Informe Laboratorio 5 Computación Científica

Josue Peña Atencio, Jeffrey García Gallego

Abril 2020

Introducción

Las funciones para realizar las aproximaciones y comparaciones con las soluciones analíticas fueron definidas manualmente mediante el manejador de funciones de Matlab (no se hizo uso de las funciones para resolver ODEs en Matlab)

En las tablas de cálculo de tiempo, este es tomado mediante la herramienta de stopwatch de Matlab (tic toc), y para cada método se mide solamente el tiempo de ejecución estrictamente necesario para computar la cantidad de puntos especificada.

Método de aproximación	**Función implementada en MATLAB**
Euler	`euler.m`
Runge Kutta 2	`RK2.m`
Runge Kutta 4	`RK4.m`
Multi-paso orden 2	`twoStep.m`
Multi-paso orden 4	`adamsBash.m`
Diferencias finitas	`finDiff.m`
Elementos finitos	`finElm.m`

La aglomeración de los métodos, el cálculo de tiempo, errores y demás, está dividido en dos archivos de main diferentes, `main_1.m` y `main_2.m`. El primero contiene lo relacionado a los métodos de Euler, RK2 y RK4, mientras que el segundo el resto de métodos.

ECM: Error cuadrático medio
Std: Standard deviation/desviación estándar. Esta se calcula a partir de todos los errores absolutos entre los puntos analíticos y aproximados para cada método.

Punto 1

El tiempo de observación en los métodos del primer punto es desde $t_0 = 0$ hasta $t_n = 30$. La cantidad de puntos procesada en los tres casos para el valor h se ajustan de tal forma que caigan en este rango.

ODEs usadas

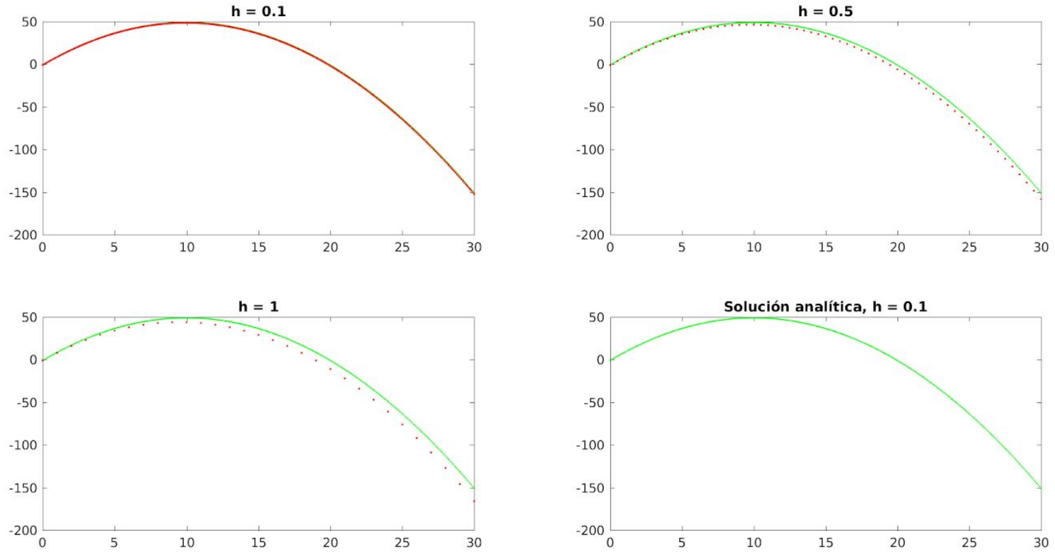
$$y_1' = 10 - t$$
$$y_1(0) = -1$$
$$y_1 = 10t - \frac{t^2}{2} - 1$$

$$y_2' = 9t^2 - 4t$$
$$y_2(0) = 1$$
$$y_2 = 3t^3 - 2t^2 + 1$$

Método de Euler

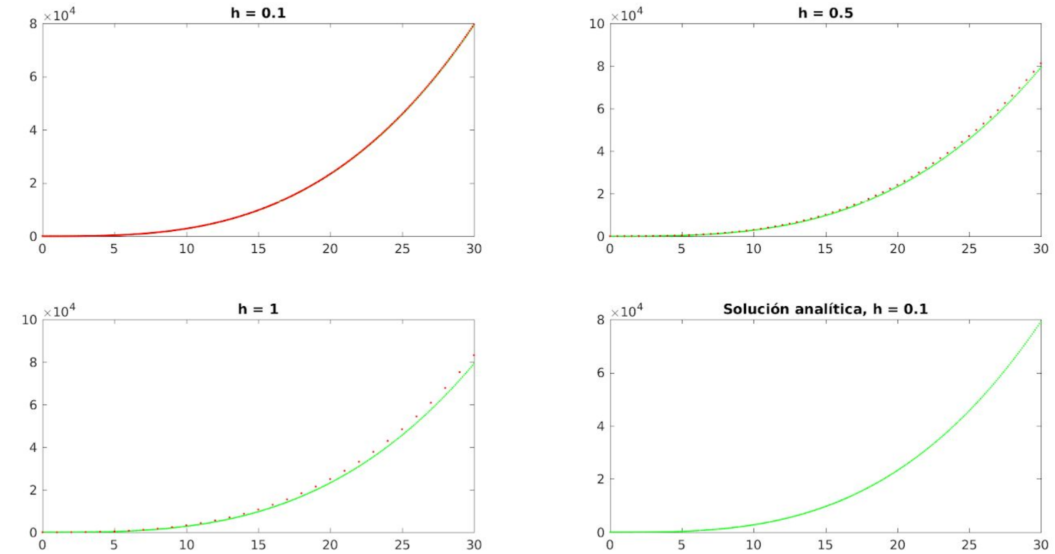
Gráficas

Función 1



Función 2

Tiempo



Función y1		
Función Matlab	**Tiempo (s)**	**Parámetro**
`euler.m`	0.00452900000000000	$h = 0.1$
`euler.m`	0.00182900000000000	$h = 0.5$

`euler.m`	0.00170400000000000	$h = 1$

Función y2		
Función Matlab	**Tiempo (s)**	**Parámetro**
`euler.m`	0.00458000000000000	$h = 0.1$
`euler.m`	0.00208700000000000	$h = 0.5$
`euler.m`	0.00863900000000000	$h = 1$

Error

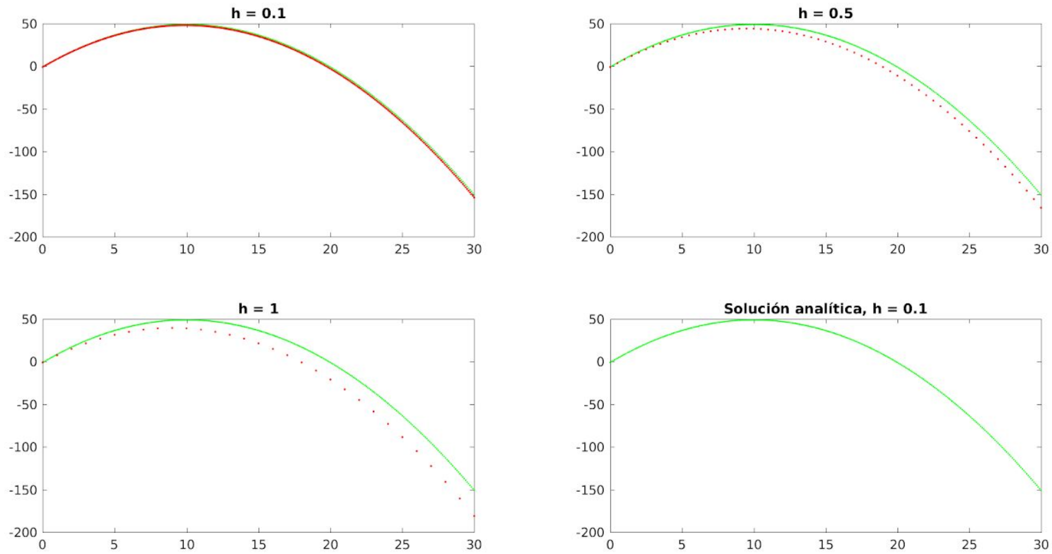
Función y1			
Función Matlab	**ECM**	**Std**	**Parámetro**
`euler.m`	0.754572259136207	0.433245225943548	$h = 0.1$
`euler.m`	18.9226434426229	2.19450965121815	$h = 0.5$
`euler.m`	76.2822580645161	4.49432138717102	$h = 1$

Función y2			
Función Matlab	**ECM**	**Std**	**Parámetro**
`euler.m`	31851.7522322583	119.814332411860	$h = 0.1$
`euler.m`	821482.860143443	614.056751446973	$h = 0.5$
`euler.m`	3413801.03225806	1265.76889633326	$h = 1$

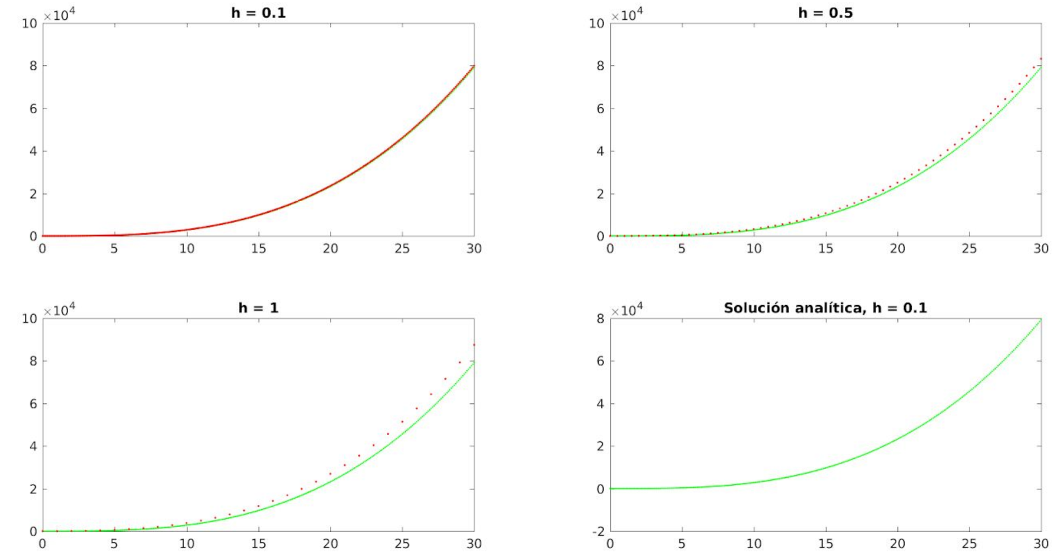
Método RK2

Gráficas

Función 1



Función 2



Tiempo

Función y1		
Función Matlab	**Tiempo (s)**	**Parámetro**
`RK2.m`	0.00379400000000000	$h = 0.1$
`RK2.m`	0.00179500000000000	$h = 0.5$
`RK2.m`	0.00256400000000000	$h = 1$

Función y2		
Función Matlab	**Tiempo (s)**	**Parámetro**
`RK2.m`	0.0123240000000000	$h = 0.1$
`RK2.m`	0.0192360000000000	$h = 0.5$
`RK2.m`	0.00855500000000000	$h = 1$

Error

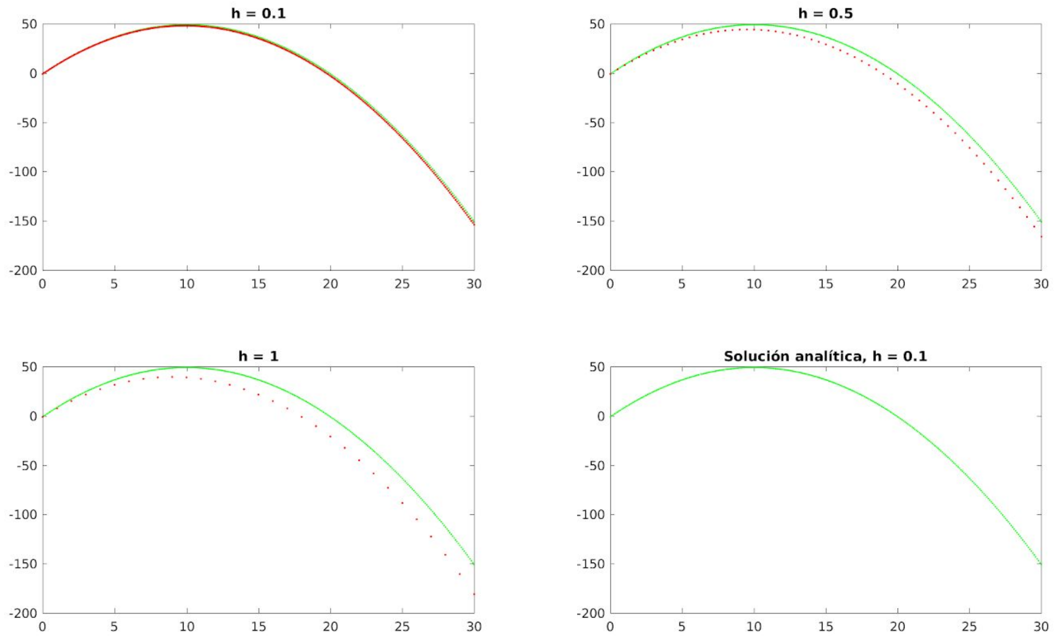
Función y1			
Función Matlab	**ECM**	**Std**	**Parámetro**
`RK2.m`	3.00832225913620	0.866509622452559	$h = 0.1$
`RK2.m`	75.6413934426229	4.41183749806015	$h = 0.5$
`RK2.m`	305.032258064516	9.03874574989156	$h = 1$

Función y2			
Función Matlab	**ECM**	**Std**	**Parámetro**
`RK2.m`	128306.299024763	240.264092165222	$h = 0.1$
`RK2.m`	3401381.58670594	1244.22766669759	$h = 0.5$
`RK2.m`	14609410.2822581	2597.51188619508	$h = 1$

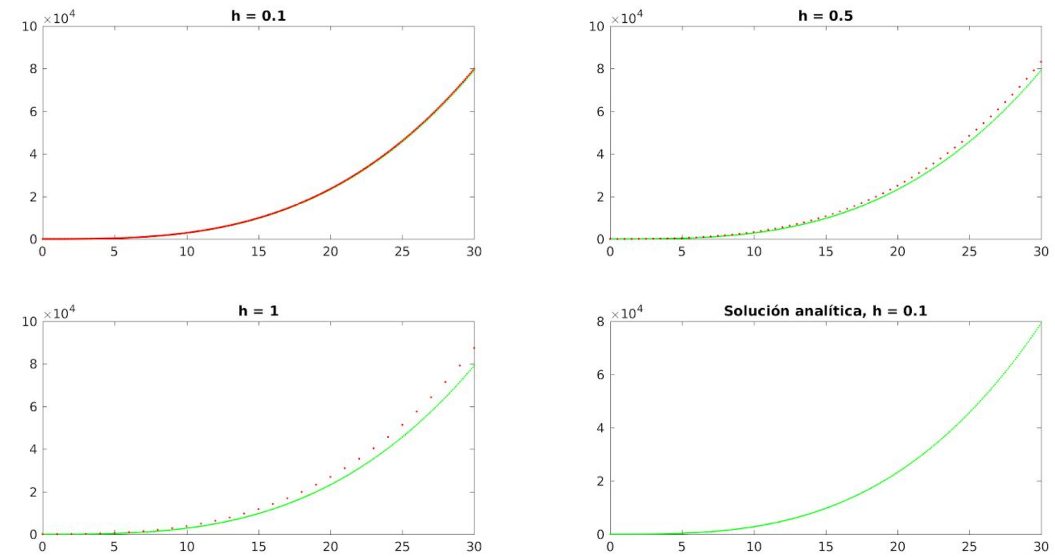
Método RK4

Gráficas

Función 1



Función 2



Tiempo

Función y1		
Función Matlab	**Tiempo (s)**	**Parámetro**
`RK4.m`	0.00398300000000000	$h = 0.1$
`RK4.m`	0.00254400000000000	$h = 0.5$

`RK4.m`	0.00212400000000000	$h = 1$

Función y2		
Función Matlab	**Tiempo (s)**	**Parámetro**
`RK4.m`	0.00504100000000000	$h = 0.1$
`RK4.m`	0.00265900000000000	$h = 0.5$
`RK4.m`	0.00182400000000000	$h = 1$

Error

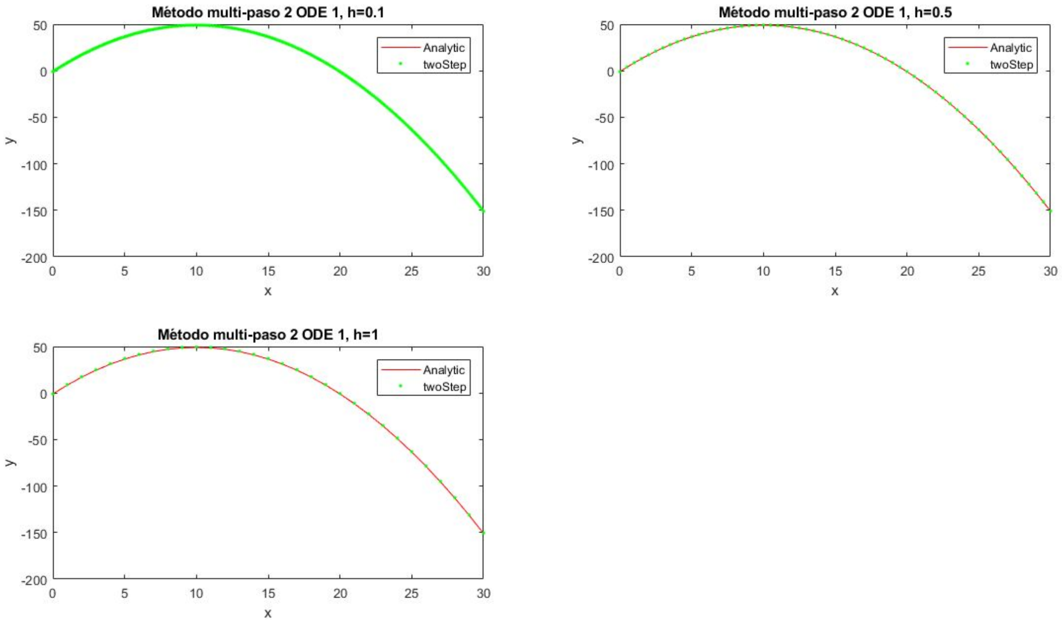
Función y1			
Función Matlab	**ECM**	**Std**	**Parámetro**
`RK4.m`	3.00832225913622	0.866509622452564	$h = 0.1$
`RK4.m`	75.6413934426229	4.41183749806015	$h = 0.5$
`RK4.m`	305.032258064516	9.03874574989156	$h = 1$

Función y2			
Función Matlab	**ECM**	**Std**	**Parámetro**
`RK4.m`	128126.168562265	240.137736178451	$h = 0.1$
`RK4.m`	3378121.39139344	1241.00731839506	$h = 0.5$
`RK4.m`	14415824.0322581	2584.31489933497	$h = 1$

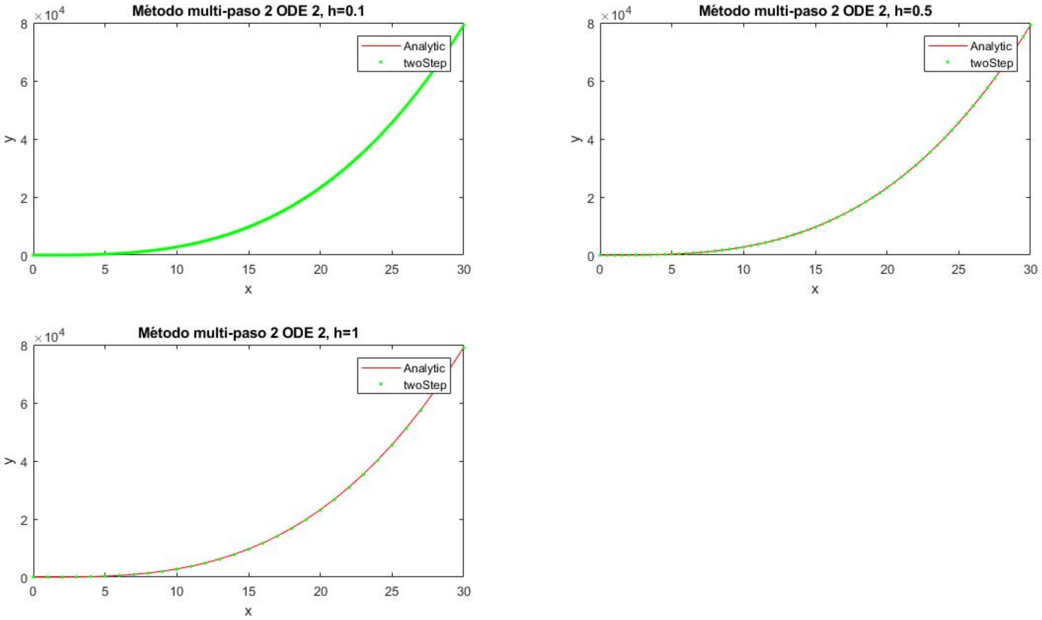
Método multi-paso orden 2

Gráficas

Función 1



Función 2



Tiempo

Función y1		
Función Matlab	**Tiempo (s)**	**Parámetro**

`twoStep.m`	0,00181050000000000	$h = 0.1$
`twoStep.m`	0,000390700000000000	$h = 0.5$
`twoStep.m`	0,000215600000000000	$h = 1$

Función y2		
Función Matlab	**Tiempo (s)**	**Parámetro**
`twoStep.m`	0,00277500000000000	$h = 0.1$
`twoStep.m`	0,000283500000000000	$h = 0.5$
`twoStep.m`	0,000229300000000000	$h = 1$

Error

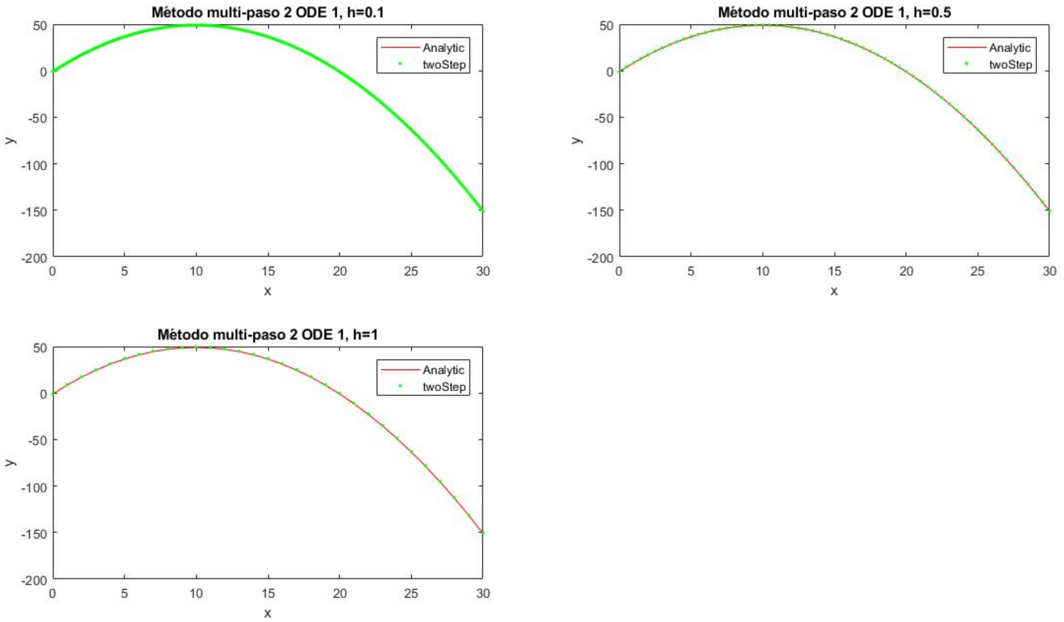
Función y1			
Función Matlab	**ECM**	**Std**	**Parámetro**
`twoStep.m`	2,49169435215506e-05	0,000288195208851863	$h = 0.1$
`twoStep.m`	0,0153688524590164	0,0160046099916120	$h = 0.5$
`twoStep.m`	0,241935483870968	0,0898026510133875	$h = 1$

Función y2			
Función Matlab	**ECM**	**Std**	**Parámetro**
`twoStep.m`	1,63578575579164	0,652305995884344	$h = 0.1$
`twoStep.m`	1004,82133709016	16,6071876901156	$h = 0.5$
`twoStep.m`	15734,6370967742	67,8425596747780	$h = 1$

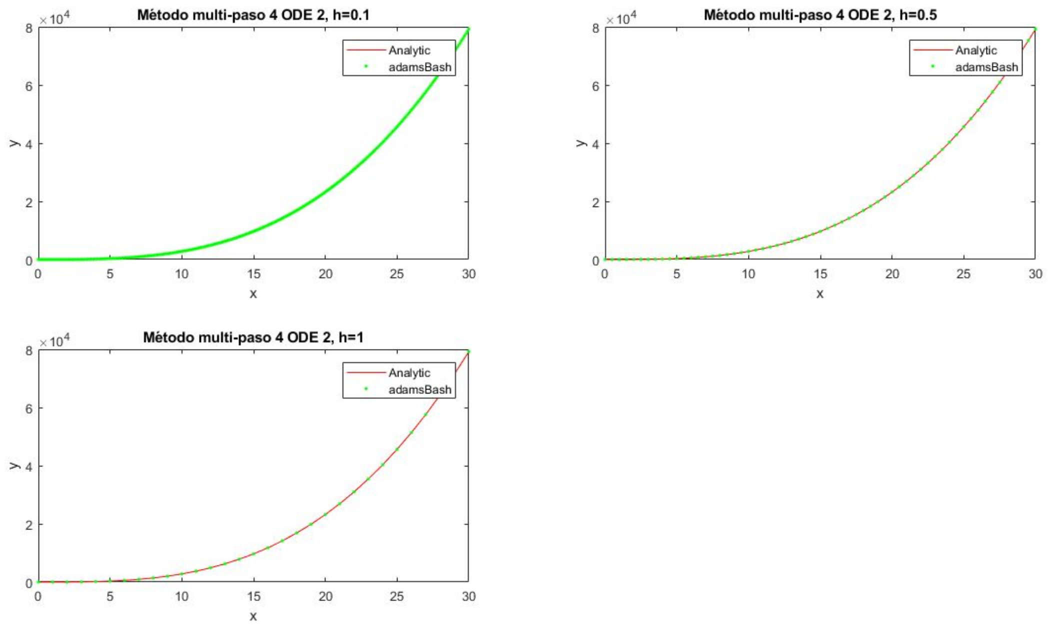
Método multi-paso orden 4

Gráficas

Función 1



Función 2



Tiempo

Función y1		
Función Matlab	**Tiempo (s)**	**Parámetro**

`adamsBash.m`	0,000647100000000000	$h = 0.1$
`adamsBash.m`	0,000120300000000000	$h = 0.5$
`adamsBash.m`	0,000109500000000000	$h = 1$

Función y2		
Función Matlab	**Tiempo (s)**	**Parámetro**
`adamsBash.m`	0,00232030000000000	$h = 0.1$
`adamsBash.m`	0,000408500000000000	$h = 0.5$
`adamsBash.m`	0,000232200000000000	$h = 1$

Error

Función y1			
Función Matlab	**ECM**	**Std**	**Parámetro**
`adamsBash.m`	2,49169435214765e-05	0,000288195208851434	$h = 0.1$
`adamsBash.m`	0,0153688524590164	0,0160046099916120	$h = 0.5$
`adamsBash.m`	0,241935483870968	0,0898026510133875	$h = 1$

Función y2			
Función Matlab	**ECM**	**Std**	**Parámetro**
`adamsBash.m`	9,06104651565350e-05	0,000698776814564637	$h = 0.1$
`adamsBash.m`	0,638767930327869	0,135149747446944	$h = 0.5$
`adamsBash.m`	67,6209677419355	2,00201511386129	$h = 1$

Punto 3

ODEs usadas

$y_1'' = 6t$
$y_1(0) = 0$
$y_1(1) = 1$

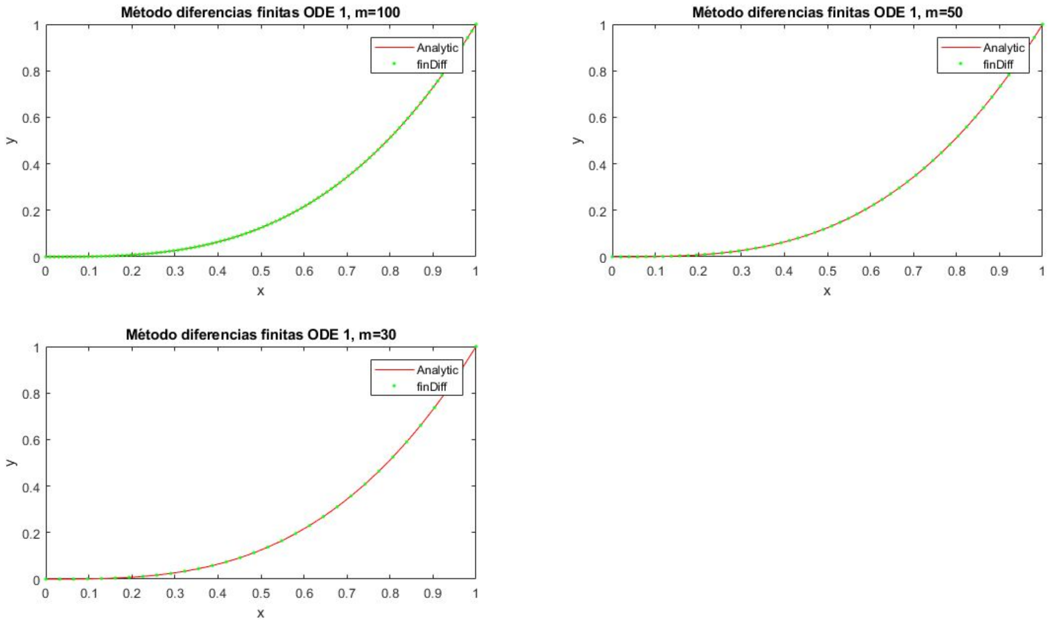
$y_1 \ = t^3$

$y_2' \ = sin(t)$
$y_2(0) \ = \ 1$
$y_2(1) \ = \ 1.5708$
$y_2 \ =- sin(t) + 1.6366t$

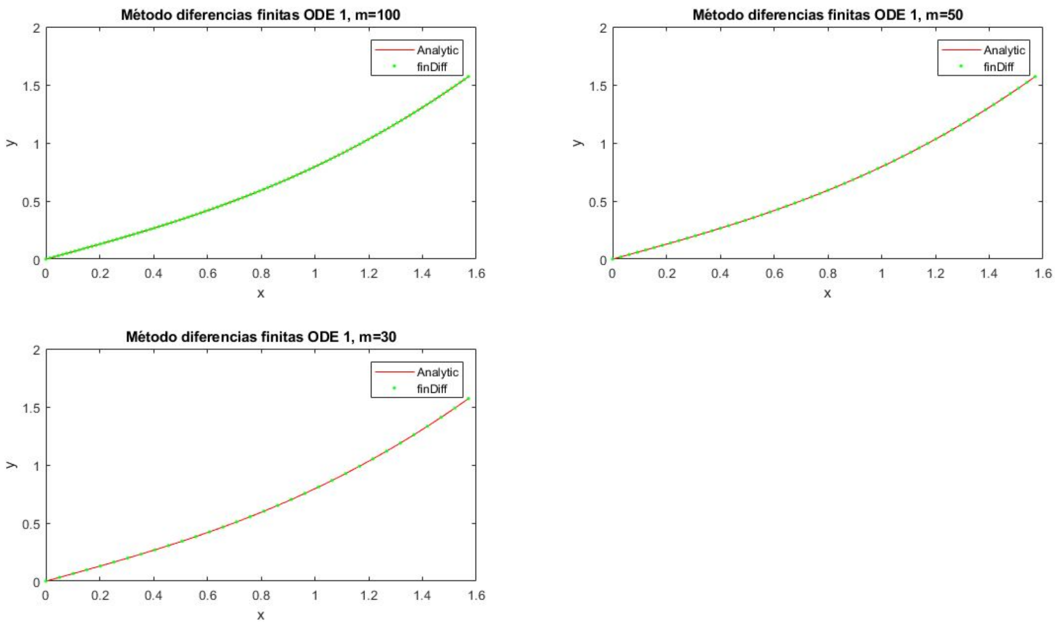
Método de diferencias finitas

Gráficas

Función 1



Función 2



Tiempo

Función y1		
Función Matlab	**Tiempo (s)**	**Parámetro**
`finDiff.m`	0,0117123000000000	$m = 100$
`finDiff.m`	0,00354400000000000	$m = 50$
`finDiff.m`	0,000900900000000000	$m = 30$

Función y2		
Función Matlab	**Tiempo (s)**	**Parámetro**
`finDiff.m`	0,0225214000000000	$m = 100$
`finDiff.m`	0,00341380000000000	$m = 50$
`finDiff.m`	0,00162540000000000	$m = 30$

Error

Función y1			
Función Matlab	**ECM**	**Std**	**Parámetro**
`finDiff.m`	2,06301492122519e-30	7,61573515894429e-16	$m = 100$
`finDiff.m`	5,41654869958939e-31	3,83051369139991e-16	$m = 50$
`finDiff.m`	2,23040070372165e-31	2,68233709222404e-16	$m = 30$

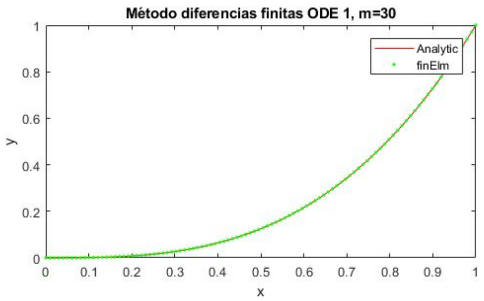
Función y2			
Función Matlab	**ECM**	**Std**	**Parámetro**
`finDiff.m`	3,13498726273611e-10	1,00086930935128e-05	$m = 100$

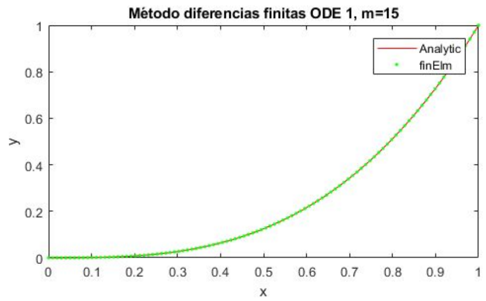
`finDiff.m`	1,58221307993600e-10	9,93913907630707e-06	$m = 50$
`finDiff.m`	3,83549344030635e-10	9,18939690450672e-06	$m = 30$

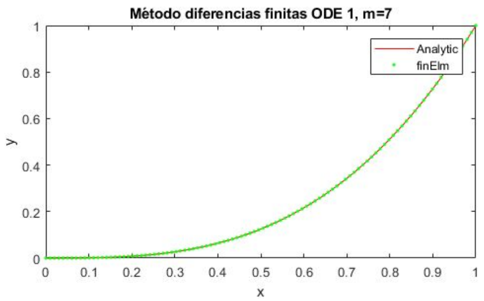
Método de elementos finitos

Gráficas

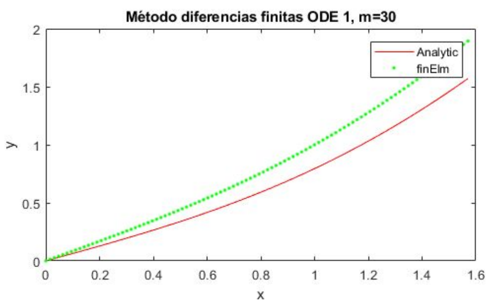
Función 1

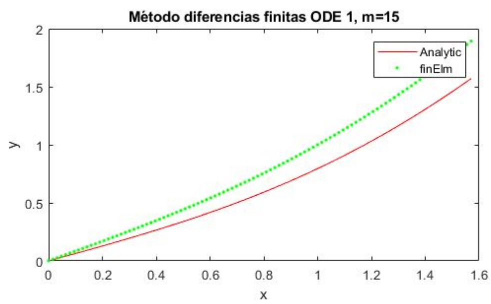


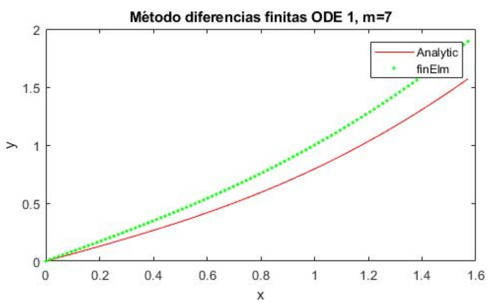




Función 2







Tiempo

Función y1		
Función Matlab	**Tiempo (s)**	**Parámetro**
`finElm.m`	0,00241970000000000	$m = 30$
`finElm.m`	0,00105440000000000	$m = 15$
`finElm.m`	0,000388300000000000	$m = 7$

Función y2		
Función Matlab	**Tiempo (s)**	**Parámetro**
`finElm.m`	0,00249290000000000	$m = 30$
`finElm.m`	0,000648000000000000	$m = 15$
`finElm.m`	0,000587800000000000	$m = 7$

Error

Función y1			
Función Matlab	**ECM**	**Std**	**Parámetro**
`finElm.m`	1,20137909577857e-20	7,40827856434979e-11	$m = 30$
`finElm.m`	7,58134141167354e-34	2,62531808074085e-17	$m = 15$
`finElm.m`	7,58134141167354e-34	2,62531808074085e-17	$m = 7$

Función y2			
Función Matlab	**ECM**	**Std**	**Parámetro**
`finElm.m`	0,0347360346384430	0,0944076091754488	$m = 30$
`finElm.m`	0,0346958242964242	0,0943061810746834	$m = 15$
`finElm.m`	0,0346958241988642	0,0943061729191162	$m = 7$

Análisis y conclusiones

En los métodos de aproximación para ODEs de primer grado, se pudo corroborar que el crecimiento del error cuadrático medio crece a proporción con el parámetro de h. Entre más grande era este parámetro, menor era la cantidad de puntos (ya que el rango de tiempo observado está fijado desde 0 hasta 30), y la diferencia entre los puntos procesados era cada vez mayor.

Algo que se debe admitir es que debido a la cantidad de puntos para cada valor de h, los cálculos de error empleados no son los más adecuados, no se está midiendo un error en igualdad de condiciones (misma cantidad de muestras en cada error cuadrático medio)

El valor de h más pequeño (0.1) demostró ser el más costoso en términos de complejidad temporal. Esto se debe a que es el caso en el que se tienen más puntos de muestra. Resaltando las diferencias entre las dos funciones usadas, no hubo un cambio significativo en la complejidad temporal a pesar de que una de las funciones polinomiales era de grado mayor (3).

No obstante, la diferencia en el grado y naturaleza de las dos funciones utilizadas en el laboratorio devolvieron resultados muy marcados con respecto al cálculo de errores. Para la segunda función (y_2), el error cuadrático medio resultó varios órdenes de magnitud más grande que para la función y_1 en la mayoría de métodos. Creemos que esto se debe a que el error se ve mucho más propagado en la función polinomial aproximada para la función 2 mediante los métodos, mientras que para la función 1 (debido a su grado más bajo) el error no se propaga de forma tan acelerada.

Referente a los métodos multipaso se puede observar una clara mejoría en la precisión de los cálculos de aproximación realizados en las ODE teniendo errores varios grados de magnitud por debajo que los que se conseguían con métodos 'unipaso'.

Para las dos funciones probadas con cada uno de los dos métodos se consiguió un menor ECM en el método multipaso 4 (`adamsBash.m`) logrando un desempeño mayor sobretodo en la ejecución de la segunda función utilizada, donde su error y su desviación estándar fueron notablemente inferiores. Esta mejoría en el desempeño es debido a que este método toma como base para sus cálculos más valores (calculados anteriormente) que el método multipaso 2.

En relación a los métodos para ODEs de orden 2 se encontró un problema en la implementación del método de elementos finitos, dado que este necesita de la realización de varios cálculos previos que involucran las condiciones de frontera, hecho que hace que la función se afecte cuando la primera condición de frontera no es 0. La solución entregada no ha sido probada en estos casos críticos.

Quizá sea debido a estas condiciones propias de la solución que el método de diferencias medias tuvo un mayor desempeño a la hora de ejecutar las dos funciones proporcionadas. Otro factor que posiblemente influya es el hecho de que el método de elementos finitos retorna un vector de coeficientes representando un polinomio, por lo que puede haber un sesgo mucho mayor que cuando se retornan directamente los valores calculados para esos puntos, como se realiza en el método de diferencias finitas.

El parámetro m tuvo cierta incidencia en los resultados mostrados por el método de elementos finitos, ya que al influir este parámetro directamente en el grado del polinomio hacía los cálculos de éste inestable. Por lo que se tomó la decisión de disminuir los valores del parámetro *m.* Se observa, adicionalmente, que el error calculado no es proporcional al parámetro *m.*