Desarrollo y análisis del rendimiento de la Programación Funcional para la solución de Ecuaciones Diferenciales*

Yezid Donoso Meisel**, Carlos Ardila Hernández***

Resumen

En este artículo se presenta el desarrollo de soluciones de métodos numéricos aplicados a las ecuaciones diferenciales en ingeniería mediante de la programación funcional, cuyo objetivo es precisamente dar una solución robusta, eficiente y práctica a aquellas aplicaciones cuya fundamentación sean funciones matemáticas.
Palabras claves: Programación funcional, métodos numéricos, Runge-Kutta.

Abstract

In this article we will present the develop of numeric methods solutions apply to diferential equations in engineering thue the funtional programming which objetive is give a eficient, practics and great solution to those application whose bases matemathical function.
Key words: Funtional programming, métodos numéricos, Runge-Kutta.

Fecha de recepción: 13 de marzo de 2000

1. INTRODUCCIÓN

Después de conocer el desarrollo de programas mediante Lenguajes Funcionales, se empieza a visualizar el abanico de posibilidades en las que se puede aplicar este paradigma de programación. Una de esas posibilidades es el desarrollo de aplicaciones para métodos numéricos en ingeniería, tales como: raíces de ecuaciones, integración y diferenciación numérica, mínimos cuadrados, ecuaciones diferenciales ordinarias y parciales, elementos finitos, etc., y sus respectivas aplicaciones en el diario vivir. El objetivo de este artículo es el de aplicar este paradigma de programación funcional en la solución de ecuaciones diferenciales ordinarias mediante el método de Runge-Kutta de cuarto

* Al momento de presentar este artículo a la revista de *Ingeniería & Desarrollo* se encuentra en proceso de validación por el Congreso Internacional ICIEY2K en Argentina.

** Ingeniero de Sistemas de la Universidad del Norte; Magister en Ingeniería de Sistemas; profesor e investigador del Departamento de Sistemas de la Universidad del Norte. *(e-mail:ydonoso@uninorte.edu.co)*

*** Ingeniero de Sistemas de la Universidad del Norte. Actualmente cursa el Magister en Matemáticas en la Universidad del Norte. Profesor e investigador del Departamento de Sistemas de la misma universidad. *(e-mail:cardila@uninorte.edu.co)*

orden, y además exponer una metodología para el desarrollo de estas aplicaciones.

2. INTRODUCCIÓN Y UNA METODOLOGÍA PARA LA *PROGRAMACIÓN FUNCIONAL*

La programación funcional nació como una proyección para realizar programas, los cuales pretenden cubrir el campo de desarrollo de aplicaciones cuya solución se soporte o fundamente en teorías o definiciones matemáticas. La fundamentación de este estilo de programación se encuentra en el Cálculo Lambda. En esta sección se explicará en forma general y básica cómo se realiza un programa mediante un lenguaje funcional.

Los pasos necesarios básicos para realizar esta programación los podemos resumir de la siguiente manera:

- Conceptualizar el problema (comprender el problema y su entorno).
- Analizar las entradas y las salidas.
- Establecer las funciones matemáticas que solucionen el problema.
- Realizar el programa mediante un lenguaje funcional según las especificaciones del paso anterior.

A continuación se presenta un ejemplo:

Supongamos que necesitamos realizar un programa que calcule el factorial de un número, entonces apliquemos la metodología mencionada anteriormente así:

- *Conceptualizar el problema*

- Calcular el factorial de un número dado, el cual se basa en la multiplicación de una secuencia de números desde el *1* hasta *n* de la siguiente forma: $1 * 2 * 3 * (n-1) * n = n!$

- *Analizar las entradas y las salidas*

  - Entrada
  - n , n e z
  - Salida
  - $n!$, $n!$ e z

- *Establecer las funciones matemáticas que solucionen el problema*

- Definición de la función factorial

$$n! = \begin{cases} 1, & \text{si } n = 0 \quad \text{Condición de Salida} \\ n * (n-1)! & \text{Si } n > 0 \quad \text{Condición de Recursividad} \end{cases}$$

- *Realizar el programa mediante un lenguaje funcional*

- **Programa 1.** *Factorial de un número*

```
let rec factorial n =
    if ( n = 0) then 1
    else n * (factorial (n-1));
```

Explicación en detalle del programa:

- Let: Es la instrucción básica de definición.
- rec: Indica que la función es recursiva.
- factorial: Nombre de la función.
- if ... then ... else: Instrucción condicional.
- Si el valor de *n (entrada)* es igual a cero, entonces devuelve 1.
- Si el valor de *n (entrada)* es diferente de cero, la función se llamará nuevamente con el valor restándole 1, así: n * factorial(n-1).

3. APLICACIÓN DE LA PROGRAMACIÓN FUNCIONAL EN LA *SOLUCIÓN DE ECUACIONES DIFERENCIALES*

En este caso se analizará la solución de ecuaciones diferenciales mediante el método de Runge-Kutta de cuarto orden. Ahora, este estilo de programación funcional es utilizado también en otros tipos de soluciones como:

- Métodos para encontrar raíces en ecuaciones
- Regresión con mínimos cuadrados
- Integración numérica
- Ecuaciones diferenciales ordinales
- Ecuaciones diferenciales parciales
- Método de elementos finitos
- Todas las aplicaciones prácticas de los métodos numéricos en la ingeniería.

A continuación se presenta el método de Runge-Kutta implementándolo mediante la programación funcional.

La forma general de la ecuación para el método de Runge-Kutta de cuarto orden es:

$$y_{i+1} = y_i + f(x_i, y_i, h)h$$

donde $f(x_i, y_i, h)$ se le llama función de incremento, y puede interpretarse como el promedio de la pendiente sobre el intervalo. La función de incremento se puede escribir en la forma general como:

$$f = a_1k_1 + a_2k_2 + ... + a_nk_n$$

donde a_i son constantes.

Para dar solución a través del método de Runge-Kutta se realiza el siguiente cálculo dado por la ecuación:

$$y_{i+1} = y_i + [1/6(k_1 + 2k_2 + 2k_3 + k_4)]h$$

donde

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + {}^1/_2h,\ y_i + {}^1/_2k_1h)$$

$$k_3 = f(x_i + {}^1/_2h,\ y_i + {}^1/_2k_2h)$$

$$k_4 = f(x_i + h,\ y_i + k_3h)$$

La aplicación desarrollada mediante Programación Funcional es la siguiente:

- **Programa 2.** *Método de Runge-Kutta de cuarto orden por Programación Funcional*

```
let rec runge_kutta f xi xf h x0 y0 =
  if (xi >= xf) then [y0]
 else
  y0::(runge_kutta f (xi +. h) xf h x0 yi)
 where yi = y0 +. (((1.0 /. 6.0) *. (k1 +. (2.0
 *. k2) +. (2.0 *. k3) +. k4)) *. h)
 where k1 = (f xi)
 and k2 = (f (xi +. ((1.0 /. 2.0) *. h)))
 and k3 = (f (xi +. ((1.0 /. 2.0) *. h)))
 and k4 = (f (xi +. h));
```

donde:

f : *Función que se va a evaluar*
xi : *Valor inicial del intervalo en x*
xf : *Valor final del intervalo en x*
paso: *Incremento*
x0 : *Condición inicial de x*
y0 : *Condición inicial de y*

Este programa funcional se ejecutó con la siguiente ecuación diferencial de primer orden:

$$f(x) = -2x^3 + 12x^2 - 20x + 8.5$$

Los resultados de ejecutar el programa anterior con un h = 0.5 son:

```
#let f1 = (fun x -> (12.0 *. x *. x) -. (2.0 *.
x *. x*. x) -. (20.0 *. x) +. 8.5);

#runge_kutta  f1  0.0  4.0  0.5  0.0  1.0;

#- : float list =
 [1.0; 3.21875; 3.0; 2.21875; 2.0; 2.71875;
4.0; 4.71875; 3.0]
```

Tabla 1
Análisis de los valores de Runge-Kutta con h=0.5

X	Y_{real}	$Y_{h=0.5}$
0.0	1.000000	1.000000
0.5	3.21875	3.21875
1.0	3.00000	3.00000
1.5	2.21875	2.21875
2.0	2.00000	2.00000
2.5	2.71875	2.71875
3.0	4.000000	4.000000
3.5	4.71875	4.71875
4.0	3.00000	3.00000

Como se puede observar, el programa funcional que da solución al método numérico se deduce en forma directa de la ecuación matemática de recurrencia, cumpliendo así con la metodología planteada para el desarrollo de aplicaciones funcionales.

4. ANÁLISIS COMPARATIVO DEL RENDIMIENTO CON OTROS PARADIGMAS DE PROGRAMACIÓN

Tomando como referencia el porcentaje de uso del procesador para la ejecución de un proceso y analizando el programa de Runge-Kutta realizado mediante programación funcional en Caml Ligth, se comparó con el mismo método numérico desarrollado en forma recursiva mediante programación imperativa y en programación orientada a objetos implementandolos en C++. Las pruebas se realizaron en un equipo servidor con sistema operativo Windows NT sin clientes conectados con un procesador

Pentium de 300MHz con 64M de memoria RAM.

Los resultados observados del porcentaje de uso del procesador para diferentes valores de los intervalos de tiempo (h) y bajo los diferentes paradigmas de programación fueron los siguientes:

Tabla 2
Análisis del Rendimiento

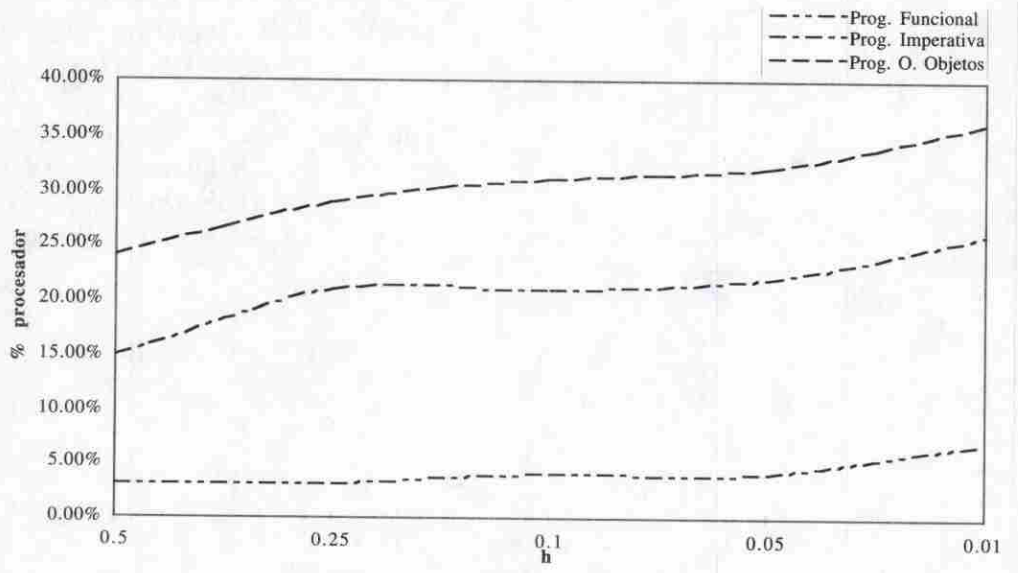
Intervalo (h) (tiempo)	**Prog. Func.** (%)	**Prog. Imper.** (%)	**Prog. O. Obj.** (%)
0.5	3.01	15.0	24.0
0.25	3.01	21.01	29.01
0.1	4.01	21.01	30.99
0.05	4.01	22.00	32.00
0.01	7.0	26.01	36.01

Como se puede observar en la gráfica de abajo, la programación funcional en este caso presenta un mejor rendimiento en el uso del procesador que los otros dos paradigmas (imperativo y objetos). El objetivo será seguir explorando la utilización y ventajas de rendimiento que pueda presentar este paradigma de programación en la implementación de los algoritmos presentados en los diferentes tópicos del análisis numérico y en otros campos de las matemáticas aplicadas a la ingeniería.

CONCLUSIONES

Se puede sacar como conclusiones lo siguiente:

- La gran importancia que tiene el desarrollo de programas mediante len-



Análisis Comparativo del Rendimiento

guajes funcionales debido a la cercanía con la fundamentación y definición matemática del mismo problema.

- El gran campo por explorar de las aplicaciones en ingenierías, donde se podría utilizar la Programación Funcional para dar soluciones más concretas, eficientes y eficaces.
- Lo fácil que sería demostrar que un programa es correcto, ya que la Programación Funcional se basa en funciones matemáticas, y si éstas son correctas, el programa directamente es correcto.
- El mejor rendimiento a nivel del uso del procesador que presentan las aplicaciones desarrolladas mediante programación funcional en el caso de ecuaciones diferenciales.

Referencias

Programación Funcional

http://pauillac.inria.fr/caml/

[Leroy97] LEROY, Xavier. *The Objective Caml System release 1.07.* Inria (Institut National de Recherche en Informatique et Automatique), 1997.

[Mauny95] MAUNY, Michel. *Functional programming using Caml Light.* Inria, 1995.

[Bird86] Science *Introduction to Functional Programming. Series in Computer Science.* Prentice-Hall, 1986.

Análisis Numérico

[Burden98] Thompon BURDEN, Richard y FAIRES, Douglas. *Análisis numérico.* 6ª ed., 1998.

[Chapra99] CHAPRA, Steven y CANALE, Raymond. *Métodos numéricos para ingenieros.* 3ª ed. McGraw-Hill, 1999.