forma,

$$\dot{x} = v,$$

$$\dot{v} = -kx.$$

Para resolver este sistema de ecuaciones diferenciales de primer orden se escribió un módulo, `odes.f90`, que calcula un paso de integración de Euler, RK2 y RK4; que, en principio, está vectorizado para sistemas de ecuaciones más grandes, pero se necesita especificar la función del problema.

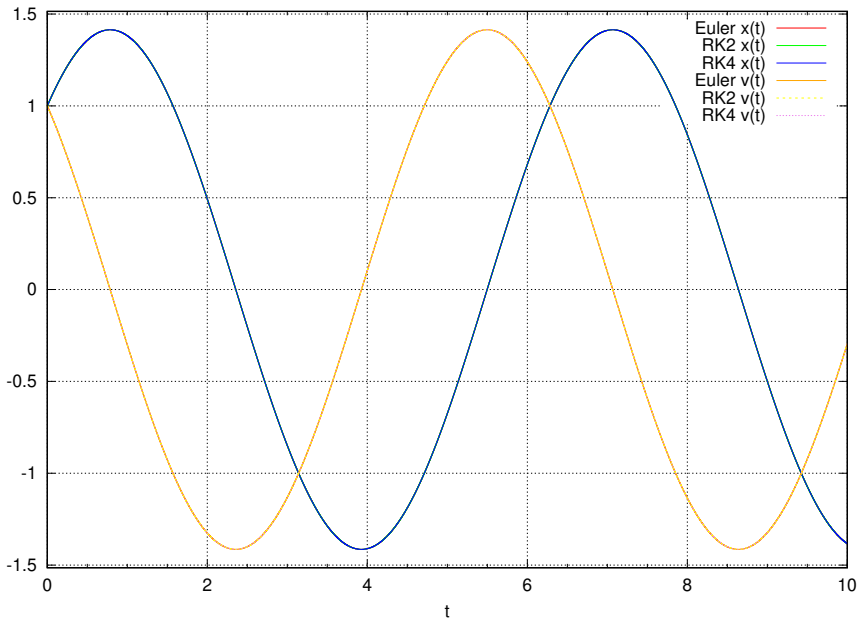

Figura 3: Solución de la posición y la velocidad en función del tiempo para los distintos métodos.

Para obtener el ancho de paso óptimo h_{opt} se realizó, previamente, un análisis del error total, $\varepsilon_{TOTAL} = \varepsilon_{algoritmo} + \varepsilon_{redondeo}$. Teniendo el orden del error de cada método y que el error de redondeo va como $\sqrt{N}\varepsilon_m$, entonces derivando ε_{TOTAL} respecto a h e igualando a cero se puede despejar h_{opt} para cada algoritmo; 10^{-6}, 10^{-5} y 10^{-3} para Euler, RK2 y RK4, respectivamente.

Para obtener el error relativo en cada paso temporal se aprovecha que la solución de (4) es conocida y, para dichas condiciones iniciales, se tiene

$$x(t) = \sqrt{2}\cos\left(t - \frac{\pi}{4}\right),$$

$$v(t) = -\sqrt{2}\,\mathrm{sen}\left(t - \frac{\pi}{4}\right).$$

Todo esto se encuentra dentro de `main_ab.f90`.

En la figura (3) podemos observar las curvas de la posición y la velocidad en función del tiempo, obtenidas a partir de cada método, superpuestas. A simple vista no se ve diferencia entre un método y otro, por lo cual se analiza el error local de la posición $|x(t) - x_{exacto}(t)|$ en función de t. Observando la figura (4) puede decirse que se va acumulando error a lo largo del tiempo y que por la forma funcional de la solución hay puntos singulares en los que el error disminuye abruptamente, coincidiendo para los métodos de RK2 y RK4, desplazandose para la regla de Euler. También se ve que entre la precisión de Euler y RK4 hay siete ordenes de magnitud y que el primero, al tener un h_{opt} tres ordenes de magnitud más pequeño, demora mucho más tiempo de computo, ya que tiene que hacer más pasos para llegar al mismo $t = 10$.

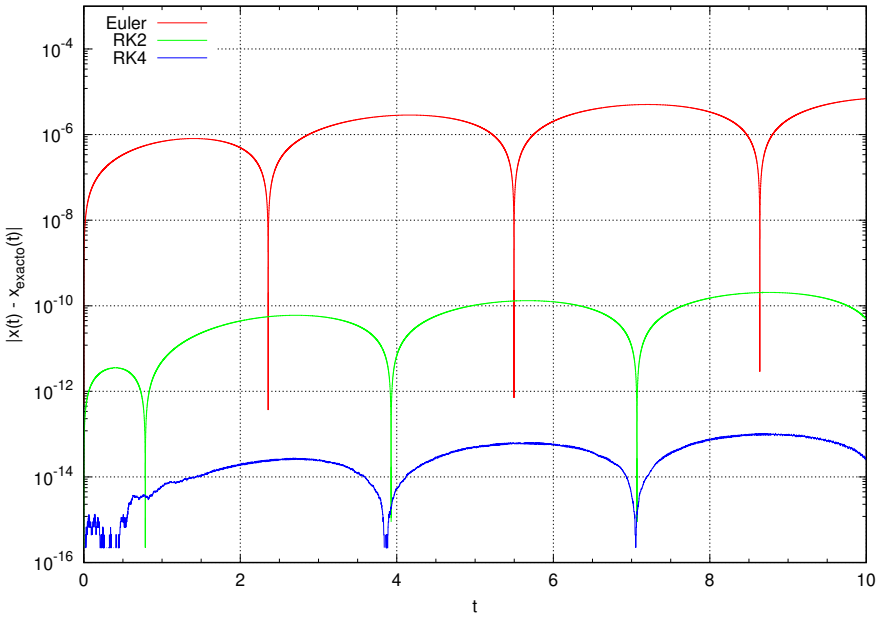

Figura 4: Error relativo en función de t, donde h es igual al respectivo h_{opt} del método.

Dentro del mismo `main_ab.f90` se calcula la energía total del sistema E, que tiene una contribución de la energía cinética y otra de la energía potencial, y es igual a

$$E(t) = \frac{1}{2}(x(t)^2 + v(t)^2);$$

el gráfico puede observarse en la figura (5). Como el problema que se está analizando es conservativo, entonces la energía no debería variar en función del tiempo, sin embargo, en el caso de Euler, vemos que al final del intervalo de tiempo analizado se aprecia un aumento de 10^{-5}. Vale la pena aclarar que RK2 y RK4, ambos, conservan la energía y sus curvas se ven superpuestas.

Por último, puede definirse el error global entre $[0, t_f]$ como

$$\varepsilon_g = \frac{|x_{num}(t_f) - x_{exacto}(t_f)|}{x_{exacto}(t_f)},$$

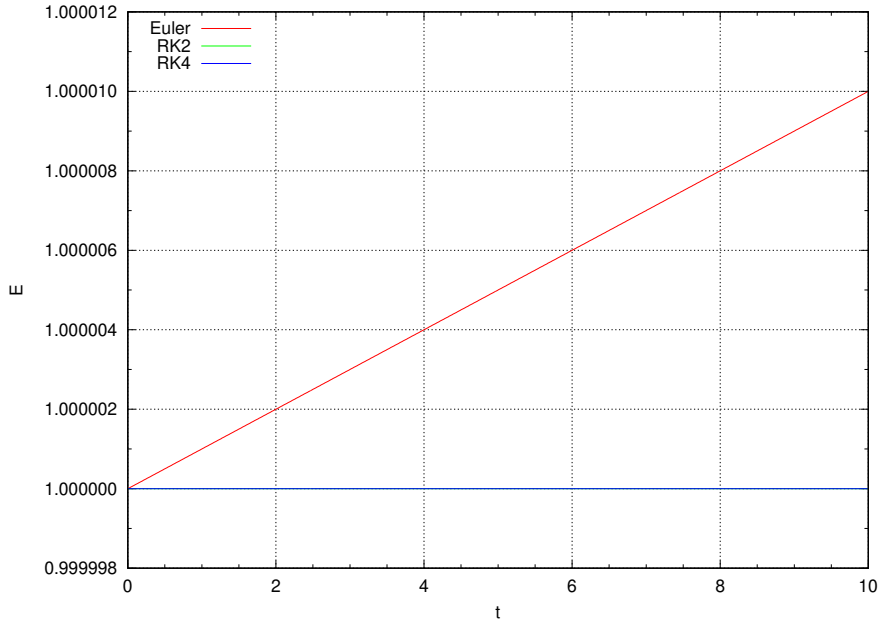

Figura 5: Energía en función del tiempo para los distintos métodos.

el mismo puede puede analizarse en función de la longitud del paso. Esta es otra forma de encontrar el h_{opt} y está desarrollada en el archivo `main_c.f90`, que devuelve los datos de h desordenados, por lo cual se escribió un script en Python, `sort.py`, que los ordena, también cambiando el orden del resto de las columnas. En la figura (6) podemos ver la ley de potencia a la que decae ε_g en función de $1/h$ para los tres métodos. Y, análogo a lo que sucedido con la integración numéricas, no se alcanza a ver el error de redondeo para Euler y RK2, por lo cual se presenta el mismo gráfico pero en simple precisión en la figura (7), donde puede apreciarse como cuando crece $1/h$ empieza a aparecer el error de redondeo, que va como $\sim \sqrt{1/h}\varepsilon_m$, donde ahora el error de la maquina es $\sim 10^{-7}$. Al igual que en el caso de la regla de Simpson, hay algunos de h para los cuales el valor numérico de $x(10)$ es igual al exacto.

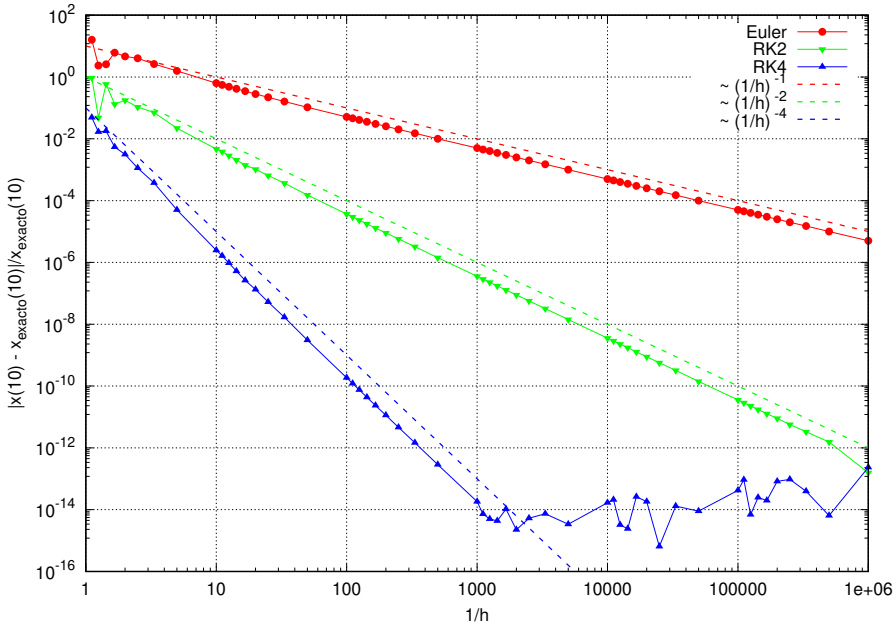

Figura 6: Error global en función de $1/h$.

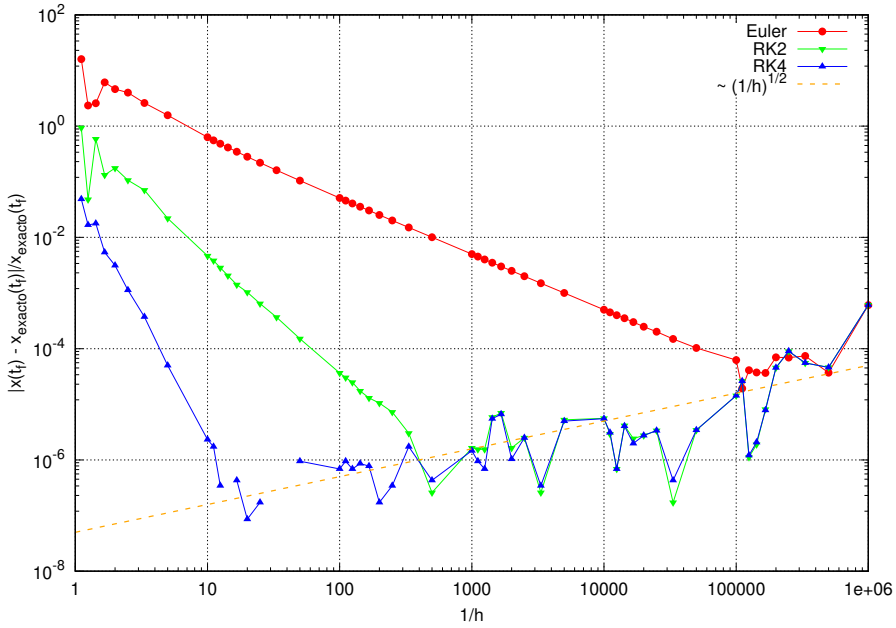

Figura 7: Error global en función de $1/h$.

Como conclusión general del problema podemos decir que para este caso RK4 es el método más preciso y que requiere menos tiempo total de computo, ya que al tener un ancho de paso considerablemente mayor que en los otros métodos se realizan menos operaciones (si bien el método intrinsecamente realiza más).