Introducción

$$I_{exacta} = \int_a^b f(x)dx \tag{1}$$

Los métodos de integración Trapecio, Simpson 1/3 y Simpson 3/8 son parte de una familia de métodos de integracion llamadas fórmulas cerradas (porque se conocen los límites de integración) de Newton-Cotes que, básicamente, se basan en reemplazar la función a integrar $f(x)$ por otras funciones (polinomios) que son más sencillas de integrar.

Método del Trapecio

Aquí se aproxima la función original a integrar por un polinomio de grado uno (una recta) entonces, la fórmula del Trapecio de aplicación múltiple es:

$$I \approx \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{(n-1)} f(x_i) + f(x_n)\right];\ h = \frac{(b-a)}{n} \tag{2}$$

o visto de forma geométrica $\text{Area} = (\text{ancho}) \cdot (\text{altura promedio})$ donde

$$\text{ancho} \equiv (b-a);\ (\text{altura promedio}) \equiv \frac{1}{2n}\left[f(x_0) + 2\sum_{i=1}^{(n-1)} f(x_i) + f(x_n)\right] \tag{3}$$

El error de truncamiento global es la suma de los errores de truncamiento locales (para cada iteración, un único segmento),

$$\epsilon_{trunc,loc}^{trap.} = -\frac{(b-a)^3}{12}\max_{a\leqslant\xi\leqslant b}\left[f^{(2)}(\xi)\right] = -\frac{h^3}{12}\max_{a\leqslant\xi\leqslant b}\left[f^{(2)}(\xi)\right] \rightarrow \text{para una sola iteración} \tag{4}$$

$$\Rightarrow \epsilon_{trunc,glob}^{trap.} = -\frac{(b-a)^3}{12n^3}\sum_{i=1}^{n}\max_{x_i\leqslant\xi_i\leqslant x_{i+1}}\left[f^{(2)}(\xi_i)\right] \tag{5}$$

si aproximamos el valor promedio de la segunda derivada en todo el intervalo como

$$f_{prom.}^{(2)} = \frac{1}{n}\sum_{i=1}^{n}\max_{x_i\leqslant\xi_i\leqslant x_{i+1}} f^{(2)}(\xi_i) \tag{6}$$

$$\Rightarrow \epsilon_{trunc,glob}^{trap.} = -\frac{(b-a)^3}{12n^2} f_{prom.}^{(2)} \Rightarrow \epsilon_{trunc,glob}^{trap.} \approx O\left(n^{-2}\right) \tag{7}$$

Así, si se duplica el número de segmentos, el error de truncamiento se divide entre cuatro. Si se requiere de alta exactitud, la regla del trapecio de múltiples segmentos exige un gran trabajo computacional. Aunque este trabajo resulta insignificante para una sola aplicación, puede ser muy importante cuando: a) se evalúan numerosas integrales, o b) donde la función en sí, es compleja, y consume tiempo apreciable su evaluación.

Métodos de Simpson

Simpson 1/3

Aquí se aproxima la función original a integrar por un polinomio de segundo grado que pasa por tres puntos.

$$I \approx \frac{h}{3}\left[f(x_0)+4\sum_{i=1,3,5...}^{(n-1)} f(x_i)+2\sum_{i=2,4,6...}^{(n-2)} f(x_i)+f(x_n)\right];\ h=\frac{(b-a)}{n} \tag{8}$$

o visto de forma geométrica Area $=$ (ancho) $\cdot$ (altura promedio) donde

$$\text{ancho} \equiv (b-a);\ \text{(altura promedio)} \equiv \frac{1}{3n}[f(x_0)+4\sum_{i=1,3,5...}^{(n-1)} f(x_i)+2\sum_{i=2,4,6...}^{(n-2)} f(x_i)+f(x_n] \tag{9}$$

El error de truncamiento global es la suma de los errores de truncamiento locales (para cada iteración, dos segmentos),

$$\epsilon_{trunc,loc}^{simp.1/3} = -\frac{(b-a)^5}{2880}\max_{a\leqslant\xi\leqslant b} f^{(4)}(\xi) = -\frac{h^5}{90}\max_{a\leqslant\xi\leqslant b} f^{(4)}(\xi) \rightarrow \text{para una sola iteración} \tag{10}$$

$$\Rightarrow \epsilon_{trunc,glob}^{simp.1/3} = -\frac{(b-a)^5}{180n^4} f_{prom.}^{(4)}$$

Notando que la regla de Simpson 1/3 es más exacta que la regla del trapecio. Y además, en lugar de ser proporcional a la tercera derivada, el error es proporcional a la cuarta derivada, en consecuencia, Simpson 1/3 alcanza una precisión de tercer orden aun cuando se base en sólo tres puntos (da resultados exactos para polinomios cúbicos aun cuando el resultado se obtenga de una parábola).

Una limitación muy fuerte de este método es que se debe utilizar un número par de segmentos (numero impar de puntos) para implementar el método.

Simpson 3/8

Aquí se aproxima la función original a integrar por un polinomio de Lagrange de tercer grado que pasa por cuatro puntos.

$$I \approx \frac{h}{3}\left[f(x_0)+4\sum_{i=1,3,5...}^{(n-1)} f(x_i)+2\sum_{i=2,4,6...}^{(n-2)} f(x_i)+f(x_n)\right];\ h=\frac{(b-a)}{n} \tag{11}$$

o visto de forma geométrica Area $=$ (ancho) $\cdot$ (altura promedio) donde

$$\text{ancho} \equiv (b-a);\ \text{(altura promedio)} \equiv \frac{1}{3n}[f(x_0)+4\sum_{i=1,3,5...}^{(n-1)} f(x_i)+2\sum_{i=2,4,6...}^{(n-2)} f(x_i)+f(x_n] \tag{12}$$

El error de truncamiento global es la suma de los errores de truncamiento locales (para cada iteración, tres segmentos),

$$\epsilon_{trunc,loc}^{simp.1/3} = -\frac{(b-a)^5}{6480}\max_{a\leqslant\xi\leqslant b} f^{(4)}(\xi) = -\frac{3h^5}{80}\max_{a\leqslant\xi\leqslant b} f^{(4)}(\xi) \rightarrow \text{para una sola iteración} \tag{13}$$

$$\Rightarrow \epsilon_{trunc,glob}^{simp.1/3} = -\frac{(b-a)^5}{80n^4} f_{prom.}^{(4)} \tag{14}$$

Notando que la regla de Simpson 3/8 es más exacta que la relga de Simpson 1/3. Sin embargo, es preferible utilizar Simpson 1/3 porque alcanza una exactitud de tercer orden utilizando tres puntos (cuatro segmentos) en lugar de Simpson 3/8 que porque alcanza una exactitud de tercer orden utilizando cuatro puntos (cinco segmentos).

Una limitación muy fuerte de este método es que se debe utilizar un número impar de segmentos (numero par de puntos) para implementar el método. Por ello, en general se utiliza una

combinación de estos dos métodos de simpson, para integrar en un intervalo $[a, b]$ uno para integrar sobre un sub-intervalo con segmentos pares (Simpson 1/3) y otro para integrar sobre un sub-intervalo con segmentos impares (Simpson 3/8).

Método de cuadratura Gauss-Legendre

Este método esta específicamente diseñado para calcular integrales de funciones que conocemos explícitamente y pueden escribirse como funciones gaussianas.

La esencia de este método podemos entenderla recordando que el método del trapecio necesita los puntos extremos para evaluar la función a integrar y calcular el área debajo de la recta de aproximación, claramente, hay zonas donde este método tiene un gran error. Entonces, si suponemos ahora que eliminamos la restricción de los puntos fijos tendremos libertad de evaluar el área bajo una línea recta de aproximación que une dos puntos cualesquiera (no necesariamente los puntos extremos) y al ubicar esos puntos en forma inteligente, definiríamos una línea recta que equilibrara los errores negativo y positivo reduciendo el error de truncamiento local y, en consecuencia, el error de truncamiento global.

$$I \approx \sum_{i=1}^{n} f(x_i) w_i \tag{15}$$

donde se llevo a cabo un mapeo uniforme de la variable $y_i \in [-1, 1]$ y la variable original $x_i \in [a, b]$ para poder utilizar el módulo estándar de generación de los puntos y pesos, obteniendo la siguiente relación

$$x_i = \frac{(b+a)}{2} + \frac{(b-a)}{2} y_i;\ \omega_i = \frac{(b-a)}{2} (\omega_i)' \Rightarrow \int_a^b f(x)dx = \frac{(b-a)}{2} \int_{-1}^{1} f[x(y)]dy \tag{16}$$

El error en las fórmulas de Gauss-Legendre

$$\epsilon_{trunc,glob}^{simp.1/3} = -\frac{2^{(2n+3)}[(n+1)!]^4}{(2n+3)[(2n+2)!]^3} \max_{-1 \leqslant \xi \leqslant 1} f^{(2n+2)}(\xi)$$

donde n es el número de puntos menos uno y $f^{(2n+2)}(\xi)$ es la la $(2n+1)$-ésima derivada de la función, después del cambio de variable con ξ localizada en algún lugar en el intervalo desde –1 hasta 1. El método de cuadratura de Gauss-Legendre es más preciso que las fórmulas de Newton Cotes siempre que las derivadas de orden superior no aumenten sustancialmente cuando se incremente n (muy preciso para funciones gaussianas).

Como la cuadratura de Gauss requiere evaluaciones de la función en puntos irregularmente espaciados dentro del intervalo de integración, no es apropiada para los casos donde la función no se conoce. Es decir, para problemas que tratan con datos tabulados, será necesario interpolar para obtener la evaluación de la función en lugares específicos. Sin embargo, cuando se conoce la función, su eficiencia es de una ventaja muy superior al resto de los métodos, en particular cuando se deben realizar muchas evaluaciones de la integral.

Por otro lado, como el método de cuadratura de Gauss no requiere la evaluación de la función en los límites del intervalo es muy útil para funciones que presentan singularidades, simplemente se debe calcular la integral en sub-intervalos que excluyen dichas singularidades. Está técnica no es posible con los otros métodos de integración.

Resultados y Discusiones

Teniendo en cuenta que el método de Simpson 1/3 es malo para un número de intervalos pares y Simpson 3/8 es malo para un número de intervalos impares se realizaron corridas del algoritmo parametrizadas en número de intervalos todos pares o todos impares (múltiplos de tres)

obteniendo los resultados que se mostrarán en esta sección.

Notemos que para el caso de intervalos al corroborar el parámetro de chequeo nos dieron buenos resultados (prácticamente cero).

```
mendez@mendez-notebook:~/path_directory$ ./script_run.sh
 Ingrese el limite inferior de integración (a) y presione Enter.
0
 Ingrese el limite superior de integración (a) y presione Enter.
1
 ---------------------------------------
 Input/Output file. istat =            0
 ---------------------------------------
 check_value must be equal to zero if integration was ok
 check_value of simpson 1/3 =        0.00E+00
 check_value of simpson 1/3 =        0.00E+00
 check_value of simpson 1/3 =        0.00E+00
 check_value of simpson 1/3 =        0.00E+00
 check_value of simpson 1/3 =        0.00E+00
 check_value of simpson 1/3 =        0.00E+00
 check_value of simpson 1/3 =        0.00E+00
 check_value of simpson 1/3 =        0.00E+00
 check_value of simpson 1/3 =        0.00E+00
 check_value of simpson 1/3 =        0.00E+00
 check_value of simpson 1/3 =        0.00E+00
 check_value of simpson 1/3 =        0.00E+00
 check_value of simpson 1/3 =        0.00E+00
 check_value of simpson 1/3 =        0.00E+00
 check_value of simpson 1/3 =        0.00E+00
 check_value of simpson 1/3 =        0.00E+00
```

Sin embargo, para intervalos pares el algoritmo no es bueno y lo corroboramos con el parámetro de chequeo obteniendo diferencias no nulas bien significativas, de hasta 0,1. Estos erres se van reduciendo a medida que aumenta el número de intervalos pero el algoritmo nunca es eficiente al requerir una cantidad de puntos pares (intervalos impares) para su aplicación.

```
mendez@mendez-notebook:~/path_directoy$ ./script_run.sh
 Ingrese el limite inferior de integración (a) y presione Enter.
0
 Ingrese el limite superior de integración (a) y presione Enter.
1
 ---------------------------------------
 Input/Output file. istat =            0
 ---------------------------------------
 check_value must be equal to zero if integration was ok
 check_value of simpson 1/3 =       -0.11E+00
 check_value of simpson 1/3 =       -0.37E-01
 check_value of simpson 1/3 =       -0.12E-01
 check_value of simpson 1/3 =       -0.41E-02
 check_value of simpson 1/3 =       -0.14E-02
 check_value of simpson 1/3 =       -0.46E-03
 check_value of simpson 1/3 =       -0.15E-03
 check_value of simpson 1/3 =       -0.51E-04
 check_value of simpson 1/3 =       -0.17E-04
 check_value of simpson 1/3 =       -0.56E-05
```

En curvas punteadas se graficaron funciones que son potencias del número de intervalos.

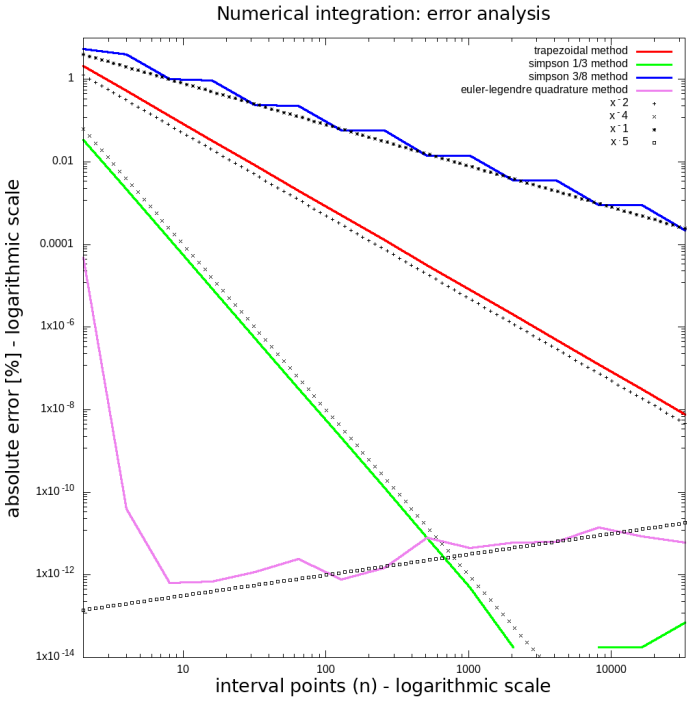


Figure 1: Resultados para número de intervalos pares $n = 2^k$ (link)

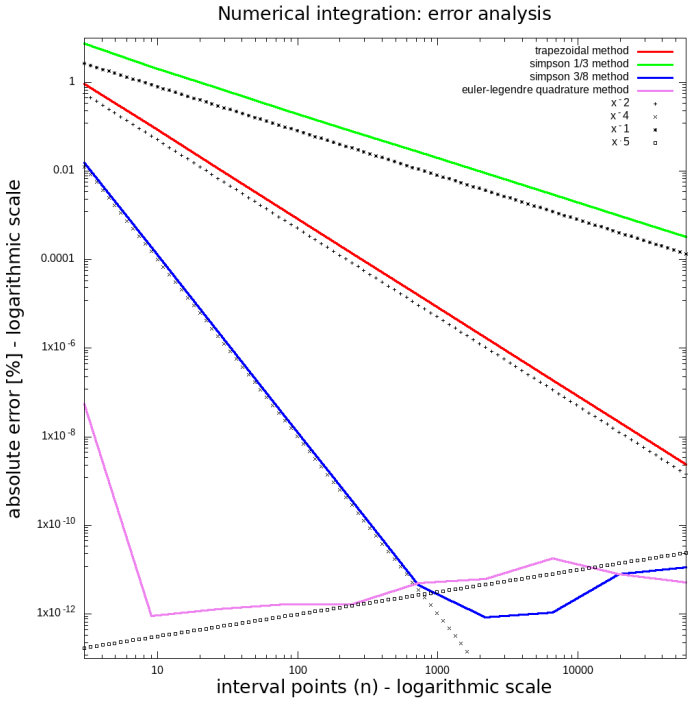


Figure 2: Resultados para número de intervalos impares (múltiplos de tres) $n = 3^k$ (link)

Teniendo en cuenta que podemos estimar en qué orden de magnitud debe rondar el número de intervalos para que los métodos del trapecio y simpson alcancen la precisión de la máquina

$$n_{opt}^{trap} \approx \frac{1}{(\epsilon_m)^{2/5}}; \; n_{opt}^{simp} \approx \frac{1}{(\epsilon_m)^{2/9}}$$

entonces usando una función intrínseca de Fortran podemos conocer el error de la máquina (que es sensible al tipo de precisión que usa el algoritmo) obteniendo los siguientes resultados (ver programa en el siguiente link test_epsilon.f90).

```
mendez@mendez-notebook:~/path_directory$ gfortran -o test_epsilon.o test_epsilon.f90 ../modules/module_presition.f90
&& ./test_epsilon.o
epsilon_machine_sp =  0.119E-06
epsilon_machine_dp =  0.222E-15
```

Con estos resultados obtenemos, bajo la hipótesis de que el algoritmo usa únicamente doble precisión (lo cual es cierto en su mayoría), los valores

$$n_{opt}^{trap} \approx O(10^6);\ n_{opt}^{simp} \approx O(10^3)$$

y esto se puede corroborar con las gráficas obtenidas notando que los métodos de simpson (1/3 para n par y 3/8 para n impar) alcanzan la precisión de la máquina para valores del orden de 10^3 y el método del trapecio debería alcanzar la precisión de la máquina para valores del orden de 10^6, lo cual se podría corroborar o bien extrapolando la curva del trapecio o bien simulando para esta cantidad de intervalos (lo cual requiere mucho tiempo de computo!).

Además, observamos que el error relativo, para todos los métodos, disminuye conforme n se incrementa. Sin embargo, note también que para grandes valores de n, el error empieza a aumentar conforme los errores de redondeo empiezan a dominar (esto sólo se ve en los métodos de cuadratura de Gauss y Simpson porque alcanzan relativamente rápido la precisión de la máquina). Y comparando los métodos vemos que para la regla del trapecio y la regla de Simpson se requiere un número muy grande de evaluaciones de la función (y, por lo tanto, de más trabajo de cálculo) para alcanzar altos niveles de precisión, comparados con el método de la cuadratura de Gauss. Sin embargo, como el método de cuadratura de Gauss tiene la fuerte restricción de que es necesario conocer la función a integrar en la mayoría de los casos reales se podría aplicar sin problemas algoritmos que mezclen la regla se Simpson 1/3 con 3/8 (para abarcar intervalos pares e impares) e incluso se podría diseñar un método adaptativo (donde el n o el h varíe en función de las zonas de complejidad de la función que podemos estimar a partir de datos experimentales).

Códigos

Repositorio GitHub

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

Programa principal

https://github.com/mendzmartin/fiscomp2022/blob/notebook/lab01/prob04/num_integ_gauss.f90

Bash script para correr el código

https://github.com/mendzmartin/fiscomp2022/blob/notebook/lab01/prob04/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_numerical_error.f90
https://github.com/mendzmartin/fiscomp2022/blob/notebook/modules/module_num_integrals.f90
https://github.com/mendzmartin/fiscomp2022/blob/notebook/modules/module_gauss.f90

Otros códigos desarrollados

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

Wikipedia. (2021). *Regla de Simpson — Wikipedia, La enciclopedia libre*.

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

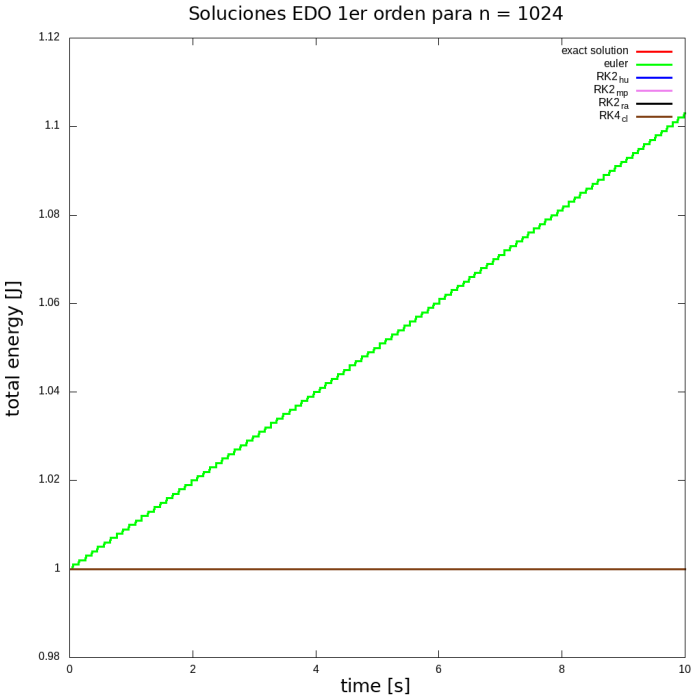


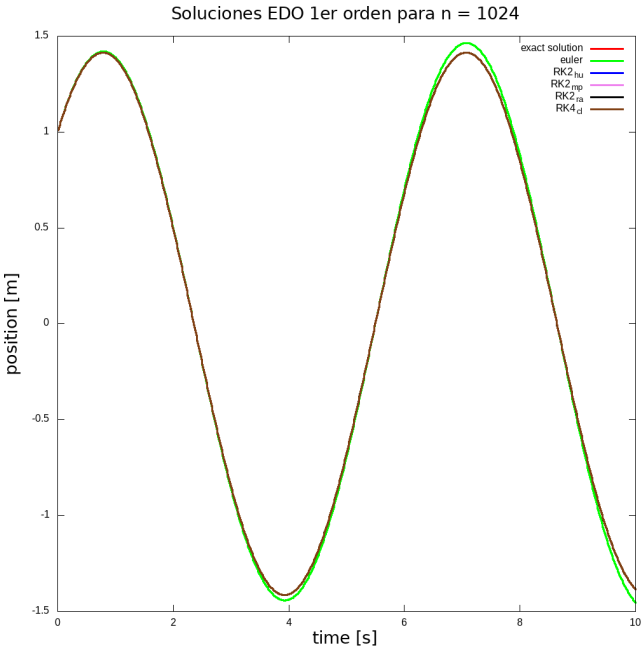
Figure 1: Energía total $= (E_{cin} + E_{pot})$ vs tiempo (real time)(link)

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.



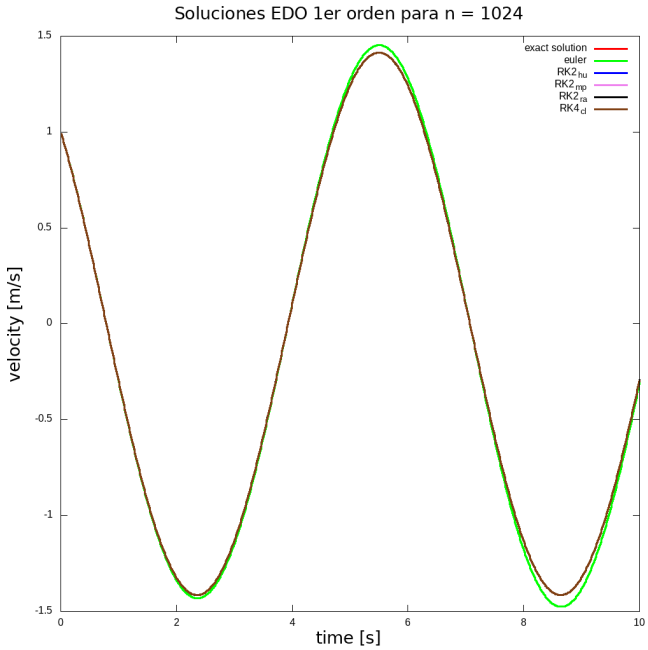


Figure 2: Posición vs tiempo (real time)(link) **Figure 3:** Velocidad vs tiempo (real time)(link)

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.

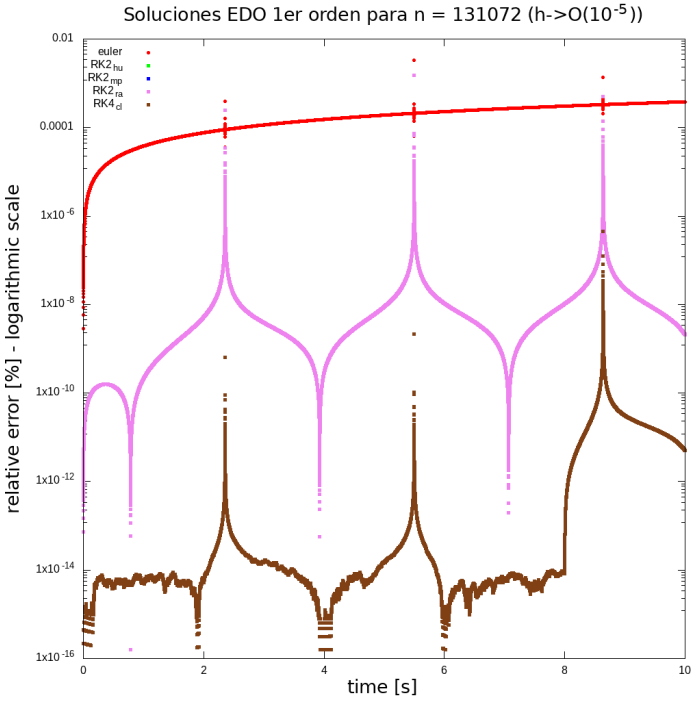
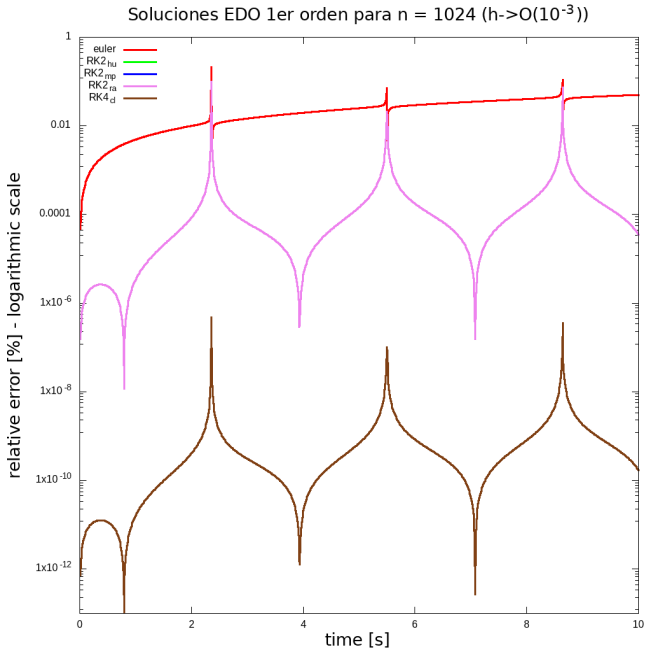


Figure 4: error relativo vs tiempo (real time)(link)



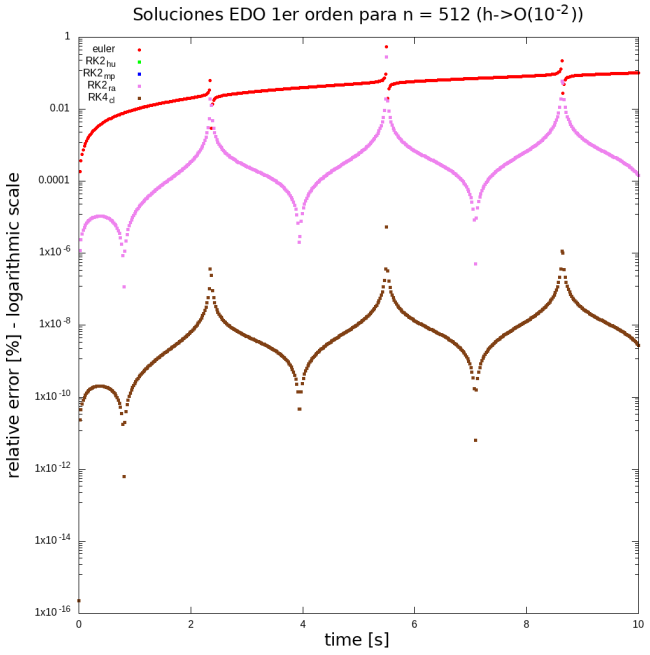


Figure 5: error relativo vs tiempo (real time)(link)**Figure 6:** error relativo vs tiempo (real time)(link)

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 $O\left(10^{-15}\right)$.

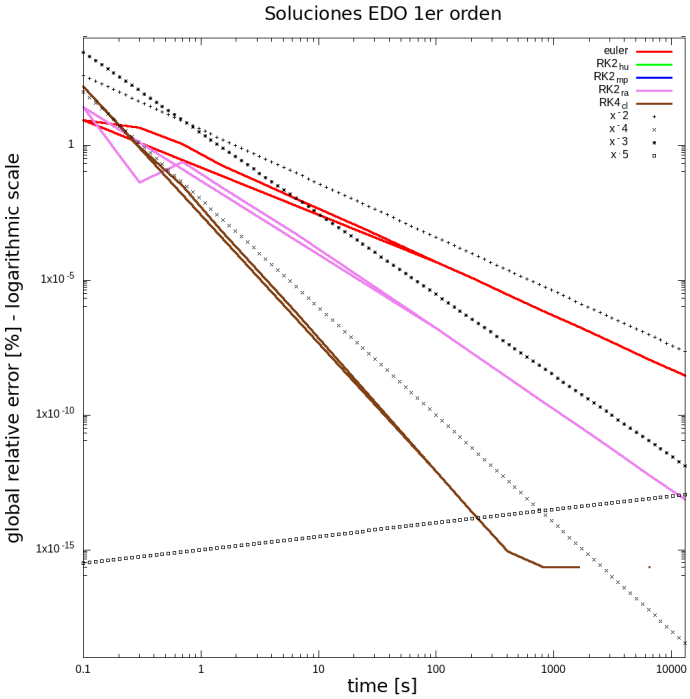


Figure 7: error relativo global vs tiempo (real time)(link)

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