Universidad de los Andes
Departamento de Ingeniería Eléctrica y Electrónica
IELE 2009 Computación Científica
Gerardo Andrés Riaño Briceño 201112388

Tarea 4 – Métodos de integración para solución de EDO

Ejercicio 7.1

Tabla 1. Resultados obtenidos con el método de Runge-Kutta 4.

n	x(n)	y(n)	yreal(n)	Error absoluto
1	0.001	-0.002004	-0.002004	-1.65E-10
2	0.002	-0.00401603	-0.00401603	1.61E-11
3	0.003	-0.00603612	-0.00603612	8.13E-11
4	0.004	-0.00806428	-0.00806428	-3.27E-11
5	0.005	-0.01010054	-0.01010054	3.15E-11
6	0.006	-0.01214494	-0.01214494	-2.23E-11
7	0.007	-0.01419749	-0.01419749	-2.25E-11
8	0.008	-0.01625823	-0.01625823	2.14E-11
9	0.009	-0.01832718	-0.01832718	-2.07E-11
10	0.01	-0.02040437	-0.02040437	1.71E-11

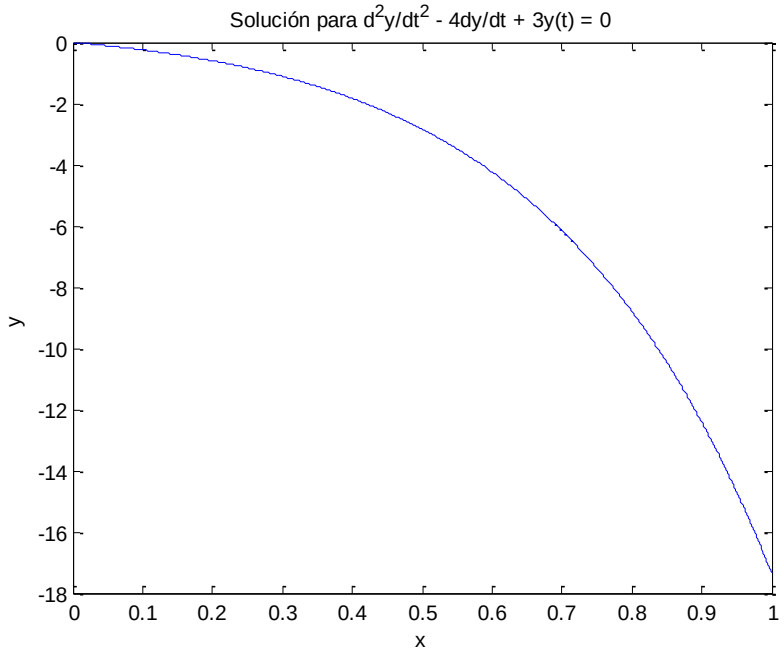


Figura 1. Solución de la EDO con el método RK4 para 0≤x≤1.

El método de Runge-Kutta de cuarto orden permite obtener los resultados errores relativos bastante bajos; son de orden cercano al de los errores de redondeo de la máquina como se muestra en la **Tabla 1**. El error relativo y absoluto disminuye varios órdenes de magnitud con respecto a los métodos como el de Euler. Efectivamente el error global es $O(h^4)$ y el error en cada iteración $O(h^5)$ como se muestra en [2]. Por ende, se tiene que,

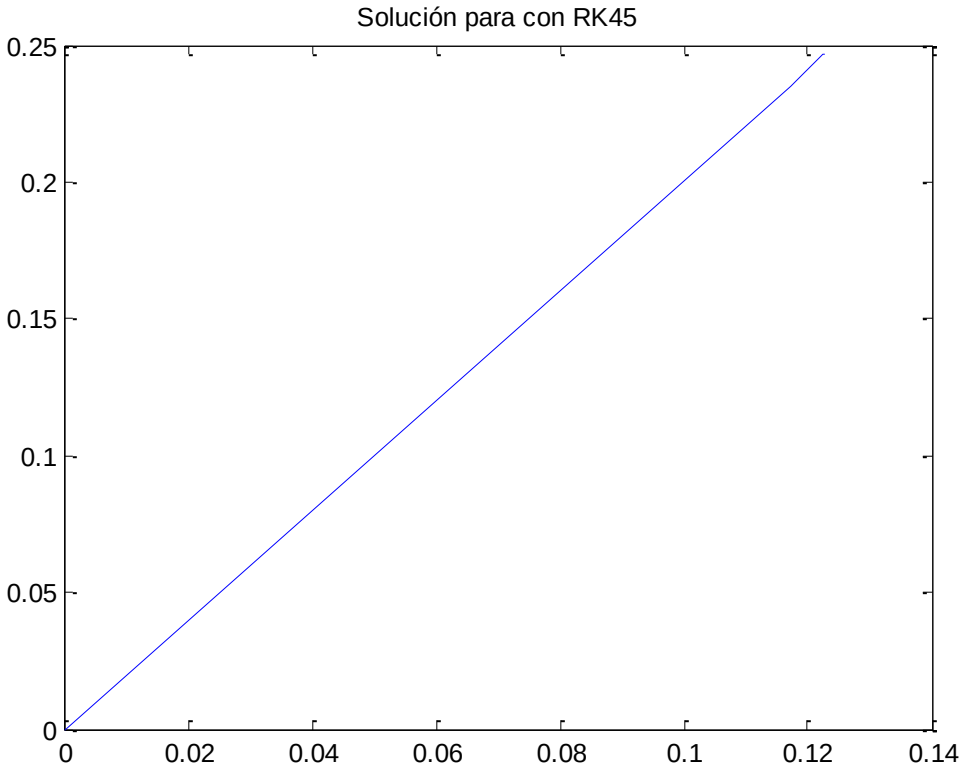
Universidad de los Andes
Departamento de Ingeniería Eléctrica y Electrónica
IELE 2009 Computación Científica
Gerardo Andrés Riaño Briceño 201112388

$$y1_{i+1} = y1_i + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5) \quad ; \quad y2_{i+1} = y2_i + \frac{l_1}{6} + \frac{l_2}{3} + \frac{l_3}{3} + \frac{l_4}{6} + O(h^5)$$

Algoritmo

1. Definir la ecuación diferencial de orden n, como un sistema de n ecuaciones diferenciales de primer orden
2. Para $x = x_{min}$ hasta x_{max}
   a. Calcular $k_j, l_j \quad j = 1, 2, 3, 4$
   b. Calcular valores de $y_k \quad k = 1, 2, \dots, n$

Ejercicio 7.2



Universidad de los Andes
Departamento de Ingeniería Eléctrica y Electrónica
IELE 2009 Computación Científica
Gerardo Andrés Riaño Briceño 201112388

Ejercicio 7.3

Tabla 2. Resultados obtenidos con el método de Hamming.

n	x(n)	p(n)	y(n)	yreal(n)	Error absoluto
0	0	0	1	1	0.00E+00
1	0.01	0	0.99014983	0.99014983	-2.83E-12
2	0.02	0	0.98059867	0.98059867	-6.24E-12
3	0.03	0	0.97134553	0.97134553	-8.49E-12
4	0.04	0.96238944	0.96238944	0.96238944	-8.68E-12
5	0.05	0.95372942	0.95372942	0.95372942	-8.29E-12
6	0.06	0.94536453	0.94536453	0.94536453	-7.75E-12
7	0.07	0.93729382	0.93729382	0.93729382	-8.05E-12
8	0.08	0.92951635	0.92951635	0.92951635	-8.36E-12
9	0.09	0.92203119	0.92203119	0.92203119	-7.77E-12
10	0.1	0.91483742	0.91483742	0.91483742	-8.04E-12
300	3	9.04978707	9.04978707	9.04978707	-1.14E-12

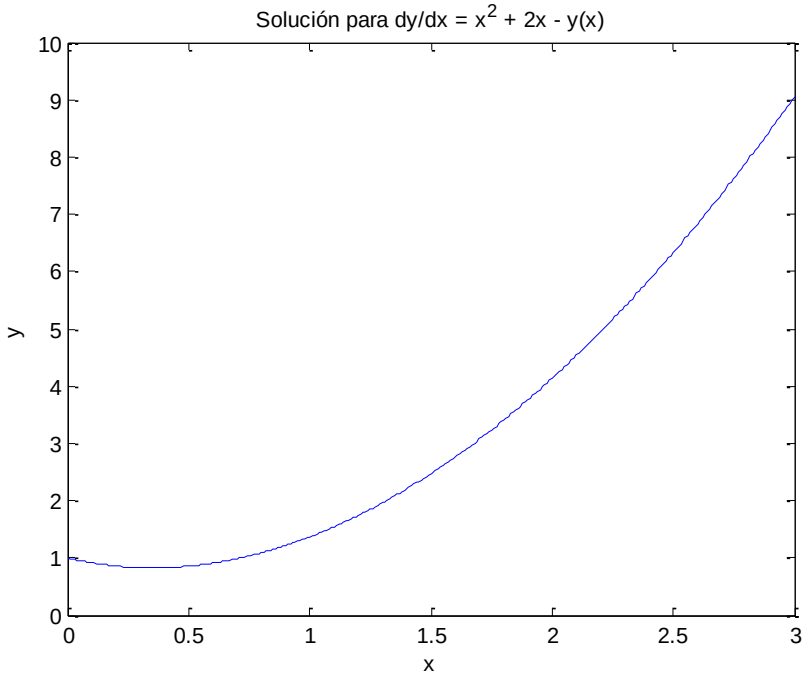


Figura 2. Solución de la EDO con el método de Hamming para $0 \leq x \leq 3$.

En los resultados se evidencia que el método con predicción mejora significativamente la solución de la EDO.

Universidad de los Andes
Departamento de Ingeniería Eléctrica y Electrónica
IELE 2009 Computación Científica
Gerardo Andrés Riaño Briceño 201112388

Referencias

[1] U. o. T. a. Austin, 2013. [En línea]. Available: http://farside.ph.utexas.edu/teaching/329/lectures/node35.html.

[2] S. Rosloniec, Fundamental Numerical Methods for Electrical Engineering, Warsaw: Springer, 2008.

Anexos

Código en Matlab

```
%% Runge-Kutta RK4

clc; clear all;
h = 0.001;
x = 0:h:1;

f1 = @(x, y1, y2) y2;
f2 = @(x, y1, y2) 4*y2 - 3*y1;

y1 = zeros(length(x), 1);
y2 = zeros(length(x), 1);
y1(1) = 0; y2(1) = -2;

for i=2 : length(x)
    k1 = h*f1(x(i-1), y1(i-1), y2(i-1));
    l1 = h*f2(x(i-1), y1(i-1), y2(i-1));

    k2 = h*f1(x(i-1) + h/2, y1(i-1) + k1/2, y2(i-1) + l1/2);
    l2 = h*f2(x(i-1) + h/2, y1(i-1) + k1/2, y2(i-1) + l1/2);

    k3 = h*f1(x(i-1) + h/2, y1(i-1) + k2/2, y2(i-1) + l2/2);
    l3 = h*f2(x(i-1) + h/2, y1(i-1) + k2/2, y2(i-1) + l2/2);

    k4 = h*f1(x(i-1) + h, y1(i-1) + k3, y2(i-1) + l3);
    l4 = h*f2(x(i-1) + h, y1(i-1) + k3, y2(i-1) + l3);

    y1(i) = y1(i-1) + (k1 + 2*k2 + 2*k3 + k4)/6;
    y2(i) = y2(i-1) + (l1 + 2*l2 + 2*l3 + l4)/6;
end
```

Universidad de los Andes
Departamento de Ingeniería Eléctrica y Electrónica
IELE 2009 Computación Científica
Gerardo Andrés Riaño Briceño 201112388

%% Runge-Kutta RKF 45

```
clc; clear all;
f = @(x, y) 2 + y.^2/2;
tol = 1e-7;
h = 0.1;
y = []; y(1) = 0; x = []; x(1) = 0;
z = []; z(1) = 0;
error = 1000;
i = 2;

while i < 1000
    k1 = h*f(x(i-1), y(i-1));
    k2 = h*f(x(i-1) + h/4, y(i-1) + k1/4);
    k3 = h*f(x(i-1) + 3*h/8, y(i-1) + 3*k1/32 + 9*k2/32);
    k4 = h*f(x(i-1) + 12*h/13, y(i-1) + 1932*k1/2197 - 7200*k2/2197 +
7296*k3/2197);
    k5 = h*f(x(i-1) + h, y(i-1) + 439*k1/216 - 8*k2 + 3680*k3/513 -
845*k4/4104);
    k6 = h*f(x(i-1) + h/2, y(i-1) - 8*k1/27 + 2*k2 - 3544*k3/2565 -
1859*k4/4104 - 11*k5/40);
    y(i) = y(i-1) + 25*k1/216 + 1408*k3/2565 + 2197*k4/4104 - k5/5;
    z(i) = y(i-1) + 16*k1/135 + 6656*k3/12825 + 28561*k4/56430 - 9*k5/50
+ 2*k6/55;

    % Cálculo del nuevo tamaño de paso
    s(i) = (tol*h/(2*abs(z(i) - y(i)))).^0.25;
    x(i) = x(i-1) + h;
    h = h*s(i);
    error = abs(z(i) - y(i));
    i = i + 1;
end
```

%% Método de Hamming

```
f = @(x,y) x^2 + 2*x - y;
h = 0.01;
x = 0:h:3;
y = ones(length(x), 1);

% Se definen los primeros cuatro valores de la EDO
% con el método de Runge-Kutta de 4 orden (RK4)
for i=2 : length(x)
    k1 = h*f(x(i-1), y(i-1));
    k2 = h*f(x(i-1) + h/2, y(i-1) + k1/2);
    k3 = h*f(x(i-1) + h/2, y(i-1) + k2/2);
    k4 = h*f(x(i-1) + h, y(i-1) + k3);
    y(i) = y(i-1) + (k1 + 2*k2 + 2*k3 + k4)/6;
end

% Se inicializan parámetros para Hamming
f0 = f(x(1),y(1)); f1 = f(x(2),y(2));
```

Universidad de los Andes
Departamento de Ingeniería Eléctrica y Electrónica
IELE 2009 Computación Científica
Gerardo Andrés Riaño Briceño 201112388

```
f2 = f(x(3),y(3)); f3 = f(x(4),y(4));

p_old = 0; c_old = 0;

for i=4 : length(x)-1
    % Cálculo de predicción
    p(i+1) = y(i-3) + 4*h*(2*f1 - f2 + 2*f3)/3;
    % Corrección
    p_mod = p(i+1) + 112*(c_old-p_old)/121;
    f4 = f(x(i+1),p_mod);
    c_new = (9*y(i) - y(i-2) + 3*h*(-f2+2*f3+f4))/8;
    % Valor corregido
    y(i+1) = c_new + 9*(p(i+1)-c_new)/121;
    % Actualización de variables
    p_old = p(i+1); c_old = c_new;
    f1 = f2; f2 = f3;
    f3 = f(x(i+1),y(i+1));
end
```