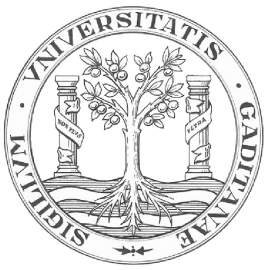
UNIVERSIDAD DE CÁDIZ

GRADO EN MATEMÁTICAS



RESOLUCIÓN NUMÉRICA DE ECUACIONES DIFERENCIALES DIFUSAS

TRABAJO FIN DE GRADO

CURSO ACADÉMICO: 2017/2018

JOSÉ CARLOS GARCÍA ORTEGA

Dr. Rafael Rodríguez Galván

Dr. Jesús Medina Moreno

Universidad de Cádiz

Grado en Matemáticas

Resolución numérica de ecuaciones diferenciales difusas

Trabajo fin de grado
Curso académico: 2017/2018

José Carlos García Ortega

Dr. Rafael Rodríguez Galván Dr. Jesús Medina Moreno

Firma del alumno Firma del tutor Firma del tutor

Jerez de la Frontera, Cádiz, septiembre 2018

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Resumen

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Pa mi

ÍNDICE GENERAL

SOFTWARE

En este trabajo se ha hecho bastante hincapié en utilizar software libre para su desarrollo, el software que se ha utilizado para generar esta memoria y la presentación:

- LaTeX
- *Matplotlib*; Una librería en Python para generar gráficos.
- *SlideFramework*; Un software para crear presentación en HTML5 creado por mi mismo.
- *CUDA*; Un lenguaje de programación de NVIDIA ® para cálculos en GPU.

NOTACIÓN

$\mathbb{U}$	$\triangleq$	Conjunto universo
$cl\, A$	$\triangleq$	Clausura de A
$\mathcal{F}_{\mathcal{C}}(\mathbb{U})$	$\triangleq$	La familia de conjuntos de $\mathbb{U}$ tales que sus α-cortes son no vacios, compactos en $\mathbb{U}$
$\mathcal{F}_{\mathcal{H}}(\mathbb{U})$	$\triangleq$	La familia de conjuntos de $\mathbb{U}$ tales que sus α-cortes son no vacios, compactos y convexos en $\mathbb{U}$
$\ominus_H$	$\triangleq$	Diferencia de Hukuhara
$(a; b; c)$	$\triangleq$	Número triangular

CAPÍTULO 1

CONJUNTOS DIFUSOS

El contenido de este capítulo se basa en las definiciones que podemos encontrar en la referencia [1]

1.1. Subconjuntos difusos

La teoría clásica de conjuntos sólo abarca la posibilidad de que un elemento pertenezca o no, a un conjunto. Pero la realidad no es así, y pueden existir ciertos casos en lo que la **pertenencia** o no, a un conjunto haya que definirlo mediante un **grado de pertenencia.**

1.1.1. Subconjuntos difusos

En primer lugar, introduciremos el concepto de conjunto difusos que nos servirá para formalizar el concepto de conjuntos con grados de pertenencia.

Definición 1 (Subconjunto difuso)**.** *Un* ***subconjunto difuso*** A *es un par ordenado* $(\mu_A, \mathbb{U})$ *con:*

$$\mu_A : \mathbb{U} \longrightarrow [0, 1]$$

Denominamos μ_A ***función de pertenencia.***

Tenemos que notar que esta función de pertenencia no define necesariamente una probabilidad, y no hace referencia por ejemplo a la probabilidad de que una persona sea alta o baja, si no que nos da una medida de cuanto de alta o baja es.

Por tanto, es natural definir la igualdad de conjuntos de la siguiente forma:

Definición 2 (Igualdad de conjuntos difusos)**.** *Decimos que dos conjuntos difusos* A *y* B *en* $\mathbb{U}$ *son iguales si para todo* $x \in \mathbb{U}$*, se cumple* $\mu_A(x) = \mu_B(x)$

Si se que cumpliese que $\mu_A(\mathbb{U}) = \{0, 1\}$ **tendríamos que** A **es un conjunto clásico**, y llamamos a μ_A función característica de A. En este caso, tenemos que si $x \in A$, entonces $\mu_A(x) = 1$, y por el contrario si $x \notin A$ entonces, $\mu_A(x) = 0$
Por tanto, **la función** μ_A **representa una generalización del concepto de función característica clásica** dónde μ_A representa el grado de pertenencia a un conjunto.

1.1.2. α-corte

Ahora introducimos un concepto fundamental en la teoría de conjuntos difusos, y es el concepto de α-corte, que nos permitirá crear particiones de los conjuntos separados por los valores de la función de pertenencia.

Definición 3 (α-corte)**.** *Dado un conjunto difuso A, los α-corte son los subconjuntos clásicos dados por:*

$$[A]_\alpha = \begin{cases} \{x \in \mathbb{U} : \mu_A(x) \geq \alpha\} & si \quad \alpha \in (0, 1] \\ cl\{x \in \mathbb{U} : \mu_A(x) > 0\} & si \quad \alpha = 0 \end{cases}$$

Además, se define:

$$soporte\, A = \{x \in \mathbb{U} : \mu_A(x) > 0\}$$
$$nucleo\, A = \{x \in \mathbb{U} : \mu_A(x) = 1\}$$

De esta definición, se puede extraer que un conjunto difuso también puede estar definido por sus α-corte, de manera que **dos conjuntos difusos** A **y** B **son iguales, si todos sus** α**-corte son iguales.**
Dado unos α-corte podemos construir una función de pertenencia de la siguiente forma: [2]

$$\mu_A = \text{máx}\{\alpha A_\alpha(x) : \alpha \in [0, 1]\}$$
$$A_\alpha(x) = \begin{cases} 1 & si \quad x \in [A]_\alpha \\ 0 & si \quad x \notin [A]_\alpha \end{cases}$$

Desde aquí, centraremos nuestro estudio en los α-corte de los conjuntos difusos.

1.2. Números difusos

Para poder trabajar con sistemas de ecuaciones diferenciales difusos, es interesante introducir el concepto de números difusos, que podrían ser valores iniciales de nuestro sistema, o números que aparecen directamente en nuestro sistema de ecuaciones diferenciales.
Necesitamos en primer lugar dos definiciones antes de definir el concepto de número difuso.

Definición 4 (Conjunto difuso normal)**.** *Un conjunto difuso A decimos que es normal si $nucleo\, A \neq \emptyset$*

Definición 5 (Conjunto difuso convexo)**.** *Dado un conjunto difuso A decimos que es convexo si su función de pertenencia es cuasicóncava, esto es:*

$$\mu_A(\lambda x + (1 - \lambda)y) \geq \text{mín}\{\mu_A(x), \mu_A(y)\}, \lambda \in [0, 1], x, y \in \mathbb{U}$$

En este caso, si $\mathbb{U} = \mathbb{R}$ tendríamos que si A es un conjunto difuso convexo, entonces los α-corte son intervalos.
Finalmente, introducimos el concepto de número difuso:

Definición 6 (Número difuso)**.** *Decimos que A es un número difuso si $\mathbb{U} = \mathbb{R}$, A es normal y convexo, y además su función de pertenencia es continua por la derecha.*

Observamos que si $x \in \mathbb{R}$ el conjunto $\{x\}$ es un número difuso, con la función de pertenencia.

1.2.1. Caracterización números difusos

A continuación, introducimos dos teoremas que nos permitirán caracterizar los números difusos. Las demostraciones de estos dos resultados se pueden encontrar [2].

Teorema 1 (Teorema de Stacking)**.** *Sea A un número difuso, entonces:*

1. *Sus α-cortes son intervalos cerrados no vacíos para todo $\alpha \in [0,1]$*
2. *Si $0 \leq \alpha_1 \leq \alpha_2 \leq 1$ entonces $[A]_{\alpha_1} \subset [A]_{\alpha_2}$*
3. *Para toda sucesión no decreciente de $\alpha_n \in [0,1]$ tal que $\alpha_n \longrightarrow \alpha$ se tiene que:*

$$\bigcap_{n=1}^{\infty} [A]_{\alpha_n} = [A]_{\alpha}$$

4. *Para toda sucesión no creciente $\alpha_n \in [0,1]$ convergente a 0 se tiene:*

$$cl\left(\bigcup_{n=1}^{\infty} [A]_{\alpha_n} = [A]_0\right)$$

Teorema 2 (Teorema de caracterización)**.** *Sea $A = \{A_\alpha : \alpha \in [0,1]\}$ una familia de subconjuntos de $\mathbb{R}$ tal que:*

1. *Sus α-cortes son intervalos cerrados no vacíos para todo $\alpha \in [0,1]$*
2. *Si $0 \leq \alpha_1 \leq \alpha_2 \leq 1$ entonces $[A]_{\alpha_1} \subset [A]_{\alpha_2}$*
3. *Para toda sucesión no decreciente $\alpha_n \in [0,1]$ tal que $\alpha_n \longrightarrow \alpha$ se tiene que*

$$\bigcap_{n=1}^{\infty} [A]_{\alpha_n} = [A]_{\alpha}$$

4. *Para toda sucesión no creciente $\alpha_n \in [0,1]$ convergente a 0 se tiene:*

$$cl\left(\bigcup_{n=1}^{\infty} [A]_{\alpha_n}\right) = [A]_0$$

Entonces, A es un número difuso.

Ejemplo 1. *Sea $\mathbb{U} = \mathbb{R}$ y consideremos $\mu_{\mathcal{A}} : \mathbb{R} \longrightarrow [0,1]$ definida de la siguiente forma:*

$$\mu_{\mathcal{A}}(x) = \begin{cases} \frac{x-a}{b-a} & si \quad x \in [a,b] \\ \frac{c-x}{c-b} & si \quad x \in (b,c] \\ 0, & si \quad x \notin [a,c] \end{cases}$$

*Con $a < b < c$. Decimos que **un número difuso definido de la forma anterior es triangular (a; b; c).***

1.3. Principio de extensión de Zadeh

Nos gustaría ahora que dado un conjunto difuso, y una función clásica, pudiéramos obtener una función difusa, para esto, tenemos el principio de extensión de Zadeh.

Definición 7 (Principio de extensión de Zadeh)**.** *Sean $\mathbb{U}$ y $\mathbb{V}$ dos conjuntos de universos, y sea $f : \mathbb{U} \longrightarrow \mathbb{V}$ una función clásica. Definimos el principio de extensión de Zadeh para todo conjunto difuso A en $\mathbb{U}$ con μ_A su función de pertenencia, de manera que $\hat{f}(A)$ en $\mathbb{V}$ y su función de pertenencia viene dada por:*

$$\mu_{\hat{f}(A)} = \begin{cases} \sup_{x \in f^{-1}(y)} \mu_A(x) & si \quad f^{-1}(y) \neq \emptyset \\ 0 & si \quad f^{-1} = \emptyset \end{cases}$$

Notemos, que si la función es inyectiva, la función de pertenencia se simplificaría:

$$\mu_{\hat{f}(A)} = \begin{cases} \mu_A(f^{-1}(y)) & si \quad f^{-1}(y) \neq \emptyset \\ 0 & si \quad f^{-1} = \emptyset \end{cases}$$

En el apéndice, se pueden encontrar ejemplos de funciones de pertenencias dadas por el principio de extensión de Zadeh.

1.3.1. Teoremas de continuidad

Sería ideal que no importase el orden el que obtenemos los α-corte de la imagen de una función mediante el principio de extensión de Zadeh, y esto lo asegura el siguiente teorema.

Teorema 3. *Sea $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ una función.*

1. *Si f es sobreyectiva, entonces $[\hat{f}(A)]_\alpha = f([A]_\alpha)$ si y sólo si $\sup\{\mu_A(x) : x \in f^{-1}(y)\}$ para todo $y \in \mathbb{R}^m$*
2. *Si f es continua, entonces $\hat{f} : \mathcal{F}_{\mathcal{H}}(\mathbb{R}^n) \longrightarrow \mathcal{F}_{\mathcal{H}}(\mathbb{R}^n)$ está bien definido, y además,*

   $$[\hat{f}(A)]_\alpha = f([A]_\alpha)$$

   para todo $\alpha \in [0,1]$

La primera implicación es clara por la definición de principio de extensión de Zadeh, para la segunda implicación veamos un caso más general:

Teorema 4. *Sean $\mathbb{U}$ y $\mathbb{V}$ unos espacios de Haussdorf, y sea $f : \mathbb{U} \longrightarrow \mathbb{V}$ una función. Si f es continua, entonces $\hat{f} : \mathcal{F}_{\mathcal{H}}(\mathbb{R}^n) \longrightarrow \mathcal{F}_{\mathcal{H}}(\mathbb{R}^n)$ está bien definida y*

$$[\hat{f}(A)]_\alpha = f([A]_\alpha)$$

para todo $\alpha \in [0, 1]$

Demostración. Por la definición del principio de Zadeh, tenemos que $\hat{f}(A)$ es un subconjunto difuso de $\mathbb{V}$.
Para probar que $\hat{f} : \mathcal{F}_{\mathcal{H}}(\mathbb{R}^n) \longrightarrow \mathcal{F}_{\mathcal{H}}(\mathbb{R}^n)$ hay que ver que todos los $\alpha - corte\, [\hat{f}(A)]_\alpha$ son no vacíos, compactos en $\mathbb{V}$.
Dado que f es continua por hipótesis, la imagen de compactos, son compactos, por tanto, sólo debemos probar $[\hat{f}(A)]_\alpha = f([A]_\alpha)$.
Probémoslo por doble inclusión.

- $[\hat{f}(A)]_\alpha \subset f([A]_\alpha)$. Sea $\mathcal{A} \in \mathcal{F}_{\mathcal{H}}(\mathbb{U})$ y $y \in f([A]_\alpha)$. Por tanto, existe al menos un $x \in [A]_\alpha$ tal que $f(x) = y$. Por el principio de extensión de Zadeh, tenemos $\mu_{\hat{f}(A)}(y) = \sup_{x \in f^{-1}(y)} \mu_A(x) \geq \alpha$. De donde, $y \in [\hat{f}(\mathcal{A})_\alpha]$

- Por otro lado, veamos que $[\hat{f}(A)]_\alpha \supset f([A]_\alpha)$. Dado que $\mathbb{V}$ y $\mathbb{U}$ son espacios de Haussdorf, un punto $y \in \mathbb{V}$ es cerrado. Y además, dado que f es continua, $f^{-1}(y)$ es cerrado. Dado que $[A]_0$ es compacto, ya que es la clausura, la intersección de compactos también es compactos, por tanto $f^{-1}(y) \cap [A]_0$ es compacto. Para $\alpha > 0$, consideramos $y \in [\hat{f}(A)]_\alpha$. Entonces $\mu_{\hat{f}(A)}(y) = \sup_{x \in f^{-1}(y)} \mu_A(x) \geq \alpha > 0$, y además, existe un $x \in f^{-1}(y)$ tal que $f^{-1}(y) \cap [A]_0 \neq \emptyset$

Finalmente, debido a que $\mu_A(x)$ es continúa por la derecha, y $f^{-1}(y) \cap [A]_0$ es compacto, existe un $x \in f^{-1}(y) \cap [A]_0$ con $\mu_{\hat{f}(\mathcal{A})}(y) = \mu_{\mathcal{A}}(x) \geq \alpha$. Esto es porque $y = f(x)$ para algún $x \in [A]_\alpha$.
Para $\alpha = 0$, obtenemos:

$$\bigcup_{\alpha \in (0,1]} [\hat{f}(\mathcal{A})]_\alpha = \bigcup_{\alpha \in (0,1]} f([\mathcal{A}]_\alpha) \subset f([\mathcal{A}]_0).$$

Dado que $f([\mathcal{A}]_0)$ es cerrado:

$$[\hat{f}(\mathcal{A})]_0 = cl\left(\bigcup_{\alpha \in (0,1]} [\hat{f}(\mathcal{A})]_\alpha\right) = cl\left(\bigcup_{\alpha \in (0,1]} f([\mathcal{A}]_\alpha)\right) \subset f([\mathcal{A}]_0).$$

Y por la doble inclusión anterior, tenemos que $[\hat{f}(\mathcal{A})]_\alpha \subset f([\mathcal{A}]_\alpha)$ para todo $\alpha \in [0, 1]$ □

1.4. Aritmética difusa

El siguiente paso **para poder construir métodos numéricos es necesario definir las operaciones aritméticas básicas** entre conjuntos difusos.
La definiciones de **estas operaciones son bastante naturales**, pero pueden ocasionarnos algunos problemas en ciertos escenarios.
Debido a la equivalencia entre trabajar con conjuntos difusos, y sus $\alpha - corte$, daremos todas las operaciones en términos de $\alpha - cortes$.
En primer lugar, **recordemos la definición habitual de aritmética en teoría de conjuntos clásicos.**

1.4.1. Aritmética en conjuntos clásicos

Sean A, B dos conjuntos entonces:

- $A + B = \{a + b : a \in A, b \in B\}$
- $A - B = \{a - b : a \in A, b \in B\}$
- $A * B = \{ab : a \in A, b \in B\}$
- $A/B = \{a/b : a \in A, b \in B\}$

Recordados los conceptos básicos de aritmética en conjuntos clásicos, pasamos a generalizarlo para conjuntos difusos.

1.4.2. Aritmética en conjuntos difusos

Sean μ_A, μ_B dos funciones de pertenencia y sea $\odot \in \{+, -, \cdot, \div\}$ se define la función de pertenencia de la operación aritmética como:

$$\mu_{A \odot B}(x) = \sup_{a \odot b = c} \min\{\mu_A(a), \mu_B(b)\}$$

Y dado que **las operaciones aritméticas son funciones continuas**, es equivalente trabajar con los $\alpha - corte$**, aplicando el principio de extensión de Zadeh tenemos:**
Sean A y B dos números difusos con $\alpha - corte$ dados por $[A]_\alpha = [a_\alpha^-, a_\alpha^+]$ y $[B]_\alpha = [b_\alpha^-, b_\alpha^+]$, podemos definir entonces las operaciones aritméticas como:

$$[A + B]_\alpha = [a_\alpha^- + b_\alpha^-, a_\alpha^+ + b_\alpha^+]$$

$$[A - B]_\alpha = [a_\alpha^- - a_\alpha^+, a_\alpha^+ - b_\alpha^-]$$

$$[A \cdot B]_\alpha = \left[\min_{s,r \in \{-,+\}} a_\alpha^s \cdot b_\alpha^r, \max_{s,r \in \{-,+\}} a_\alpha^s \cdot b_\alpha^r \right]$$

$$[A \div B]_\alpha = \left[\min_{s,r \in \{-,+\}} \frac{a_\alpha^s}{b_\alpha^r}, \max_{s,r \in \{-,+\}} \frac{a_\alpha^s}{b_\alpha^r} \right]$$

Problemas al definir estas operaciones aritméticas

Nuestro objetivo final es **definir la diferencial de una función**, para funciones escalares de una sola variable podemos definir la diferencial como:

$$\lim_{h\to 0^+} \frac{f(x+h)-f(x)}{h}$$

Consideremos ahora una función $f(x) = A \in \mathcal{F}_{\mathcal{H}}(\mathbb{U})$, donde A es un número difuso constante.

$$f(x+h) - f(x) = [A - A]$$

Sean $[A]_\alpha$ los $\alpha - cortes$ de A entonces:

$$[A-A]_\alpha = [a_\alpha^- - a_\alpha^+, a_\alpha^+ - a_\alpha^-]$$

Si $A \neq 0$ tenemos que $[A-A]_\alpha \neq 0$, por tanto al dividir por $h \longrightarrow 0$, tenemos una indeterminación, de donde, tal y **como hemos definido las operaciones aritméticas para los conjuntos difusos no tendríamos definidas las derivadas de funciones constantes**, y esto, es un problema.
Necesitamos definir **un nuevo concepto de diferencia**, que para ello, **utilizaremos la diferencia de Hukuhara.**

1.4.3. Hukuhara y diferencia generalizada

Debido al problema especificado en la sección anterior, **necesitamos definir una operación resta que cumpla que** $A - A = \{0\}$, Hukuhara dio una definición de resta que cumplía esta propiedad.

Definición 8. *Dados dos números difusos $A, B \in \mathcal{F}_{\mathcal{C}}\mathbb{R}$ la **diferencia de Hukuhara (H-Diferencia)** se define como $A \ominus_H B = C$ donde C es el número difuso que cumple $A = B + C$, si existe.*

Observación 1. *Sean $A, B, C \in \mathcal{F}_{\mathcal{C}}\mathbb{R}$ consideremos sus $\alpha - cortes$, por tanto $[A]_\alpha = [a_\alpha^-, a_\alpha^+]$, $[B]_\alpha = [b_\alpha^-, b_\alpha^+]$ y $[C]_\alpha = [c_\alpha^-, c_\alpha^+]$.*
De donde,

$$[a_\alpha^-, a_\alpha^+] = [b_\alpha^- + c_\alpha^-, b_\alpha^+ + c_\alpha^+]$$

Por tanto,

$$[A \ominus_H B]_\alpha = [a_\alpha^- - b_\alpha^-, a_\alpha^+ - b_\alpha^+]$$

Con esta observación es fácil ver que la H-Diferencia cumple que $A - A = \{0\}$

1.5. Interactividad

El principio de extensión de Zadeh se puede aplicar a funciones de distinto número de argumentos. Los ejemplos más simples podrían ser la suma, la resta, la multiplicación y la división de números difusos. Esta situación es más compleja, debido a que tenemos que tener en cuenta las estructuras entre los distintos argumentos. Esta dependencia mutua entre los distintos conjuntos difusos, viene dada por una función de pertenencia común que llamaremos **función de pertenencia conjunta**. En términos de conjuntos difusos, esta dependencia se llama interactividad.

Definición 9 (Interactividad, función de pertenencia conjunta)**.** *Sea $\hat{a} \in \mathcal{F}(V)$ y sea $\hat{b} \in \mathcal{F}(W)$. Entonces la interactividad de $\hat{a}$ y $\hat{b}$ se define por la función de pertenencia conjunta:* $\mu_{\hat{a},\hat{b}} : V \times W \to [0,1]$

Para calcular las funciones de pertenencia marginales, simplemente tenemos que aplicar el principio de extensión de Zadeh:

Definición 10 (Función marginal de una función de pertencia conjunta)**.** *Definimos la función de pertenencia marginal respecto a como:*

$$\mu_a(a) = \sup \lim_{b \in W} \mu_{\hat{a},\hat{b}}(a,b)$$

Se suele suponer no interactividad al trabajar con varios conjuntos difusos, esto es;

Definición 11 (No interactivos o independientes)**.** *Dos conjuntos difusos $\hat{a} \in \mathcal{F}(V)$ y $\hat{b} \in \mathcal{F}(W)$ se dicen que son no interactivos, o independientes si* $\mu_{\hat{a},\hat{b}} = \min(\mu_{\hat{a}}(a), \mu_{\hat{b}}(b))$

Ejemplo 2. *En el caso de conjuntos no interactivos, podemos escribir las operaciones aritméticas de los conjuntos difusos de la siguiente manera:*

$$[\hat{a} + \hat{b}]_\alpha = [[\hat{a}_\alpha^- + [\hat{b}]_\alpha^-, [\hat{a}]_\alpha^+ + [\hat{b}]_\alpha^+]$$

$$[\hat{a} - \hat{b}]_\alpha = [[\hat{a}_\alpha^- - [\hat{b}]_\alpha^-, [\hat{a}]_\alpha^+ - [\hat{b}]_\alpha^+]$$

$$[\hat{a} * \hat{b}]_\alpha = \max\{[\hat{a}]_\alpha^i [\hat{b}]_\alpha^j\}, i, j \in \{+,-\}$$

$$[\hat{a}/\hat{b}]_\alpha = \max\{[\hat{a}]_\alpha^i / [\hat{b}]_\alpha^j\}, i, j \in \{+,-\}$$

1.6. Métrica en conjuntos difusos

En esta sección vamos a generalizar la definición de espacio métrico a conjuntos difusos, y vamos a dar algunos resultados importantes. Todas las demostraciones de esta sección pueden encontrarse en [2].
En primer lugar, nos acercaremos al concepto de métrica:

Definición 12 (Pseudométrica)**.** *Sean A y B dos subconjuntos de un espacio métrico $\mathbb{U}$ compactos. Entonces definimos la pseudométrica como:*

$$\rho(A, B) = \sup_{a \in A} d(a, B)$$

Donde:

$$d(a, B) = \inf_{b \in B} ||a - b||$$

es la separación de Haussdorf

Definición 13 (Métrica de Pompeiu-Hausdorff)**.** *Sean A y B dos conjuntos difusos en $\mathbb{U}$, un espacio métrico. La métrica de Pompeiu-Hausdorff, denotada por d_∞ se define:*

$$d_\infty(A, B) = \sup_{\alpha \in [0,1]} \text{máx}\{\rho([A]_\alpha, [B]_\alpha), \rho([B]_\alpha, [A]_\alpha)\}$$

Si A y B fueran números difusos, tendríamos:

$$d_\infty(A, B) = \sup_{\alpha \in [0,1]} \text{máx}\{|a_\alpha^- - b_\alpha^-|, |a_\alpha^+ - b_\alpha^+|\}$$

Teorema 5 ([3])**.** *El espacio de los números difusos, con la métrica d_∞ es un espacio de Banach.*

1.7. Funciones difusas

CAPÍTULO 2

CÁLCULO DIFUSO

En este capítulo exploramos los diferentes acercamientos a **conceptos clásicos del cálculo mediante una perspectiva difusa**. Exploramos la definición de integral de Riemann, Aumann & Henstock, y haremos una revisión de la definición de derivada de Hukuhara (Diferencia de Hukuhara). Además, hablaremos de la **versión difusa del teorema fundamental del cálculo** que nos ayudará a tener una mejor visión de lo que está pasando.

2.1. Cálculo difuso para funciones definidas en conjuntos difusos

En esta sección nos centraremos en aquellas funciones con dominios reales, y que su imagen son conjuntos difusos, es decir,

Definición 14 (Funciones definidas en conjuntos difusos)**.** *Decimos que una función F* ***es una función definida en un conjunto difuso*** *si:*

1. $dom\ F \subset \mathbb{R}$
2. $Im\ F \subset \mathcal{F}_{\mathcal{H}}(\mathbb{R}^n)$

2.1.1. Derivada

La derivada de Hukuhara

La derivada de Hukuhara está basada en el concepto de **diferenciabilidad de Hukuhara** para funciones evaluadas en intervalos ([4])

Definición 15 (Diferenciable según Hukuhara)**.** *Sea F una función definida en un conjunto difuso. Supongamos que los límites:*

$$\lim_{h\to 0^+} \frac{F(x_0+h)\ominus_H F(x_0)}{h} \quad \Big\| \quad \lim_{h\to 0^+} \frac{F(x_0)\ominus_H F(x_0-h)}{h}$$

Existen, y son iguales a cierto elemento $F'_H(x_0) \in \mathcal{F}_{\mathcal{H}}(\mathbb{R}^n)$, entonces F es diferenciable según Hukuhara (H-Diferenciable) en x_0 y decimos que $F'_H(x_0)$ es su derivada en x_0

Ejemplo 3 (Función constante)**.** *Sea $F(x) = A$ con $A = (-1; 0; 1)$, calculemos su H-Derivada en un punto arbitrario $x_0 \in \mathbb{R}$:*

$$F(x_0 + h) \ominus_H F(x_0) = A \ominus_H A = 0$$

$$F(x_0) \ominus_H F(x_0 - h) = A \ominus_H A = 0$$

Por tanto, $F'_H(x_0) = 0$

Ejemplo 4 (Función lineal)**.** *Sea $F(x) = Ax$ con $A = (-1; 0; 1)$, supongamos en primer lugar que $x \geq 0$:*

$$F(x + h) \ominus_H F(x) = A(x + h) \ominus_H A = (-x - h; 0; x + h) \ominus_H (-x; 0; x) = (-h; 0; h)$$

Análogamente,

$$F(x) \ominus_H F(x - h) = (-h; 0; h)$$

De donde,

$$\lim_{h \to 0^+} \frac{(-h;0;h)}{h} = A \;\Big\|\; \lim_{h \to 0^+} \frac{(-h;0;h)}{h} = A$$

Por tanto, $F'_H(x) = A$, si $x > 0$.
Por otro lado, si $x < 0$, podemos ver que $F(x + h) \ominus_H F(x)$ no está definido, pues:

- *$(-x - h; 0; x + h)$ no sería un número triangular, pues $-x - h > x + h$*

Esto se puede ver más fácilmente en un resultado que mostramos a continuación.

Proposición 1 (Construcción de funciones H-Diferenciables [5])**.** *Sea G una función definida en conjuntos difusos tal que $G(x) = Bg(x)$ donde $g(x) > 0, g'(x) > 0$ y B es un número difuso entonces, $G(x)$ es H-Diferenciable, más aún,*

$$G'_H(x) = Bg'(x)$$

Una función H-Diferenciable tiene α-corte diferenciables, sin embargo, que todos los α-corte sean diferenciables, no implica que la función sea H-diferenciable.

La derivada de Seikkala

Otro tipo de derivada que veremos, será la derivada de Seikkala, que usaremos después para introducir un concepto de derivada más fuerte.

Definición 16 (Derivada de Seikkala)**.** *Sea $F : [a, b] \to \mathcal{F}_H(\mathbb{R})$ si:*

CAPÍTULO 3

COMPUTACIÓN CIENTÍFICA DE ALTO RENDIMIENTO

En los anteriores capítulos nos hemos centrado en dar definiciones y técnicas para generalizar los conceptos clásicos del cálculos en un contexto difuso, no obstante el objetivo final de este trabajo es encontrar y desarrollar métodos numéricos para resolver problemas en ecuaciones diferenciales difusas.
En este capítulo, **abordaremos las distintas técnicas numéricas** que se van a aplicar en los capítulos posteriores para atacar estos problemas de la forma más eficiente posible.
Este capítulo trata más los problemas desde el punto de vista informático, que necesitaremos para desarrollar correctamente la resolución numérica de los problemas planteados.
Este capítulo está escrito desde la experiencia propia avalada por mi breve, pero intensa experiencia laboral en el mundo de los servicios en la nube, por mi curiosidad, y por mis años como alumno colaborador en la Universidad de Cádiz, y basado en el trabajo realizado en esta diapositiva: [6]

3.1. Conceptos básicos

Dentro de la computación científica, podemos distinguir **una serie de conceptos esenciales**, que tienen que ver en cierta medida con las partes que forman un ordenador. Todo el mundo que tiene un ordenador, seguro que conoce ciertas partes de su ordenador, pues todo el mundo ha hablado alguna vez de la **memoria RAM de su ordenador, el procesador, tarjeta gráfica, tarjeta de red...** pero, ¿Qué impacto tiene cada uno de estos elementos en el desarrollo de la computación científica?

3.1.1. Memoria RAM

Los programas que ejecutamos en nuestros ordenadores se alojan en la memoria RAM, y una vez alojados en la memoria RAM, el procesador se encarga de procesar las instrucciones y ejecuta el código que nosotros le hemos pedido que ejecute. **La memoria RAM es donde se**

almacenan las instrucciones que va a ejecutar el procesador, y todos los datos que generamos en nuestro sistema operativo.

La memoria RAM nos impone un límite de la cantidad de datos que puede estar en ejecución en un momento dado, y si queremos obtener el máximo rendimiento posible, tenemos que **evitar escribir más memoria RAM de la disponible**, sino, nuestro sistema operativo empezará a usar el disco duro para alojar información, esta tecnología se le conoce como ***swap***, y es **mucho más lenta que la memoria RAM.**

Dentro de un ordenador, podemos **dividir la memoria RAM en dos grupos**; **la memoria RAM disponible para el procesador**, y **la memoria RAM disponible para la tarjeta gráfica**. Esto es bastante importante, pues cuando trabajemos con la **tarjeta gráfica, no podremos acceder a punteros alojados en la memoria RAM del procesador**, por tanto, **antes de acceder a esta información deberemos copiarla a la memoria RAM de la tarjeta gráfica.** La operación de copiar datos desde la memoria de la tarjeta gráfica al procesador es bastante lenta, y es conocido que en este punto tenemos **un cuello de botella.**

Tener mucha memoria RAM en nuestra máquina también podría reducir el consumo energético, pues reduciríamos el acceso al disco duro.

3.1.2. Procesador (CPU)

El procesador es la parte de nuestro ordenador que se encarga de interpretar las instrucciones que hemos generado con nuestro programa, y es también **fundamental a la hora de conseguir un rendimiento óptimo de nuestro programa**.

Algunos de los aspectos a tener en cuenta para obtener un **mejor rendimiento sería los hilos de nuestro procesador, y los GHz**. Los **hilos del procesador nos permitirá ejecutar tareas de manera simultanea**, de manera que en muchos casos poder realizar tareas simultaneas (o en paralelo). Por ejemplo, si nuestro procesador tiene 10 hilos, y queremos sumar un vector de 10 elementos, podemos hacer que las 10 sumas que hacen falta para sumar el vector se hagan simultáneamente.

3.1.3. Tarjeta gráfica (GPU)

Habitualmente, nos centramos sólo en el mundo *gaming* cuando se habla de tarjetas gráficas, y pensamos que su única utilidad es para jugar a videojuegos en alta resolución. Sin embargo, son más útiles de lo que puede parecer.

En este caso, las gráficas podemos tratarlas de forma parecida a las CPU, sin embargo, la gran ventaja que ofrecen las GPU es que **están construidas para procesar muchos datos simultáneamente**, debido a que las tarjetas gráficas tienen **muchos más hilos disponibles que la CPU**, pero por otro lado, **cada hilo es más lento.**

3.1.4. Tarjeta de Red (Internet/Intranet/Cluster)

Usar internet o intranet para trabajar con computación de alto rendimiento es una buena práctica también, cuando **conectamos varios ordenadores y los coordinamos para realizar una tarea, se les llaman *cluster***.

Es importante saber que la información a través de la tarjeta de red viaja **más lento que por las otras vías**, y hay otros factores a tener en cuenta, como la distancia física entre los ordenadores, sin embargo si queremos trabajar con **muchísimos hilos es la mejor opción que existe; conectar muchos ordenadores a realizar una misma operación.**

3.2. Técnicas de alto rendimiento

Una vez introducidos los conceptos básicos acerca de los distintos elementos de la computación científica, vamos a hablar de las distintas técnicas que podemos abordar para **obtener un rendimiento lo más óptimo posible.**

3.2.1. Programación de bajo nivel

El **lenguaje de programación que utilicemos va a decantar la balanza en temas de rendimiento**, si queremos exprimir al máximo nuestros ordenadores no nos quedará más remedio que decantarnos por lenguajes de más bajo nivel como C++ o C, al no ser un lenguaje interpretado, como pasa por ejemplo con Python, nuestro código se ejecuta directamente en nuestra máquina.
En las siguientes pruebas, vamos a revisar como se comporta Python y C, con el mismo código, en términos de tiempo de ejecución, uso de RAM: (Código del ejemplo):

- **C es más eficiente a la hora de administrar la memoria RAM**, al probar 10000000000 iteraciones, **Python no puede alocar más memoria RAM, sin embargo, C es capaz de alocar la memoria sin ningún tipo de problema.**
- Por otro lado, los tiempos de ejecución son más sorprendente aún. **Con tan sólo 10 iteraciones, C es 5 veces más rápido que** 1000000 **C es 77 veces más rápido que Python, y finalmente con** 1000000000 **iteraciones, C es 100 veces más rápido que Python.** Aquí se ve la notable diferencia entre uno y otro. Si queremos trabajar en computación científica, trabajar con C debe ser nuestra primera elección. (Ver figura 3.1)

3.2.2. Precisión mixta

Otra técnica menos conocida, pero no por ello menos importante es el uso de precisión mixta. Recordemos en primer lugar, que cuando trabajamos con números en un ordenador debemos de tener en cuenta la precisión a la que estamos trabajando. Existen algunas alternativas para

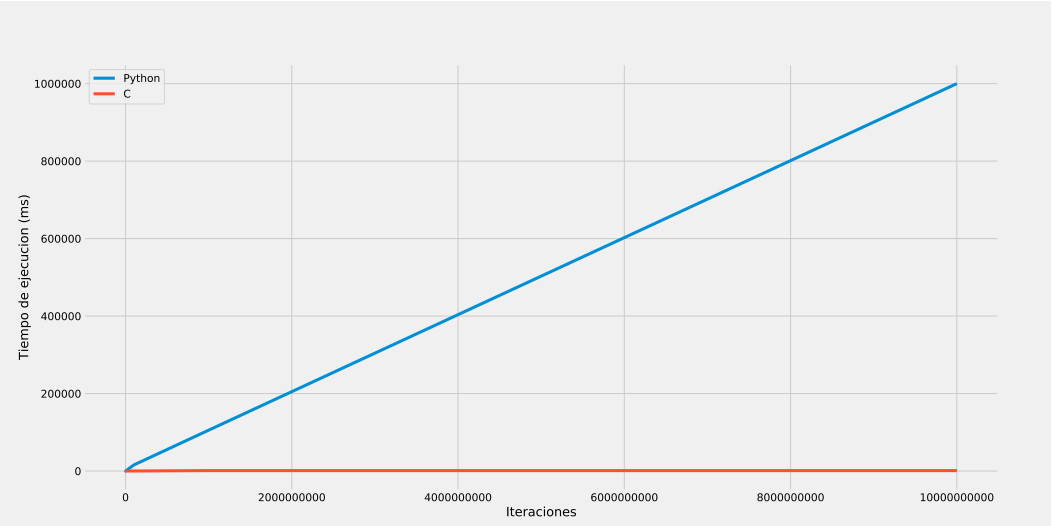


Figura 3.1: Gráfica comparativa C vs Python

trabajar con números en precisión *«infinita»*, sin embargo, no son de nuestro interés pues no rinden tan bien como queremos, así que nos centraremos en dos tipos de precisiones:

- **Precisión simple:** Los números se representan utilizando 4 bytes, por tanto podemos representar 256^4.
- **Precisión doble:** Los números se representan utilizando 8 bytes, por tanto, podemos representar 256^8

Generalmente, los procesadores están optimizados para trabajar mejor que con simple o doble precisión. Las GPU suelen estar mejor optimizadas para trabajar con simple precisión, así que en estos casos, es útil tener en cuenta lo que llamamos precisión mixta si queremos tener un resultado en doble precisión. El procedimiento para trabajar con precisión mixta es el siguiente:

- Planteamos en primer lugar nuestro problema de forma normal, y lo resolvemos en simple precisión.
- Una vez que tenemos el resultado en simple precisión, inicializamos nuestro problema con el valor obtenido, pero esta vez usando doble precisión.

A continuación, mostramos un ejemplo ilustrativo aplicando el método de Newton usando precisión mixta

Ejemplo 5 (Método de Newton precisión mixta, comparativa y desarrollo)**.** *Supongamos que queremos encontrar las raices de la siguiente función:*

$$f(x) = (x-1)^8$$

$$f'(x) = 8(x-1)^7$$

Nuestro objetivo es intentar conseguir una precisión que sea el cero de la máquina. En primer lugar resolveremos el problema en simple precisión, aplicando el método de Newton habitual:

- ***Tiempo de ejecución:*** *0m0,001s*
- ***Iteraciones en simple precisión:*** *99*
- ***Iteraciones en doble precisión:*** *0*
- ***Resultado:*** *0.99999815225601196289062500000*

Lo resolvemos ahora para doble precisión:

- ***Tiempo de ejecución:*** *0m0,001s*
- ***Iteraciones en simple precisión:*** *0*
- ***Iteraciones en doble precisión:*** *265*
- ***Resultado:*** *0.99999999999999955591079014993 7*

Si ahora resolvemos el problema aplicando los principios de la precisión mixta:

- ***Tiempo de ejecución:*** *0m0,001s*
- ***Iteraciones en simple precisión:*** *99*
- ***Iteraciones en doble precisión:*** *166*
- ***Resultado:*** *0.99999999999999955591079014993 7*

Podemos observar, que al resolver nuestro problema con precisión mixta hemos reducido en 100 las operaciones que tenemos que realizar en doble precisión, obteniendo una mejora de rendimiento.

3.2.3. Paralelización de algoritmos

Una de las técnicas más conocidas para acelerar las operaciones que realizamos con un ordenador, es paralelizar los procesos. Decimos que **un algoritmo de n pasos se puede paralelizar** si cada n iteración no depende del resto de iteraciones.
Para conseguir la paralelización, podemos hacerlo mediante diferentes tecnicas:

- **CPU:** Utilizando los hilos disponibles en el procesador de nuestro ordenador. Todos los hilos son de alto rendimiento, pero la cantidad de hilos es bastante pequeña.
- **GPU:** Utilizando los hilos disponibles en la tarjeta gráfica de nuestro ordenador. Los hilos tienen un menor desempeño que los de la CPU, sin embargo, contiene una gran cantidad de hilos.

- **Red:** Distribuimos el trabajo entre varios ordenadores a través de una conexión de red.
- **Mixta:** Cuando se mezclan distintas técnicas de paralelización, se dice que estamos trabajando en paralelización mixta.

A continuación, vamos a mostrar un ejemplo basado en el esquema de diferencias finitas de la ecuación del calor, a modo de entender mejor las mejoras que suponen cada uno:

Ejemplo 6 (Diferencias finitas: Secuencial VS. Paralelo VS. GPU [7])**.** *El esquema en diferencias finitas que vamos a tener en cuenta será:*

$$u_{i,j+1} = u_{i,j} + \mu(u_{i-1,j} - 2u_{i,j} + u_{i+1,j})$$

Podemos observar que los términos que aparecen a la derecha del esquema, dependen exclusivamente de términos de la etapa anterior, por tanto podemos paralelizar cada etapa. Podemos pasar entonces escribir el código

En esta prueba hemos escrito el mismo código en C, CUDA y hemos utilizado OpenMP para generar una versión en paralelo de nuestro código inicial en C. Con esto conseguimos:

- *Tener un código sin paralelizar para poder comparar.*
- *Tener un código exactamente igual al anterior paralelizado en CPU.*
- *Tener un código parecido al anterior pero que funciona en paralelo en la GPU.*

En la siguiente tabla, mostramos el tiempo de ejecución en segundos de cada uno de los programas utilizando las distintas técnicas:

Mallado	C-No paralelo	CPU	GPU
5X5	0,001	0,001	0,441
25X25	0,001	0,001	0,441
50X50	0,001	0,001	0,441
70X70	0,001	0,001	0,441
100X100	0,001	0,001	0,484
1000X1000	0,005	0,005	0,472
10000X10000	1,178	0,624	0,997
20000X20000	8,961	7,754	2,594
22760X22760	22,99	9,227	3,221

A continuación, mostramos el consumo energético en $\mu A/s$:

Mallado	C-No paralelo	CPU	GPU
5X5	0,1	0,23	57,33
25X25	0,1	0,23	57,33
50X50	0,1	0,23	57,33
70X70	0,1	0,23	57,33
100X100	0,1	0,23	62,92
1000X1000	0,5	1,15	61,36
10000X10000	117,8	143,52	129,61
20000X20000	896,1	1783,42	337,22
22760X22760	2299	2122,21	418,73

Cuadro 3.1: Medido con: power_app_stats

Podemos observar que trabajar con GPU nos ofrece un mejor rendimiento, tanto en términos de tiempo como económicos. El consumo energético es mucho menor en GPU que en CPU a lo largo del tiempo.

3.3. Apéndice: Experimentos numéricos

3.3.1. Prueba 1; C Vs Python

Prueba en Python

```
#!/usr/bin/python
import sys

itera  = int(sys.argv[1])

vector = range(0, itera)

for i in range(0, itera):
    vector[i] = i + 1
```

Prueba en C

```
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char ** args) {
    int itera, i, *vector;
```

```
    itera = atoi(args[1]);

    vector = malloc(sizeof(int) * itera);

    for (i = 0; i < itera; i++) {
        vector[i] = i;
    }

    for (i = 0; i < itera; i++) {
        vector[i] = i + 1;
    }
}
```

3.3.2. Prueba 2: Precisión mixta

Precisión simple

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

float f_simple(float x) {
    return pow((x-1), 8);
}

float f_prima_simple(float x) {
    return 8 * pow((x-1), 7);
}

int main(int argc, char ** args) {
    int i;
    float x0, x1;

    x1 = 0;
    i = 0;

    do {
        x0 = x1;
        x1 = x0 - f_simple(x0) / f_prima_simple(x0);

        i++;
    } while (x0 - x1 != 0);

    printf("%d %.30f\n", i, x1);
}
```

Precisión doble

```c
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

double f_double(double x) {
    return pow((x-1), 8);
}

double f_prima_double(double x) {
    return 8 * pow((x-1), 7);
}

int main(int argc, char ** args) {
    int i;
    double x0, x1;

    x1 = 0;
    i = 0;

    do {
        x0 = x1;
        x1 = x0 - f_double(x0) / f_prima_double(x0);

        i++;
    } while (x0 - x1 != 0);

    printf("%d %.30f\n", i, x1);
}
```

Precisión mixta

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

float f_simple(float x) {
    return pow((x-1), 8);
}

float f_prima_simple(float x) {
    return 8 * pow((x-1), 7);
}

double f_double(double x) {
    return pow((x-1), 8);
}

double f_prima_double(double x) {
    return 8 * pow((x-1), 7);
}

int main(int argc, char ** args) {
    int i, j;
    float x0, x1;
    double x0d, x1d;

    x1 = 0;
    i = 0;

    do {
        x0 = x1;
        x1 = x0 - f_simple(x0) / f_prima_simple(x0);

        i++;
    } while (x0 - x1 != 0);

    x1d = x1;
    j   = 0;
    do {
        x0d = x1d;
        x1d = x0d - f_double(x0d) / f_prima_double(x0d);
```

```
        j++;
    } while (x0d - x1d != 0);

    printf("%d %d %d %.30f\n", i, j, i+j, x1d);
}
```

CAPÍTULO 4

MODELOS DE ECUACIONES DIFERENCIALES DIFUSOS

4.1. Aplicaciones a las ciencias naturales

4.1.1. Aplicaciones a la mecánica clásica

4.1.2. Aplicaciones a la mecánica cuántica

4.1.3. Aplicaciones a reacciones químicas

4.1.4. Aplicaciones a la biología

4.2. Aplicaciones a las ciencias sociales

4.2.1. Aplicaciones a la economía

4.2.2. Aplicaciones a crecimientos de población

CAPÍTULO 5

RESOLUCIÓN NUMÉRICA DE ECUACIONES DIFERENCIALES DIFUSAS

5.1. Ecuaciones diferenciales

5.1.1. Método de Euler

5.1.2. Runge-Kutta

5.2. Ecuaciones en derivadas parciales

5.2.1. Elementos finitos

5.2.2. Diferencias finitas

5.3. Experimentos numéricos

5.3.1. Gráficas iteraciones VS Tiempo

5.3.2. Impacto económico

5.3.3. Medioambiente & Impacto energético

BIBLIOGRAFÍA

[1] L. C. d. B. . B. B. Luciana Takata Gomes, "Fuzzy differential equations in various approaches," *Maths*, vol. 1, pp. 11–38, 2015.

[2] J. M. M. . E. M. Reus, "Apuntes de modelización matemática," *Maths*, vol. 3, pp. 1–20, 2018.

[3] D. R. M. Puri, "Fuzzy random variables," *Maths*, vol. 1, pp. 409–422, 1986.

[4] M. Hukuhara, "Intégration des applications measurables dont la valeur est un compact convexe," pp. 205–223, 1967.

[5] B. Bede, "Mathematics of fuzzy sets and fuzzy logic (springer, berlin/heidelberg)," 2013.

[6] J. C. G. Ortega, "Breve introducción a la programación en paralelo," 2018.

[7] J. C. G. Ortega, "Experimentos numéricos," 2018.