Ingeniería en Seguridad Informática y Redes

Ingeniería en Sistemas Computacionales

Algoritmos de Solución Numérica

Resolución de problema a través de Runge-Kutta

IS727272 - Marco Ricardo Cordero Hernández

SI727576 - Edgar Guzmán Claustro

Jal., 15 de noviembre de 2023

Índice

Contexto..... 1
Resultados..... 2
    Comprobación previa..... 2
    Código..... 2
    Resultados..... 3
Conclusiones..... 4
Bibliografía.....5

Contexto

Previamente, se ha hecho una implementación del método integral Runge-Kutta de cuarto orden (Cheever, 2022) para ecuaciones diferenciales indistintas a través del software Matlab (2023). En esta ocasión, se propone la alteración del código generado para su uso flexible como función para problemas generales.

El ejercicio seleccionado para esta demostración decreta lo siguiente:

> "En el estudio del flujo no isotérmico de un fluido newtoniano entre placas paralelas se encontró la ecuación
>
> $$\frac{d^2y}{dx^2} + x^2e^y = 0,\ x > 0$$
>
> Mediante una serie de sustituciones esta ecuación se puede transformar en la ecuación de primer orden
>
> $$\frac{dv}{du} = u(\frac{u}{2} + 1)v^3 + (u + \frac{5}{2})v^2$$
>
> Usando el método numérico Runge-Kutta de cuarto orden, aproximar $v(3)$ si $v(t)$ satisface $v(2) = 0.1$, con $h = 0.1$"

Como se puede apreciar, los datos de entrada f, x_0, y_0, x_1 y h son conocidos, por lo que su traducción hacía la herramienta previamente mencionada no debería suponer un problema mayúsculo. Esta implementación es presentada en la siguiente sección.

Resultados

Comprobación previa

Valores de referencia	
f(x,y)	u * (u/2 + 1) * v^3 + (u + 5/2) * v^2
x_0	2
$y(x_0)$	0.1
h	0.1
y(3)	?

Iteraciones					
x	**y**	**k1**	**k2**	**k3**	**k4**
2	0.1	0.0049	0.005222071	0.005239205	0.005596405
2.1	0.105236493	0.005596103	0.005989759	0.006012633	0.006453002
2.2	0.111245474	0.006452558	0.00694149	0.006972709	0.007525253
2.3	0.118213176	0.007524581	0.008143402	0.008187134	0.008895046
2.4	0.126393292	0.008893999	0.009695009	0.009758214	0.010688294
2.5	0.136141416	0.010686605	0.0117521	0.011847003	0.013107428
2.6	0.147973455	0.013104582	0.014570653	0.014720107	0.016496372
2.7	0.162670535	0.016491316	0.018597445	0.018847659	0.021482002
2.8	0.181481122	0.021472407	0.024675248	0.025129691	0.029318009
2.9	0.206547838	0.02929823	0.034567767	0.035492552	0.042855642
3	**0.241926923**	0.042810489	0.052548663	0.05477964	0.069947733

Código

Función Runge-Kutta

```
function y0 = RungeKutta(f, x0, y0, h, n, showIter)
%{
   f        = Función a evaluar
   x0       = Punto inicial x
   y0       = Punto inicial y(x0)
   h        = Cambio del método
   n        = Número de iteraciones
   showIter = Mostrar iteraciones (booleano)
%}

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

% Iterar sobre número de pasos definidos
k = [];
for i = 0:n
   % Formato condicional de tabla
   if (showIter); fprintf('\t%d\t\t%f\t%f', i, x0, y0); end

   % Cálculo de términos
   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));

    % Formato condicional de tabla
    if (showIter); for kn = k; fprintf('\t%f', kn); end; 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;

    % Formato condicional de tabla
    if (showIter); fprintf('\n'); end
end

% Formato condicional de tabla
if (showIter); fprintf('\n\n'); end
```

Manejador

```
% Restablecer entorno
clear, clc

% Datos de entrada
syms u v;                                       % Declaración de variables
f = u * (u/2 + 1) * v^3 + (u + 5/2) * v^2;      % Ecuación base
x0 = 2; x1 = 3; y0 = 0.1; h = 0.1;              % Valores conocidos
n = (x1 - x0)/h;                                % Despeje de pasos

% Invocación y cálculo del método con visualización de iteraciones
fprintf('Resultado = %10.10f\n', ...
    RungeKutta(inline(f), x0, y0, h, n, true));
```

Resultados

A comparación de la implementación anterior, en este caso los parámetros de entrada están predefinidos. La función adaptada tiene la opción adicional de mostrar el formato de tabla, por lo que una ejecución como la descrita se vería así:

Iteración	x	y	k1	k2	k3	k4
0	2.000000	0.100000	0.004900	0.005222	0.005239	0.005596
1	2.100000	0.105236	0.005596	0.005990	0.006013	0.006453
2	2.200000	0.111245	0.006453	0.006941	0.006973	0.007525
3	2.300000	0.118213	0.007525	0.008143	0.008187	0.008895
4	2.400000	0.126393	0.008894	0.009695	0.009758	0.010688
5	2.500000	0.136141	0.010687	0.011752	0.011847	0.013107
6	2.600000	0.147973	0.013105	0.014571	0.014720	0.016496
7	2.700000	0.162671	0.016491	0.018597	0.018848	0.021482
8	2.800000	0.181481	0.021472	0.024675	0.025130	0.029318
9	2.900000	0.206548	0.029298	0.034568	0.035493	0.042856
10	3.000000	0.241927	0.042810	0.052549	0.054780	0.069948

Resultado = 0.2419269226

Conclusiones

Guzmán Claustro, Edgar

Este ejercicio sirvió para reforzar el método de Runge-Kutta. Por otra parte, hacer uso del lenguaje de Matlab siempre es bueno, teniendo como añadido, realizar la solución al ejercicio presentado. De manera personal, creo que nunca es suficiente para dejar de aprender y siempre se puede generar nuevo conocimiento. Este tipo de experiencias son las que crean al profesional y marcan la diferencia.

Considero que el método numérico visto es interesante y se vuelve más interesante cuando se logran realizar aplicaciones funcionales y no solo queda el concepto teórico ni los típicos ejercicios "sencillos" de clase.

Cordero Hernández, Marco R.

Como ya se advertía en el ejercicio anterior donde se implementó por primera vez el método Runge-Kutta de cuarto orden, la aplicación del método hacía problemas que involucren ecuaciones diferenciales en entornos reales resulta útil para obtener resultados en tiempos minimizados. Las ejecuciones realizadas regresaban resultados casi de manera inmediata, y en un ambiente de alta concurrencia de datos esto puede resultar crítico e incluso un requerimiento de los sistemas a implementar.

En ediciones anteriores, se ha hecho uso de Matlab para demostrar problemas reales e implementaciones propuestas sin ninguna base, algunas con sustento para respaldarlo y algunas para atender problemas en un contexto real; en esta ocasión, al contar con un problema real (quizás como ejemplo típico de libro) con una resolución tangible a través del mismo software, el conocimiento de los métodos numéricos toma mayor importancia una vez que se ven en acción y la automatización por fin resulta útil más allá de los ejemplos revisados.

Bibliografía

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.