Ingeniería en Seguridad Informática y Redes

Ingeniería en Sistemas Computacionales

Algoritmos de Solución Numérica

Método de Runge-Kutta (RK4)

IS727272 - Marco Ricardo Cordero Hernández

SI727576 - Edgar Guzmán Claustro

Jal., 15 de noviembre de 2023

Índice

Contexto

En revisiones anteriores, se tuvo la oportunidad de analizar e implementar el método de Euler (UNMdP, 2015) para resolver ecuaciones diferenciales de primer orden. Este método es parte de una familia de métodos *iterativos* denominados como *métodos Runge-Kutta* (Zheng & Zhang, 2017), los cuales son métodos de pasos singulares, pero con múltiples etapas por cada uno de ellos (Illinois Institute of Technology, s.f.).

Así como es el caso del método de Euler, existen otros métodos generalizados por la fórmula:

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i$$

En donde *s* es el orden del método y *h* es el paso por iteración ($\frac{x_1 - x_0}{n}$). El análisis de los términos dentro de la sumatoria van más allá del alcance del trabajo actual.

El método Runge-Kutta de cuarto orden, también conocido como RK4, es el más popular de estos métodos, llegando a ser referido como *el* método Runge-Kutta (Cheever, 2022). Su fama se atribuye a la precisión de los resultados que genera sin demasiada complejidad en su implementación. Cumpliendo con las condiciones iniciales:

$$y' = f(x, y);\ y(x_0) = y_0$$

El método RK4 puede implementarse haciendo uso de la siguientes fórmulas:

$$h = \frac{x_1 - x_0}{n} \text{ conociendo al menos dos de } x_1, n \text{ o } h$$

$$k_1 = h \bullet f(x_n, y_n)\,; k_2 = h \bullet f(x_n + \tfrac{1}{2}h,\ y_n + \tfrac{1}{2}k_1)$$

$$k_3 = h \bullet f(x_n + \tfrac{1}{2}h,\ y_n + \tfrac{1}{2}k_2);\ k_4 = h \bullet f(x_n + h, y_n + k_3)$$

$$y_{n+1} = y_n + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Donde x_0 es un punto inicial y y_0 es la evaluación del mismo punto inicial en *y*.

Tomando la premisa anterior del método de Euler, la implementación iterativa puede realizarse fácilmente a través de la herramienta Matlab (2023) en relativamente pocos pasos y con comprobaciones mínimas.

Resultados

Comprobación previa

Valores de referencia	
f(x,y)	2*x*y
x_0	0
$y(x_0)$	1
h	0.1
y(0.5)	?

Iteraciones					
x	y	k1	k2	k3	k4
0	1	0	0.01	0.01005	0.020201
0.1	1.010050167	0.020201003	0.03060452	0.030760573	0.04163243
0.2	1.04081077	0.041632431	0.053081349	0.053367572	0.065650701
0.3	1.094174265	0.065650456	0.078889965	0.079353347	0.093882209
0.4	1.173510814	0.093880865	0.109840612	0.110558801	0.128406961
0.5	**1.284025256**	0.128402526	0.148304917	0.149399549	0.172010977

Código

```
% Restablecer entorno
clear, clc

% Introducción y valores iniciales
fprintf('\n \tRESOLUCIÓN DE ECUACIONES DIFERENCIALES POR MÉTODO RUNGE-KUTTA DE ORDEN 4\n');
f = inline(input('\nIngrese la ecuación diferencial -> ','s'));
x0 = input('\nIngrese el primer punto x0: ');
y0 = input('\nIngrese la condición inicial y(x0): ');
n = input('\nIngrese el número de pasos n: ');

% Inicializar variables auxiliares
x1 = 0; h = 0;

% Decisión para variable adicional
dec = input('\nSelecciona entre\n 1) x1\n 2) h\n Elección: ');

if (dec == 1)
   x1 = input('\nIngrese el segundo punto x1: ');
   h = (x1 - x0)/n;
elseif (dec == 2)
   h = input('\nIngrese h: ');
   x1 = h * n + x0; % Opcional
else
   fprintf('Parámetros incorrectos. Intenta de nuevo.\n');
   return
end

% Imprimir encabezado de tabla
fprintf('\nIteración\t\tx\t\ty');
for ord=1:4; fprintf('\t\tk%d', ord); end
fprintf('\n');

% Iterar sobre número de pasos definidos
k = [];
for i = 0:n
   % Mostrar iteración
   fprintf('\t%d\t\t%f\t%f', i, x0, y0);
```

```
    % Cálculo de términos y muestra
    k(1) = h * feval(f, x0, y0);
    k(2) = h * feval(f, x0 + 1/2 * h, y0 + 1/2 * k(1));
    k(3) = h * feval(f, x0 + 1/2 * h, y0 + 1/2 * k(2));
    k(4) = h * feval(f, x0 + h, y0 + k(3));
    for kn = k
        fprintf('\t%f', kn);
    end
    if (i == n); break; end

    % Reasignación de variables para próxima iteración
    y0 = y0 + (1/6) * (k(1) + 2*k(2) + 2*k(3) + k(4));
    x0 = x0 + h;

    fprintf('\n');
end

fprintf('\n\n El punto aproximado y(x1) es = %10.10f\n', y0);
```

Resultados

Primero, el programa le solicita al usuario los datos de entrada básicos.

```
        RESOLUCIÓN DE ECUACIONES DIFERENCIALES POR MÉTODO RUNGE-KUTTA DE ORDEN 4

Ingrese la ecuación diferencial -> 2*x*y

Ingrese el primer punto x0: 0

Ingrese la condición inicial y(x0): 1

Ingrese el número de pasos n: 5
```

Después, como paso intermedio, pregunta cómo desea proceder con la variable adicional.

```
Selecciona entre
 1) x1
 2) h
 Elección: 1

Ingrese el segundo punto x1: 0.5
```

```
Selecciona entre
 1) x1
 2) h
 Elección: 2

Ingrese h: 0.1
```

Independientemente de la elección, el resultado será el mismo.

```
Iteración               x               y              k1              k2              k3              k4
        0        0.000000        1.000000        0.000000        0.010000        0.010050        0.020201
        1        0.100000        1.010050        0.020201        0.030605        0.030761        0.041632
        2        0.200000        1.040811        0.041632        0.053081        0.053368        0.065651
        3        0.300000        1.094174        0.065650        0.078890        0.079353        0.093882
        4        0.400000        1.173511        0.093881        0.109841        0.110559        0.128407
        5        0.500000        1.284025        0.128403        0.148305        0.149400        0.172011

 El punto aproximado y(x1) es = 1.2840252557
```

Conclusiones

Guzmán Claustro, Edgar

El método presentado anteriormente refiere a un método numérico que primero se desarrolló en excel, pues de esta manera es más sencillo ver su funcionamiento de una manera "manual". Posteriormente, se desarrolló en código utilizando matlab. Esto es interesante debido a que pone a prueba diversas aptitudes técnicas. Encontré la realización de este ejercicio interesante pues no conocía la existencia de dicho método numérico y programarlo me ayudó a comprenderlo de una mejor manera. Este tipo de automatizaciones son las que brindan mayor conocimiento al estudiante.

Cordero Hernández, Marco R.

Al igual que el ejercicio anterior, la revisión de este método contó con una dificultad no demasiado elevada; este hecho fue ayudado por la revisión pasada del método de Euler, puesto que el actual es tan solo una variación de aquella implementación.

Es bastante interesante encontrar múltiples métodos numéricos que van más allá de la resolución de sistemas de ecuaciones o derivadas, puesto que el campo de las ecuaciones diferenciales tiene diversas aplicaciones útiles para el día a día, y este tipo de ejercicios pueden lograr de momento un momento de reflexión acerca de lo que se ha aprendido, y en el futuro una posible aplicación con tiempos de ejecución bastante aceptables y resultados con buenas aproximaciones en todo tipo de campos de la aplicación de la ingeniería.

Bibliografía

Facultad de Ingeniería UNMdP. (2015). *Método de Euler*. Recuperado de http://www3.fi.mdp.edu.ar/metodos/apuntes/euler%20-%20Rodrigo.pdf.

Zheng, L., Zhang, X. (2017). *5.5 Runge–Kutta Methods*. Recuperado de https://www.sciencedirect.com/topics/mathematics/runge-kutta-method.

Illinois Institute of Technology. (s.f.). *Runge-Kutta Methods*. Recuperado de http://www.math.iit.edu/~fass/478578_Chapter_3.pdf.

Cheever, E. (2022). *Fourth Order Runge-Kutta*. Recuperado de https://lpsa.swarthmore.edu/NumInt/NumIntFourth.html.

The MathWorks Inc. (2023). MATLAB version: 9.14.0 (R2023a), Natick, Massachusetts: The MathWorks Inc. https://www.mathworks.com.