Método de Runge-Kutta para Ecuaciones Diferenciales

Uno de los métodos más utilizados para resolver numéricamente problemas de ecuaciones diferenciales ordinarias con condiciones iniciales es el método de Runge-Kutta de cuarto orden, el cual proporciona un pequeño margen de error con respecto a la solución real del problema y es fácilmente programable en un software para realizar las iteraciones necesarias.

El método de Runge-Kutta se utiliza para resolver ecuaciones diferenciales de la forma

$$\frac{dy(t)}{dt} = f(t, y), \qquad y(t_0) = y_0$$

Y es sumamente útil para casos en los que la solución no puede hallarse por los métodos convencionales (como separación de variables). Hay variaciones en el método de Runge-Kutta de cuarto orden pero el más utilizado es el método en el cual se elige un tamaño de paso *h* y un número máximo de iteraciones *n* tal que

$$y_0 = y(t_0)$$

$$k_1 = h \cdot f(t_i, y_i)$$

$$k_2 = h \cdot f\left(t_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right)$$

$$k_3 = h \cdot f\left(t_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right)$$

$$k_4 = h \cdot f\left(t_i + h, y_i + k_3\right)$$

Y se realiza la iteración

$$y_{i+1} = y_i + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)$$

Para $i = 0, \ldots, n-1$. La solución se da a lo largo del intervalo $(t_o, t_o + hn)$.

El algoritmo para el método de Runge-Kutta de cuarto orden en seudo código es el siguiente:

INICIO

INPUT: Número de iteraciones *n* (o tamaño de paso *h*), punto inicial del intervalo *a* (punto final del intervalo *b*), condición inicial $y\,(t_0) = y_0$.

$n = (b - a)/h;$

$t = t_0;$

$y = y_0;$

$OUTPUT \quad (t, y)$

PARA i = 1,…, *n*;

$k_1 = h \cdot f(t, y);$

$$k_2 = h \cdot f\left(t + \frac{h}{2}, y + \frac{k_1}{2}\right);$$

$$k_3 = h \cdot f\left(t + \frac{h}{2}, y + \frac{k_2}{2}\right);$$

$$k_4 = h \cdot f(t + h, y + k_3);$$

$$y = y + (k_1 + 2k_2 + 2k_3 + k_4);$$

$$t = t + i \cdot h;$$

$OUTPUT \quad (t, y)$

FIN PARA

FIN

Un software apropiado y muy útil además de fácil de programar es Matlab, con el cual resolveremos el siguiente ejemplo:

- Resolver numéricamente con 100 iteraciones en el intervalo [1, 100] la ecuación diferencial con condiciones iniciales dada a continuación:

$$t \cdot \frac{dy(t)}{dt} = \sin(4t), \quad y(1) = 0$$

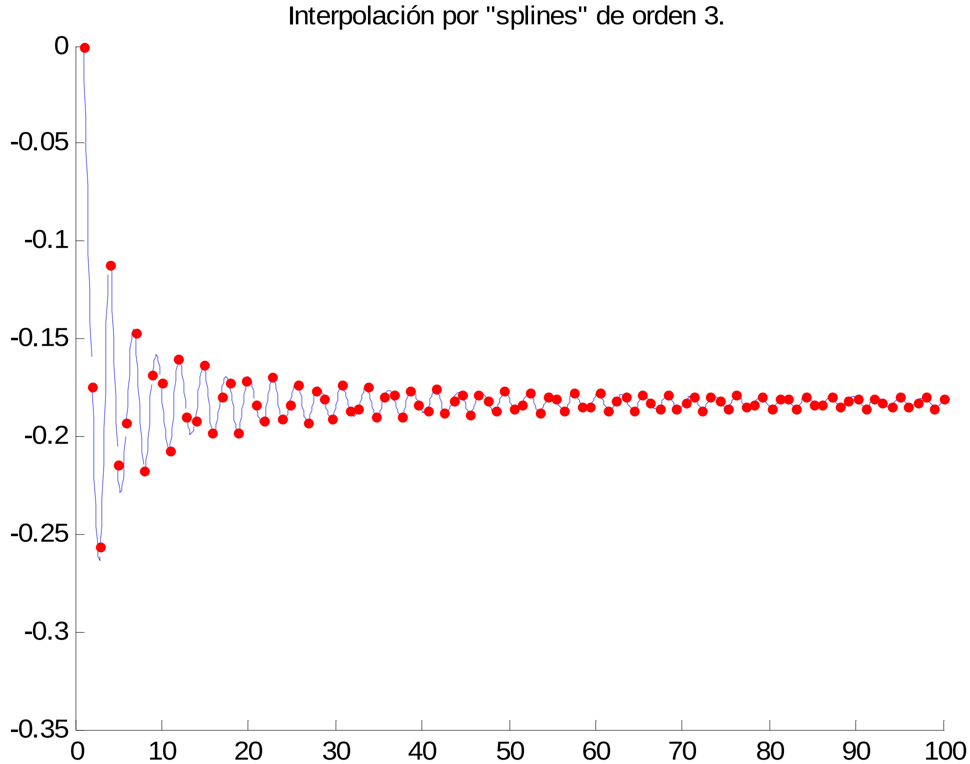
Solución: Es claro que la ecuación diferencial dada no tiene una solución analítica exacta ya que al realizar separación de variables nos encontramos con una función que no posee antiderivada. Entonces tenemos que

$$f(t, y) = \frac{\sin(4t)}{t}$$

Un algoritmo en Matlab para realizar la iteración es el siguiente:

```
function [A]=RungeKutta4

n=100;
a=1;
b=100;
t=1;
y=0;
h=(b-a)/n;

X(1,1)=t;
X(2,1)=y;

for i=1:n;
    k1=h*sin(4*t)/t;
    k2=h*sin(4*(t+h/2))/(t+h/2);
    k3=h*sin(4*(t+h/2))/(t+h/2);
    k4=h*sin(4*(t+h))/(t+h);

    y=y+(k1+2*k2+2*k3+k4)/6;
    t=a+i*h;

    X(1,i+1)=t;
    X(2,i+1)=y;
end

spline3(X);

A(:,1)=X(1,:)
A(:,2)=X(2,:)
```

La gráfica del resultado que entrega el programa es la siguiente:



- Resolver la ecuación diferencial

$$y' = 0.01 \cdot (70 - y)(50 - y) \quad con \;\; y(0) = 0$$

con el método de Runge-Kutta de 4° orden con h = 0.5 en el intervalo [0, 20] y comparar con la solución exacta

$$y(t) = 350 \frac{1 - e^{-0.2t}}{7 - 5e^{-0.2t}}$$

Solución: Para hallar la solución numérica por el método de Runge-Kutta de cuarto orden se elaboró un algoritmo en Matlab que grafica la solución numérica en rojo marcando los puntos con asteriscos y uniéndolos por medio de rectas y en la misma pantalla grafica la solución exacta en azul. El programa también entrega una tabla que tiene el valor de t en la primera columna, el valor y^* de la aproximación hallada numéricamente en la segunda columna, el valor de y exacto en la tercera columna, y el error absoluto | y-y^*|. Todo lo anterior en el intervalo [0, 20].

El algoritmo en lenguaje Matlab es el siguiente:

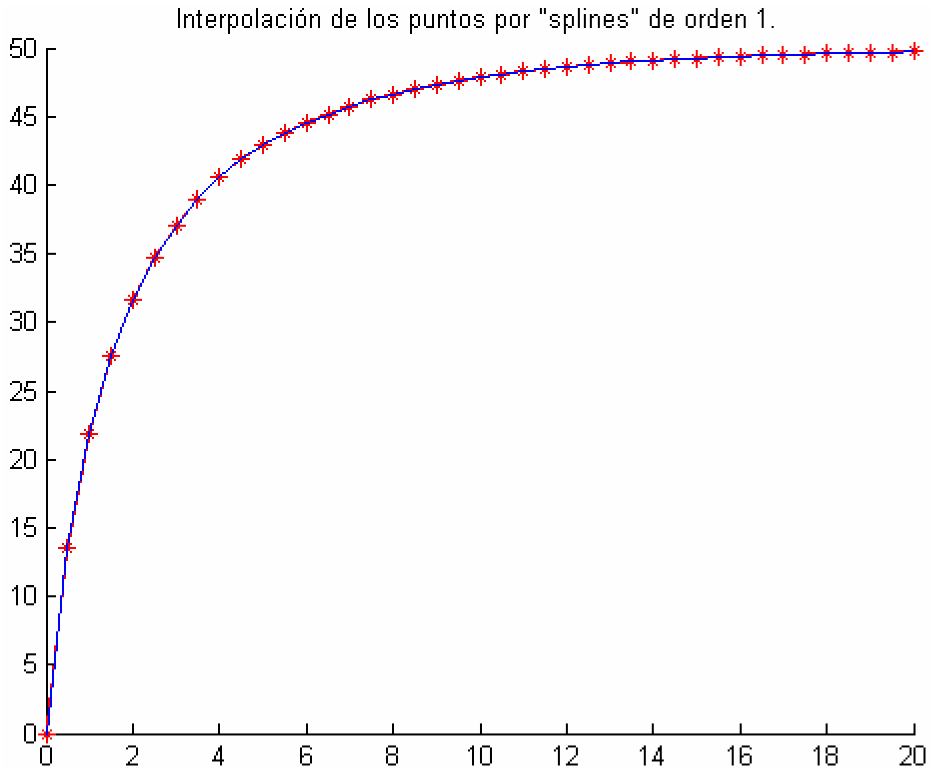
```
function [A]=RungeKutta4_1

a=0;
b=20;
t=0;
y=0;
h=0.5;

n=(b-a)/h;
X(1,1)=t;
X(2,1)=y;

for i=1:n;
    k1=h*0.01*(70-y)*(50-y);
    k2=h*0.01*(70-(y+k1/2))*(50-(y+k1/2));
    k3=h*0.01*(70-(y+k2/2))*(50-(y+k2/2));
    k4=h*0.01*(70-(y+k3))*(50-(y+k3));

    y=y+(k1+2*k2+2*k3+k4)/6;
    t=a+i*h;

    X(1,i+1)=t;
    X(2,i+1)=y;

end

n=length(X(1,:));
for i=1:n-1;
    m(i)=(X(2,i+1)-X(2,i))/(X(1,i+1)-X(1,i));
    b(i)=X(2,i);
    x=X(1,i):0.01:X(1,i+1);
    y=m(i)*(x-X(1,i))+b(i);
    hold on;
    plot(x,y,'r');
end
for i=1:n;
    hold on;
    plot (X(1,i),X(2,i),'*','MarkerEdgeColor','r','LineWidth',1);
    title('Interpolación de los puntos por "splines" de orden 1.');
end

%Solución exacta:
x=0:0.5:20;
y=350*(1-exp(-0.2*x)).*(7-5*exp(-0.2*x)).^(-1);
hold on;
plot(x,y,'b');

A(:,1)=X(1,:);
A(:,2)=X(2,:);
```

```
A(:,3)=y;
A(:,4)=abs(A(:,3)-A(:,2));
```

Al correr el programa se tiene entonces:

t	*y**	*y*	| *y*-*y**|
0	0	0	0
0.5000	13.4511	13.4529	0.0018
1.0000	21.8278	21.8296	0.0018
1.5000	27.5216	27.5231	0.0015
2.0000	31.6258	31.6270	0.0012
2.5000	34.7110	34.7119	0.0010
3.0000	37.1040	37.1048	0.0008
3.5000	39.0058	39.0065	0.0006
4.0000	40.5466	40.5471	0.0005
4.5000	41.8144	41.8148	0.0005
5.0000	42.8710	42.8714	0.0004
5.5000	43.7610	43.7614	0.0003
6.0000	44.5175	44.5178	0.0003
6.5000	45.1654	45.1656	0.0002
7.0000	45.7238	45.7240	0.0002
7.5000	46.2079	46.2081	0.0002
8.0000	46.6296	46.6297	0.0002
8.5000	46.9984	46.9986	0.0001
9.0000	47.3223	47.3224	0.0001
9.5000	47.6076	47.6077	0.0001
10.0000	47.8596	47.8597	0.0001
10.5000	48.0829	48.0829	0.0001
11.0000	48.2810	48.2811	0.0001
11.5000	48.4572	48.4572	0.0001
12.0000	48.6142	48.6142	0.0001
12.5000	48.7543	48.7543	0.0001
13.0000	48.8795	48.8795	0.0001
13.5000	48.9915	48.9915	0.0000
14.0000	49.0918	49.0918	0.0000
14.5000	49.1818	49.1818	0.0000
15.0000	49.2625	49.2625	0.0000
15.5000	49.3350	49.3350	0.0000
16.0000	49.4002	49.4002	0.0000
16.5000	49.4588	49.4588	0.0000
17.0000	49.5116	49.5116	0.0000
17.5000	49.5591	49.5591	0.0000
18.0000	49.6019	49.6019	0.0000
18.5000	49.6404	49.6405	0.0000

19.0000	49.6752	49.6752	0.0000
19.5000	49.7066	49.7066	0.0000
20.0000	49.7349	49.7349	0.0000



Puede observarse en la gráfica que la aproximación realizada por el método de Runge-Kutta es muy cercana al valor exacto de la solución, lo cual puede confirmarse con la vista de los errores absolutos, siendo el mayor error del orden de 10^{-3} e igual a cero luego de t =13.5, es decir, la aproximación numérica es igual a la solución real en esos casos.

Bibliografía:

BURDEN, Richard. Análisis Numérico. 2ª edición. Grupo Editorial Iberoamérica.