MÉTODOS NUMÉRICOS AVANZADOS
Problemas de Valor Inicial

Aproximación de soluciones de ecuaciones diferenciales por métodos iterativos

Augusto Nizzo Mc Intosh - Legajo: 47443

18 de mayo de 2009

Resumen

Este informes muestra como realizar una aproximación de la solución de una ecuación diferencial por los métodos de Euler, Heun y Runge-Kutta 4, y como el tamaño del paso de integración impacta sobre el resultado final.

Palabras Claves: métodos iterativos, error, IVP, problemas de valor inicial, tamaño de paso, ecuaciones diferenciales, euler, heun, runge-kutta 4, rk4.

1. Introducción

Son muchas las fuentes de ecuaciones diferenciales, algunas de ellas son los problemas físicos. Donde, por ejemplo, es necesario conocer la posición de una partícula sumergida en un medio viscoso en un tiempo determinado, donde dicha posición dependen tanto de la velocidad inicial de la partícula como de la fricción producida por el medio, que depende de la velocidad de la misma.

Así mismo, son muy pocos los modelos que llevan a soluciones algebráicas exáctas. Es necesario recurrir a métodos que permitan obtener una solución aproximada, donde existirá un error, que dependerá tanto del algoritmo utilizado como de parámetros computacionales.

Se mostrará en las siguientes secciones, haciendo uso del software matemático *GNU Octave*, el cómputo de una ecuación diferencial proveniente de un sistemá físico por métodos iterativos. En el cual se poseé la solución exacta que se usará para medir el impácto sobre el error debido al tamaño del paso de integración.

Este informes está destinado a personas que tengan conocimiento sobre métodos numéricos, programación y ecuaciones diferenciales.

2. Objetivo

Dado el IVP:

$$\frac{dy}{dy} = 100(\sin t - y), y(0) = 0 \tag{1}$$

cuya solución exacta es:

$$y(t) = \frac{\sin t - 0.01\cos t + 0.01e^{-100t}}{1.0001} \tag{2}$$

Integrar hasta aproximar $y(3)$ con pasos $h = 0.015, 0.020, 0.025, 0.030$ usando Euler, Heun y RK4. Comentar y explicar los resultados.

Un posible origen de la ecuación diferencial (1) es el cálculo de la intensidad de corriente en un circuito eléctrico serie formado por una resistencia R, un inductor L y una fuente de tensión V, como muestra la **Figura 1**. Aplicando la Segunda Ley de Kirchoff[1] se obtiene

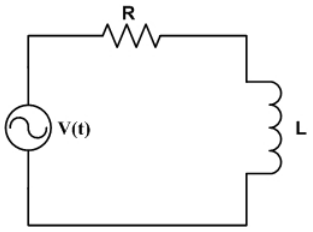

Figura 1: *Circuito RL serie*

$$i(t)R + L\frac{di(t)}{dt} - V(t) = 0 \tag{3}$$

Despejando $\frac{di(t)}{dt}$ adecuadamente

$$\frac{di(t)}{dt} = \frac{V(t)}{L} - \frac{i(t)R}{L} \tag{4}$$

Si $V(t)$ es de la forma $V(t) = V_0 \sin t$ siendo V_0 el valor máximo de la tensión entregado por la fuente

$$\frac{di(t)}{dt} = \frac{V_0}{L}\sin t - i(t)\frac{R}{L} \tag{5}$$

y además R, L y V_0 son tales que se cumple $\frac{V_0}{L} = \frac{R}{L} = 100$ se obtiene la equación diferencial (1). En formato estándar:

$$\begin{cases} y' = f(t, y(t)) = 100(\sin t - y(t)) \\ y(t_0) = 0 \end{cases} \tag{6}$$

[1] "La suma algebráica de las diferencias de potencial de los elementos de una malla cerrada es cero."

3. Resolución

Según (2) $y(3) \simeq 0.15100483254261$. Haciendo uso de los métodos de Euler, Heun y Runge-Kutta 4, se obtendran resultados distintos que se usaran para comparar con el valor exacto de $y(3)$. Se escribió en *GNU Octave* el **Código 3.1**, que define una función, que se usará para hayar las distintas aproximaciones.

Los códigos **Código 3.2.1**, **Código 3.3.1** y **Código 3.4.1**, implementan los métodos de Euler, Heun y Runge-Kutta 4 respectivamente.

Codigo 3.1. Definición de la función *ivp13* en *GNU Octave*

```
% Archivo: ivp13.m
% -------------------
% Define la funcion ivp13, cuyos argumentos son
% Entrada:
%  - t: variable independiente
%  - y: valor y de la función en t
% Salida:
% - x: valor de la derivada de y, evaluada en t.
function x = ivp13(t, x)
 x = 100*(sin(t) - y);
endfunction
```

3.1. Aplicación del Método de Euler

Este método es el más sencillo de los tres, es simple de codificar y de entender, pero no se suele utilizar porque el error acumulado es muy "grande", produciendo diferencias apreciables en el resultado final. Aunque en el ejemplo que estamos tratando, para el menor paso, es el que mejor aproxima el valor de $y(3)$.

Su construcción está basada en el polinomio de *Taylor* de grado 1 tal que, si el paso h se hace suficientemente pequeño, se puede despreciar el úlitmo término.

$$y(t+h) = y(t) + hf(t,y) + y''(c)\frac{h^2}{2}, c \in int\{t, t+h\} \tag{7}$$

Llamando $y(t)$ como y_k e $y(t+h) = y_{k+1}$ el polinomio queda discretizado

$$\begin{cases} y_{k+1} = y_k + hf(t_k, y_k) & \\ t_{k+1} = t_k + h & k = 0, 1, 2, \ldots, M-1 \\ y_0 = y(t_0) & \end{cases} \tag{8}$$

donde M es el número de pasos de integración y h el tamaño de los mismos. El error global de este algorímo es $O(h)$. La implementación del método se puede ver en el **Código 3.1.1**

En la **Tabla 1** se puede ver el impácto que tiene el tamaño del paso frente al error.

Codigo 3.1.1. Implementación del método de Euler en *GNU Octave*

```
function R = euler(f, a, b, y0, M)
% Aproxima el ivp "f", en "b" a partir de "a" con valor inicial "y0" y "M" pasos
% con el Método de Euler.
% Salida:
% Vector de dimension M+1x2, que contiene el valor de la variable
% independiente en la primer componente y el valor de la funcion
% en dicho punto.
% Entrada:
% - f : La funcion en formato estandart f(t, y(t))
% - a : ordenada inicial
% - b : ordenada final
% - y0: valor de la funcion en a.
% - M : cantidad de pasos.

 h = (b-a)/M; %Tamaño del paso.
 T=zeros(1, M+1);
 Y=zeros(1, M+1);
 T = a: h: b;
 Y(1) = y0;
 for k = 1:M
 Y(k+1) = Y(k) + h*feval(f, T(k), Y(k));
 endfor
R = [T' Y'];
endfunction
```

Al aumentar el tamaña del paso a $h = 0.020134$, es decir, $M = 149$, se puede observar en la **Figura 4** como el error se hace incontrolable. El método de Euler deja de ser útil para este problema.

3.2. Aplicación del Método de Heun

Este método propone una mejora al método de Euler, en el cual, se realiza una predicción de cuanto será el valor de la función en proximo paso iterativo, para luego corregirlo utilizando la regla del trapecio. [2]
El esquema iterativo es el siguiente:

$$\begin{cases} p_{k+1} = p_k + hf(t_k, y_k) \\ y_{k+1} = y_k + \frac{h}{2}(f(t_k, y_k) + f(t_{k+1}, p_{k+1}) \\ t_{k+1} = t_k + h \\ y_0 = y(t_0) \end{cases} \quad k = 0, 1, 2, \ldots, M-1 \tag{9}$$

El error por este método es $O(h^2)$. En la **Codigo 3.2.1** se puede ver la implementación del método de Heun con *GNU Octave*

[2] Mathews - Fink. Métodos Numéricos con MATLAB, El método de Heun.

Codigo 3.2.1. Implementación del método de Heun en *GNU Octave*

```
function R = heun(f, a, b, y0, M)
% Aproxima del ivp "f", en "b" a partir de "a" con valor inicial "y0" con "M" pasos
% con el Método de Heun.
% Salida:
% Vector de dimension (M+1)x2, que contiene el valor de la variable
% independiente en la primer componente y el valor de la funcion
% en dicho punto.
% Entrada:
% - f : La funcion en formato estandart f(t, y(t))
% - a : ordenada inicial
% - b : ordenada final
% - y0: valor de la funcion en a.
% - M : cantidad de pasos.

 h = (b-a)/M; %Tamaño del paso.
 T=zeros(1, M+1);
 Y=zeros(1, M+1);
 T = a: h: b;
 Y(1) = y0;
 for k = 1:M
 p = Y(k) +  h*feval(f, T(k), Y(k));
 Y(k+1) = Y(k) + (h/2)*(feval(f, T(k), Y(k)) + feval(f, T(k+1), p));
 endfor
R = [T' Y'];
endfunction
```

Realizando ciento cuarenta y nueve pasos, $M = 149$ ó $h = 0.020134$ se puede observar en **Figura 4** que el error relativo acumulado por el método supera el 10 %. Al igual que el método Euler, este deja de ser útil para este problema.

3.3. Aplicación del método Runge-Kutta 4

Los métodos de Runge-Kutta N se construyen a partir del método de *Taylor* de orden N tal que manera que el error global sea $O(h^N)$, de tal manera que se evite el cálculo de las derivadas parciales; esto puede conseguirse a cambio de evaluar, en cada paso, la función en varios puntos. Se utilizará el método de Runge-Kutta 4, que se abrevia como RK4, para resolver la ecuación. Es el más popular y es también, para propósitos generales, una buena elección ya que es bastante preciso, estable y fácil de programar. La mayoría de los expertos dicen que no es necesario trabajar con métodos de orden superior porque el aumento del coste computacional no compensa la mayor exactitud.[3]
El error global para este método es $O(h^4)$.

[3]Mathews - Fink. Métodos Numéricos con MATLAB, El método Runge-Kutta 4.

El esquema iterativo del método es

$$\begin{cases} f_{k,1} = f(t_k, y_k) \\ f_{k,2} = f(t_k + \frac{h}{2}, y_k + \frac{h}{2} f_{k,1}) \\ f_{k,3} = f(t_k + \frac{h}{2}, y_k + \frac{h}{2} f_{k,2}) \\ f_{k,4} = f(t_k + h, y_k + h f_{k,3}) \qquad\qquad k = 0, 1, 2, \ldots, M-1 \\ y_{k+1} = y_k + \frac{h}{6}(f_{k,1} + 2(f_{k,2} + f_{k,3}) + f_{k,4}) \\ t_{k+1} = t_k + h \\ y_0 = y(t_0) \end{cases} \tag{10}$$

Si se aumenta el tamaño del paso hasta $h = 0.0345$, es decir, $M = 87$, el error relativo es menor al 10 %. A pasos mayores, el error aumenta, y deja de ser útil el método.

En el **Código 3.3.1** se puede ver la implementación del método Runge-Kutta 4 en *GNU Octave*.

Codigo 3.3.1. Implementación del método RK4 en *GNU Octave*

```
function R = rk4(f, a, b, y0, M)
% Aproxima del ivp "f", en "b" a partir de "a" con valor inicial "y0" con "M" pasos
% con el Método de Runge-Kutta 4.
% Salida:
% Vector de dimension (M+1)x2, que contiene el valor de la variable
% independiente en la primer componente y el valor de la funcion
% en dicho punto.
% Entrada:
% - f : La funcion en formato estandart f(t, y(t))
% - a : ordenada inicial
% - b : ordenada final
% - y0: valor de la funcion en a.
% - M : cantidad de pasos.

 h = (b-a)/M; %Tamaño del paso.
 T=zeros(1, M+1);
 Y=zeros(1, M+1);
 T = a: h: b;
 Y(1) = y0;
 for k = 1:M
 f1 = feval(f, T(k), Y(k));
 f2 = feval(f, T(k) + h/2, Y(k) + (h/2)*f1);
 f3 = feval(f, T(k) + h/2, Y(k) + (h/2)*f2);
 f4 = feval(f, T(k) + h, Y(k) + (h/2)*f3);
 Y(k+1) = Y(k) + (h/6)*(f1 + 2*(f2 + f3) + f4);
 endfor
R = [T' Y'];
endfunction
```

3.4. Comparación de los métodos

En la **Figura 2** está graficado el error de las tres aproximaciones para cada punto entre 0 y 3, con un paso de $h = 0.015$. Se puede observar, que aunque se considera al método de Euler como el menos preciso, para este caso, es el que mejor aproxima a la función. Luego en precisión le sigue el método de Heun y luego Runge-Kutta 4. Los detalles de la aproximación se pueden observar en la **Tabla 1**.

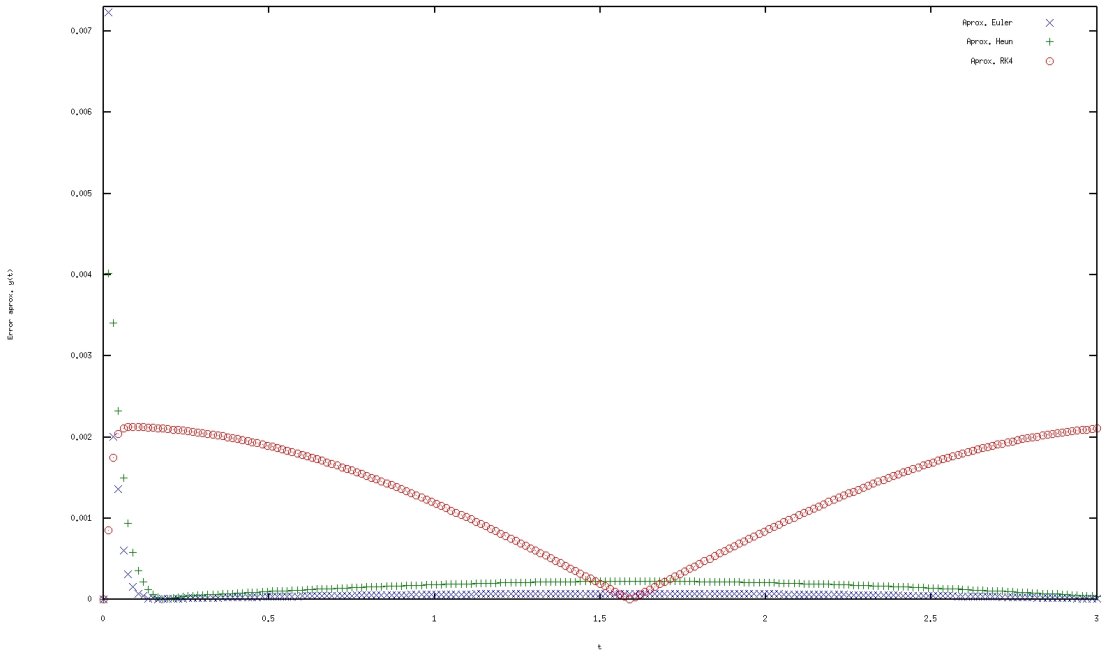

Figura 2: *Errores de aproximación de los métodos Euler, Heun y RK4 con una paso de 0.015*

En la **Figura 3** se puede observar que el método de Euler es el que produce un error mayor de aproximación, es interesante destacar que este se mantiene prácticamente constante a lo largo del intervalo $[0, 3]$, le sigue en precisión el método de Heun y Runge-Kutta 4. El paso de estas aproximaciones es de $h = 0.020$.

En la **Figura 4** se muestra como para un paso de $h = 0.020134$, realizando un paso menos que los anteriores, los métodos de Euler y Heun, producen errores mayores al 10 %. Lo que los hace inútiles para este problema. Runge-Kutta aproxima con errores menores al 3 %.

Las Figuras **Figura 5** y **Figura 6** muestran cómo para pasos de $h = 0.025$ y $h = 0.030$ el método Runge-Kutta 4 realiza aproximaciónes aceptables por debajo del 10 % de error. Para un paso mayor, $h = 0.034483$, 87 pasos, el método produce aproximaciones con un error menor al 10 %. No obstante, para 86 pasos ó $h > 0.034884$ el error relativo lo supera.

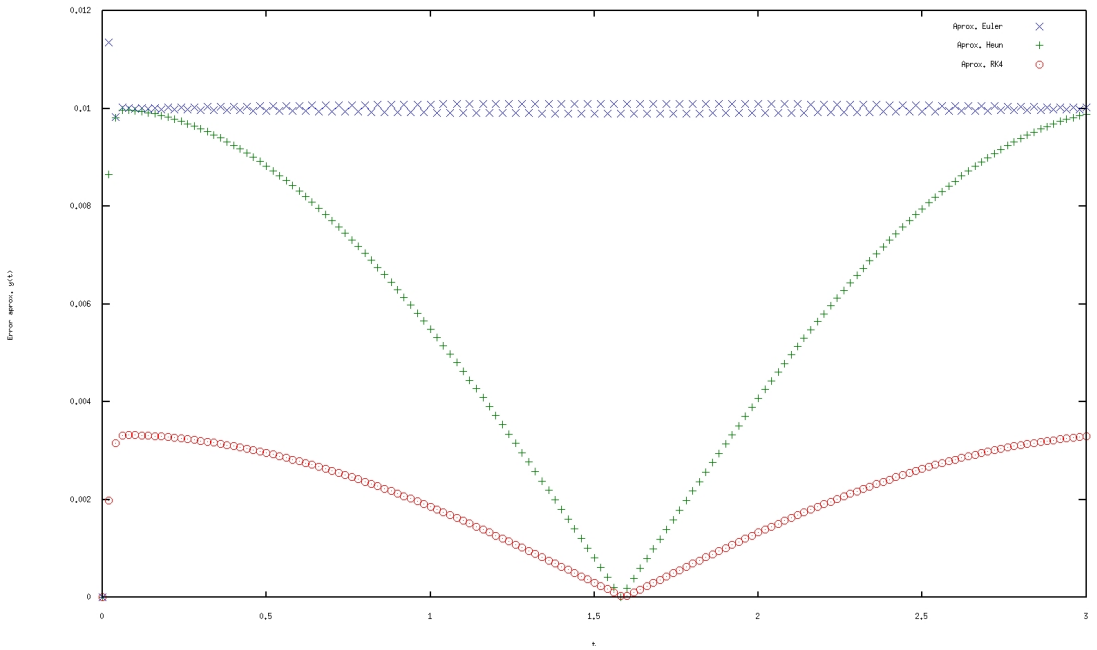

Figura 3: *Errores de aproximación de los métodos Euler, Heun y RK4 con una paso de 0.020*

Tabla 1

$h = 0.015$	Valor de la Aproximación	Cota Error Relativo
Euler	0.15101652866556	0.01 %
Heun	0.15096385497783	2.7 %
RK4	0.14889954826585	1.4 %
$h = 0.020$		
Euler	0.16102059638323	6.7 %
Heun	0.14112000805986	6.6 %
RK4	0.14889954826585	2.2 %
$h = 0.025$		
RK4	0.148899548265859	3.3 %
$h = 0.030$		
RK4	0.14311135360500	5.3 %

4. Conclusiones

Los métodos presentados, ofrecen cómo información unicamente el orden del error, $O(h)$ para el método de Euler, $O(h^2)$ para Heun y $O(h^4)$ para RK4. Esta información solo es útil para determinar que impacto tendrá el cambio del paso frente al error. En los gráficos se puede ver que las constantes que determinan el error también afectan apreciablemente a las aproximaciónes. En el primer ejemplo $h = 0.015$, el algorítmo que mayor error produjo

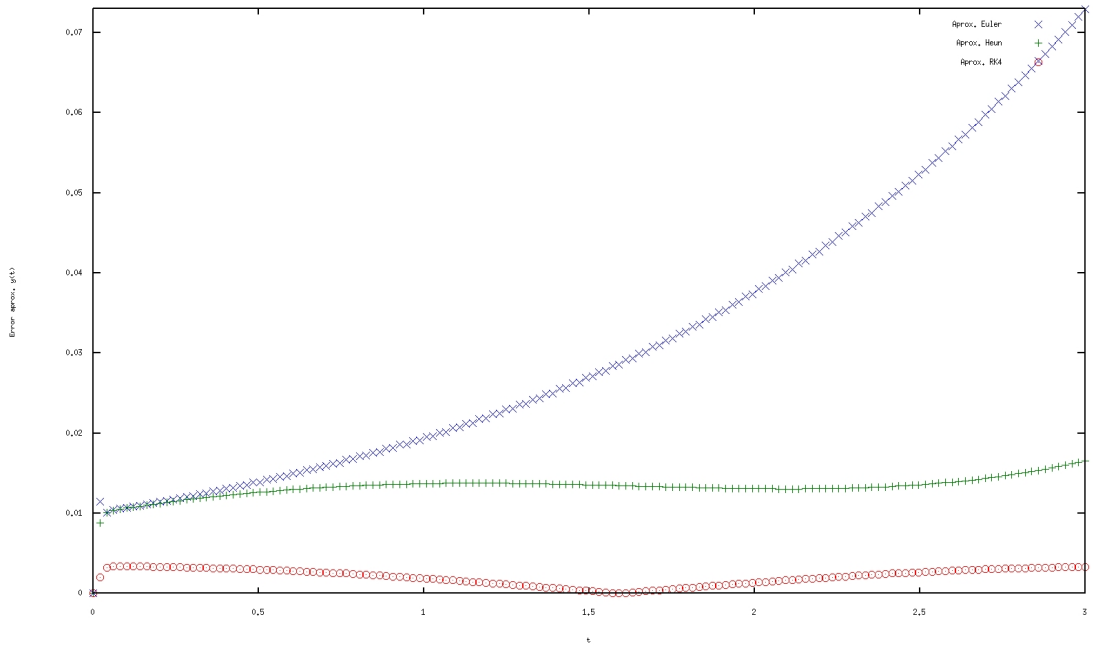

Figura 4: *Errores de aproximación de los métodos Euler, Heun y RK4 con una paso de 0.020134*

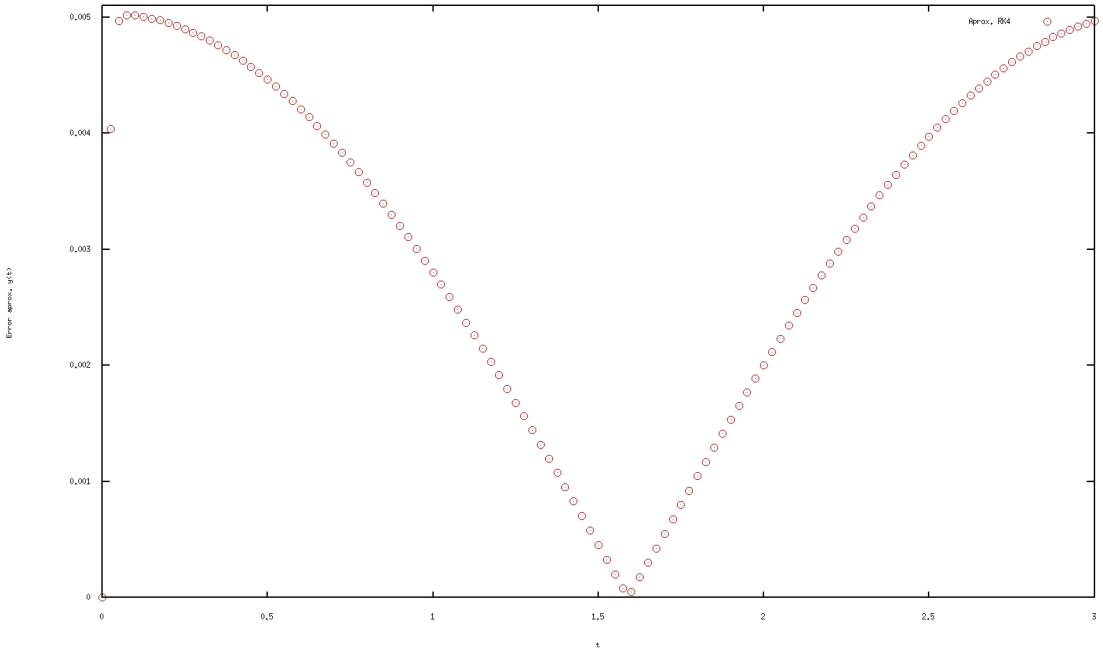

Figura 5: *Errores de aproximación del método RK4 con una paso de 0.025*

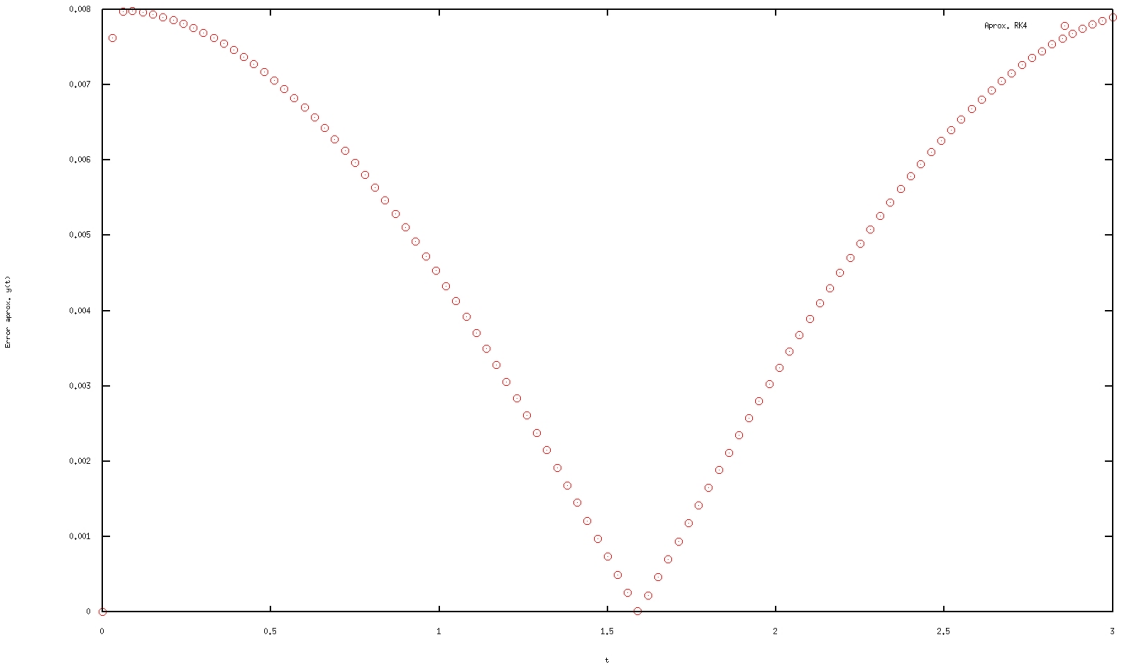

Figura 6: *Errores de aproximación del método RK4 con una paso de 0.030*

fue RK4, cuando el *Orden* del error es el menor de los tres.

El error "absoluto" de cada aproximación se conoce solo porque se cuenta con el valor exácto, la expresión algebráica exacta de $y(t)$. Esto por lo general no es viable, que en cuyo caso no tendría sentido usar los métodos. Para realizar un control sobre el error, se podrían utilizar los métodos inversos, Euler, Heun, RK4 *hacia atras* tomando como valor inicial la aproximación producida por el método *hacia adelante* fijando una tolerancia, con la cual, si se supera, se debería disminuir el tamaño del paso. De todos modos, siempre hay que realizar la aproximación, no se puede fijar un paso a *priori* para obtener un error máximo determinado.

5. Referencias

- Prentice Hall - Física V Edición, Wilson - Buffa.
- Prentice Hall - Métodos Numéricos con MATLAB, Mathews - Fink.