UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MÉTODOS ITERATIVOS FRACCIONARIOS PARA LA RESOLUCIÓN DE ECUACIONES Y SISTEMAS NO LINEALES: DISEÑO, ANÁLISIS Y ESTABILIDAD

TESIS DOCTORAL

Presentada por GIRO GUILLERMO CANDELARIO VILLALONA

Dirigida por Dra. ALICIA CORDERO BARBERO

Dr. JUAN RAMÓN TORREGROSA SÁNCHEZ

Dra. MARÍA PENKOVA VASSILEVA

Departamento de Matemática Aplicada

Universitat Politècnica de València

Valencia, mayo de 2023

MÉTODOS ITERATIVOS FRACCIONARIOS PARA LA RESOLUCIÓN DE ECUACIONES Y SISTEMAS NO LINEALES: DISEÑO, ANÁLISIS Y ESTABILIDAD

Tesis Doctoral

Giro Guillermo Candelario Villalona

Dirigida por:

Dra. Alicia Cordero Barbero
Dr. Juan Ramón Torregrosa Sánchez
Dra. María Penkova Vassileva

Departamento de Matemática Aplicada
Universitat Politècnica de València

Valencia, mayo de 2023

Alicia Cordero Barbero, Catedrática de Universidad del Departamento de Matemática Aplicada de la Universitat Politècnica de València, Juan Ramón Torregrosa Sánchez, Catedrático de Universidad del Departamento de Matemática Aplicada de la Universitat Politècnica de València y la Doctora María Penkova Vassileva, Profesora Investigadora en el Instituto Tecnológico de Santo Domingo (República Dominicana),

HACEN CONSTAR:

Que D. Giro Guillermo Candelario Villalona, Ingeniero Civil por el Instituto Tecnológico de Santo Domingo, ha realizado, bajo nuestra dirección, el trabajo que se recoge en esta memoria para optar al título de Doctor en Ciencias Matemáticas por la Universitat Politècnica de València.

Asimismo, autorizamos la presentación de este trabajo ante la Universitat Politècnica de València para que cumpla los trámites correspondientes.

Para que así conste a efectos legales, firmamos este documento en Valencia, a 16 de mayo de 2023.

Fdo. Alicia Cordero Barbero Fdo. María Penkova Vassileva Fdo. Juan R. Torregrosa Sánchez

Resumen

El cálculo fraccionario es una extensión del cálculo clásico, donde el orden de las derivadas o integrales es un número real. Hoy en día, el cálculo fraccionario tiene numerosas aplicaciones en ciencias e ingeniería. La principal razón es el mayor grado de libertad de las herramientas del cálculo fraccionario en comparación con las herramientas del cálculo clásico. Muchos problemas reales se modelan por medio de ecuaciones diferenciales fraccionarias no lineales cuyo sistema de ecuaciones es no lineal, y por tanto, es conveniente que se adapten procedimientos iterativos para resolver problemas no lineales con el uso de derivadas fraccionarias, y observar cuál es la consecuencia en la convergencia de dicho método.

En esta Tesis Doctoral diseñamos nuevos procedimientos iterativos con derivadas fraccionarias (o su aproximación) que al menos igualen a los métodos clásicos en términos de orden de convergencia, mediante la introducción de las derivadas fraccionarias de Riemann-Liouville, de Caputo y conformable (o sus aproximaciones). También, proponemos estudiar la estabilidad de estos esquemas con el uso de planos de convergencia, y planos dinámicos en algunos casos. Finalmente, pretendemos diseñar una técnica que nos permita obtener la versión fraccionaria conformable (o versión con derivada conformable o su aproximación) de cualquier procedimiento iterativo clásico para problemas no lineales.

En el Capítulo 2 se exponen los conceptos previos que serán necesarios para el desarrollo de los siguientes capítulos: Se presentan los conceptos básicos relacionados con métodos de punto fijo, se muestran los esquemas clásicos que trataremos en esta memoria, y finalmente se introducen las herramientas del cálculo fraccionario que serán necesarias para el diseño de procedimientos iterativos fraccionarios.

En el Capítulo 3 se diseñan métodos fraccionarios (o esquemas con derivadas fraccionarias) de tipo Newton-Raphson escalares con las derivadas de Caputo, de Riemann Liouville y la conformable. También diseñamos esquemas fraccionarios de Newton-Raphson escalares de mayor orden. Finalmente, realizamos el análisis de convergencia de dichos procedimientos y estudiamos su estabilidad.

En el Capítulo 4 se diseña la versión vectorial del método de Newton-Raphson conformable visto en el Capítulo 3. Antes, es necesario definir nuevos conceptos y establecer nuevos resultados que serán necesarios para el dersarrollo de este esquema. Finalmente, realizamos el análisis de convergencia y estudiamos su estabilidad.

En el Capítulo 5 se diseñan procedimientos fraccionarios de tipo Traub escalares con derivadas de Caputo y de Riemann-Liouville. También se diseña una técnica general para obtener la versión fraccionaria conformable escalar de cualquier método clásico, y se usa esta técnica para diseñar algunos esquemas conformables multipunto escalares: de tipos Traub, Chun-Kim, Ostrowski y Chun. Por último, se realiza el análisis de convergencia y se estudia la estabilidad de tales procedimientos.

En el Capítulo 6 se diseñan métodos fraccionarios libres de derivadas escalares de tipos Steffensen y Secante (el cual tiene memoria), donde es necesario la aproximación de derivadas conformables. Aquí se usa la técnica general propuesta en el Capítulo 5 para obtener la versión conformable de cada esquema. Finalmente, realizamos el análisis de convergencia y se estudia la estabilidad de dichos procedimientos. En el Capítulo 7 se presentan las conclusiones y líneas futuras de investigación.

Resum

El càlcul fraccionari és una extensió del càlcul clàssic, on l'ordre de les derivades o integrals és un nombre real. Hui dia, el càlcul fraccionari té nombroses aplicacions en ciències i enginyeria. La principal raó és el major grau de llibertat de les eines del càlcul fraccionari en comparació amb les eines del càlcul clàssic. Molts problemes reals es modelen per mitjà d'equacions diferencials fraccionàries no lineals el sistema d'equacions de les quals és no lineal, i per tant, és convenient que s'adapten procediments iteratius per a resoldre problemes no lineals amb l'ús de derivades fraccionàries, i observar quina és la conseqüència en la convergència d'aquest mètode.

En aquesta Tesi Doctoral dissenyem nous procediments iteratius amb derivades fraccionàries (o la seua aproximació) que almenys igualen als mètodes clàssics en termes d'ordre de convergència, mitjançant la introducció de les derivades fraccionàries de Riemann-Liouville, de Caputo i conformable (o les seues aproximacions). També, proposem estudiar l'estabilitat d'aquests esquemes amb l'ús de plans de convergència, i plans dinàmics en alguns casos. Finalment, pretenem dissenyar una tècnica que ens permeta obtindre la versió fraccionària conformable (o versió amb derivada conformable o la seua aproximació) de qualsevol procediment iteratiu clàssic per a problemes no lineals.

En el Capítol 2 s'exposen els conceptes previs que seran necessaris per al desenvolupament dels següents capítols: Es presenten els conceptes bàsics relacionats amb mètodes de punt fix, es mostren els esquemes clàssics que tractarem en aquesta memòria, i finalment s'introdueixen les eines del càlcul fraccionari que seran necessàries per al disseny de procediments iteratius fraccionaris.

En el Capítol 3 es dissenyen mètodes fraccionaris (o esquemes amb derivades fraccionàries) de tipus Newton-Raphson escalars amb les derivades de Caputo, de Riemann Liouville i la conformable. També dissenyem esquemes fraccionaris de Newton-Raphson escalars de major ordre. Finalment, realitzem l'anàlisi de convergència d'aquests procediments i estudiem la seua estabilitat.

En el Capítol 4 es dissenya la versió vectorial del mètode de Newton-Raphson conformable vist en el Capítol 3. Abans, és necessari definir nous conceptes i establir nous resultats que seran necessaris per al dersarrollo d'aquest esquema. Finalment, realitzem l'anàlisi de convergència i estudiem la seua estabilitat.

En el Capítol 5 es dissenyen procediments fraccionaris de tipus Traub escalars amb derivades de Caputo i de Riemann-Liouville. També es dissenya una tècnica general per a obtindre la versió fraccionària conformable escalar de qualsevol mètode clàssic, i s'usa aquesta tècnica per a dissenyar alguns esquemes conformables multipunt escalars: de tipus Traub, Chun-Kim, Ostrowski i Chun. Finalment, es realitza l'anàlisi de convergència i s'estudia l'estabilitat de tals procediments.

En el Capítol 6 es dissenyen mètodes fraccionaris lliures de derivades escalars de tipus Steffensen i Assecant (el qual té memòria), on és necessari l'aproximació de derivades conformables. Ací s'usa la tècnica general proposta en el Capítol 5 per a obtindre la versió conformable de cada esquema. Finalment, realitzem l'anàlisi de convergència i s'estudia l'estabilitat d'aquests procediments. En el Capítol 7 es presenten les conclusions i línies futures d'investigació.

Abstract

Fractional calculus is an extension of classical calculus, where the order of the derivatives or integrals is a real number. Today, fractional calculus has numerous applications in science and engineering. The main reason is the higher degree of freedom of the fractional calculus tools compared to the classical calculus tools. Many real problems are modeled by means of nonlinear fractional differential equations whose system of equations is nonlinear, and therefore it is convenient that iterative procedures are adapted to solve nonlinear problems with the use of fractional derivatives, and observe what the consequence is in the convergence of said method.

In this Doctoral Thesis we design new iterative procedures with fractional derivatives (or their approximation) that are at least equal to the classical methods in terms of convergence order, by introducing the Riemann-Liouville, Caputo and conformable fractional derivatives (or their approximations). Also, we propose to study the stability of these schemes with the use of convergence planes, and dynamic planes in some cases. Finally, we intend to design a technique that allows us to obtain the conformable fractional version (or version with conformable derivative or its approximation) of any classical iterative procedure for nonlinear problems.

In Chapter 2 the previous concepts that will be necessary for the development of the following chapters are exposed: The basic concepts related to fixed point methods are presented, the classic schemes that we will deal with in this memory are shown, and finally the tools of the fractional calculus that will be necessary for the design of fractional iterative procedures.

In Chapter 3, scalar Newton-Raphson type fractional methods (or schemes with fractional derivatives) are designed with the Caputo, Riemann Liouville and conformable derivatives. We also design higher order scalar Newton-Raphson fractional schemes. Finally, we perform the convergence analysis of these procedures and study their stability.

In Chapter 4, the vector version of the conformable Newton-Raphson method seen in Chapter 3 is designed. Before, it is necessary to define new concepts and establish new results that will be necessary for the development of this scheme. Finally, we perform the convergence analysis and study its stability.

In Chapter 5, fractional procedures of the scalar Traub type with derivatives of Caputo and Riemann-Liouville are designed. A general technique is also designed to obtain the scalar conformable fractional version of any classical method, and this technique is used to design some scalar multipoint conformable schemes: of Traub, Chun-Kim, Ostrowski and Chun types. Finally, the convergence analysis is carried out and the stability of such procedures is studied.

In Chapter 6 free fractional methods of scalar derivatives of Steffensen and Secant types (which has memory) are designed, where the conformable derivatives approximation is necessary. Here we use the general technique proposed in Chapter 5 to obtain the conformable version of each scheme. Finally, we carry out the convergence analysis and the stability of these procedures is studied. In Chapter 7 the conclusions and future lines of research are presented.

A la memoria de Ángela Carrasco

Agradecimientos

Quiero ofrecer mi más sincero agradecimiento a las personas que de alguna manera han hecho más fácil la culminación de esta memoria:

Primeramente a DIOS, por abrirme las puertas, por poner en mi camino a la gente necesaria en el tiempo oportuno.

A mis directores, Dra. Alicia Cordero, Dr. Juan Ramón Torregrosa y Dra. María Penkova, por su disposición y humildad, por su paciencia, aprecio muchísimo cada consejo, quedarán ahí para mis experiencias futuras.

A la Dra. María Penkova, por ofrecerme la oportunidad de ser parte del equipo, por interceder por mí, por procurar mi bienestar; nada de esto hubiera sido posible sin usted. A los profesores Dr. Santiago Artidiello y Dr. Javier García, por sus oportunas recomendaciones, por ayudarme a crecer.

A mi madre Elisabet Villalona y mi hermana Mirtha Mir, por siempre apoyarme en todo, por siempre estar presente.

A mi esposa Jeniffer Núñez, por su paciencia, por su apoyo, por sus inesperados consejos, y por soportarme en todo este proceso.

A todos, muchísimas gracias.

Santo Domingo, 2023.

Índice general

Glosario de símbolos

$\odot$	producto de Hadamard
$\frac{\partial_a^\alpha}{\partial x_i^\alpha} f(x)$	derivada parcial conformable de f, de orden α, en x_i, centrada en a, donde $x = (x_1, \ldots, x_i, \ldots, x_n)$
$\hat{0}$	matriz nula de orden n
a	punto donde están centradas las derivadas fraccionarias
$\hat{a}$	punto donde están centradas las Jacobianas conformables
A_1, A_2, A_3, A_4	variables auxiliares para el análisis de convergencia de los métodos de tipo Traub con derivadas de Caputo y Riemann-Liouville
α	orden de las derivadas fraccionarias, donde $\alpha \in (0, 1]$
b	punto donde se evalúan las derivadas conformables en serie de Taylor conformable escalar
$\hat{b}$	punto donde se evalúan las Jacobianas conformables en serie de Taylor conformable vectorial
β	solución de un método con memoria
C	constante asintótica del error en método iterativo sin memoria
CeCO	método de tipo Chun escalar con derivada conformable, de orden 4
CKeCO	método de tipo Chun-Kim escalar con derivada conformable, de orden 3
C^n	clase n de una función f
C_j	$= \dfrac{f^{(j)}(\bar{x})}{j! f'(\bar{x})}$
$\hat{C}_q$	$= \dfrac{1}{q!} \left[F'(\tilde{x})\right]^{-1} F^{(q)}(\tilde{x})$
C_j^{CA}	$= \dfrac{\Gamma(\alpha+1)}{\Gamma(j\alpha+1)} \dfrac{{}_cD_{0^+}^{j\alpha} f(\bar{x})}{{}_cD_{0^+}^{\alpha} f(\bar{x})}$
C_j^{RL}	$= \dfrac{\Gamma(\alpha+1)}{\Gamma(j\alpha+1)} \dfrac{D_{0^+}^{j\alpha} f(\bar{x})}{D_{0^+}^{\alpha} f(\bar{x})}$
C_j^{CO}	$= \dfrac{(T_\alpha^a f)^{(j)}(\bar{x})}{j! \alpha^{j-1} (T_\alpha^a f)(\bar{x})}$
$\hat{C}_q^{CO}$	$= \dfrac{1}{q! \alpha^{q-1}} \left[F_a^{\alpha(1)}(\tilde{x})\right]^{-1} F_a^{\alpha(q)}(\tilde{x})$
θ	parámetro amortiguador en método de Traub clásico
d	cantidad de evaluaciones funcionales de un método iterativo
D	conjunto convexo abierto en $\mathbb{R}^n$
$D^\alpha f(t)$	operador fraccionario de la función f, de orden α
${}_cD_{a^+}^\alpha f(t)$	operador fraccionario de Caputo de la función f, de orden α
$D_{a^+}^\alpha f(t)$	operador fraccionario de Riemann-Liouville de la función f, de orden α
δ_1	$= H^\alpha - L^\alpha$; $H = x - a$, $L = b - a$, donde $x \in \mathbb{R}$
Δ	$= U^{\odot\alpha} - V^{\odot\alpha}$; $U = x - \hat{a}$, $V = \hat{b} - \hat{a}$, donde $x \in \mathbb{R}^n$
e_k	error cometido en la k-ésima iteración, en $\mathbb{R}$

$e^{(k)}$	error cometido en la k-ésima iteración, en $\mathbb{R}^n$
EeCO	método de tipo Secante escalar con aproximación de la derivada conformable, de orden aproximado 1.618
$E_{\mu,\sigma}(z)$	función de *Mittag-Leffler* en dos parámetros (μ y σ), evaluada en z
$f(x)$	función escalar, donde $f : \mathbb{R} \to \mathbb{R},\ x \in \mathbb{R}$
$F(x)$	función vectorial, donde $F : \mathbb{R}^n \to \mathbb{R}^n,\ x \in \mathbb{R}^n$
$f'(x)$	derivada clásica de la función f
$F'(x)$	Jacobiana clásica de la función F
$F_{\hat{a}}^{\alpha(1)}(x)$	Jacobiana conformable de la función F
$\phi(x)$	función iterada, evaluada en x
γ	punto en torno al cual se desarrolla la serie de Taylor clásica de la función f
$\Gamma(z)$	función *Gamma*, evaluada en z
η	constante asintótica del error en método iterativo con memoria
i	unidad imaginaria (cuando no se especifique)
I	intervalo abierto en $\mathbb{R}$
I_n	matriz identidad de orden n
$\chi_{i,j}$	variable auxiliar para elementos de Jacobianas conformables grandes
K_q	coeficientes en serie de Taylor conformable centrada en el mismo punto que se evalúa ($t_0 \in \mathbb{R}^n$)
κ	parámetro de la familia de King (para Ostrowski y Chun)
$\mathcal{L}(\cdot)$	aplicación lineal de $\cdot \to \cdot$
m	multiplicidad de $\bar{x}$
NeCA	método de tipo Newton-Raphson escalar con derivada de Caputo, de orden $\alpha + 1$
NeRL	método de tipo Newton-Raphson escalar con derivada de Riemann-Liouville, de orden $\alpha + 1$
NeCO	método de tipo Newton-Raphson escalar con derivada conformable, de orden 2
NeL3	método de tipo Newton-Raphson escalar con derivada conformable, con α variable, de orden 3
NeA3	método de tipo Newton-Raphson escalar con derivada conformable, con a variable, de orden 3
NeLA4	método de tipo Newton-Raphson escalar con derivada conformable, con α y a variables, de orden 4
NvCO	método de tipo Newton-Raphson vectorial con Jacobiana conformable, de orden 2
O	orden del error de un método iterativo
OeCO	método de tipo Ostrowski escalar con derivada conformable, de orden 4
$O_R((MI), \bar{x})$	R-orden de convergencia de un método iterativo (MI) en el punto $\bar{x}$
p	orden de convergencia de un método iterativo sin memoria
$\psi(x)$	función iterada compuesta, evaluada en x
$R_m(x)$	R-factor de la sucesión de iterados $\{x_k\}$
ρ	orden de convergencia computacional aproximado (ACOC) escalar
$\hat{\rho}$	orden de convergencia computacional aproximado (ACOC) vectorial
s^*	única solución positiva del polinomio carcterístico, tal que $O_R((MI), \bar{x}) \geq s^*$
SeCO	método de tipo Steffensen escalar con aproximación de la derivada conformable, de orden 2
TeCA	método de tipo Traub escalar con derivada de Caputo, de orden $2\alpha + 1$
TeRL	método de tipo Traub escalar con derivada de Riemann-Liouville, de orden $2\alpha + 1$
TeCO	método de tipo Traub escalar con derivada conformable, de orden 3
$(T_\alpha^a f)(x)$	derivada conformable de f por la izquierda, de orden α, centrada en a, evaluada en x
x_k	k-ésimo iterado en $\mathbb{R}$
$x^{(k)}$	k-ésimo iterado en $\mathbb{R}^n$
$\bar{x}$	un cero de f
$\tilde{x}$	un cero de F
$\{x_k\}$	sucesión de iterados en $\mathbb{R}$

$\{x^{(k)}\}$	sucesión de iterados en $\mathbb{R}^n$
x_0	estimación inicial de $\bar{x}$ en $\mathbb{R}$
$x^{(0)}$	estimación inicial de $\tilde{x}$ en $\mathbb{R}^n$
$\tilde{X}_q$	variable auxiliar para calcular $\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1}$
$\bar{X}_q$	variable auxiliar para calcular $\left[F'\left(x^{(k)}\right)\right]^{-1}$
y_k	predictor de función iterada

Capítulo 1

Introducción

"Porque todo aquel que pide, recibe; y el que busca, halla; y al que llama, se le abrirá"

Jesús de Nazaret (hacia 7-3 a. C. - hacia 27-34)

El cálculo fraccionario es una generalización del cálculo clásico, donde las derivadas (o integrales) son de orden fraccionario, es decir, el orden de las derivadas (o integrales) es un número real. Dado que el cálculo clásico constituye un caso particular del fraccionario, éste último conserva muchas de las propiedades del cálculo clásico, aunque no siempre es así (véase [31, 33, 35, 37]).

El concepto de cálculo fraccionario se introdujo casi simultáneamente (pocos años después) del desarrollo del cálculo clásico. Las primeras referencias se remontan a 1695, cuando Leibniz y L'Hôpital crearon el concepto de semi-derivada, o derivada de orden 1/2 [33, 39]. Otros investigadores de la época (como Riemann, Liouville o Euler), y hasta nuestros días, han estado interesados en esta idea, proponiendo cada vez derivadas e integrales fraccionarias que preservan en mayor medida las propiedades del cálculo clásico. Desde su desarrollo inicial en el siglo XIX hasta hoy en día, el cálculo fraccionario ha evolucionado desde aspectos teóricos hasta la aparición en muchas aplicaciones del mundo real: física, biología, ingenierías, medicina, economía, optimización, control, sistemas dinámicos, procesamiento de señales, entre otras. Con frecuencia, muchos de estos problemas reales se modelizan mediante ecuaciones diferenciales con derivadas de orden fraccionario [5, 24, 30].

En la actualidad, el cálculo fraccionario tiene numerosas aplicaciones en ciencia e ingeniería. La razón fundamental es el mayor grado de libertad de las herramientas del cálculo fraccionario en comparación con las herramientas del cálculo clásico. Esto lo convierte en el procedimiento más adecuado para problemas de modelado cuyas propiedades hereditarias deben conservarse. En este sentido, una de las herramientas más significativas del cálculo fraccionario es la derivada (integral) fraccionaria. En muchas ocasiones, estos problemas de modelado están relacionados con los sistemas de ecuaciones, que pueden ser no lineales, si también lo es la ecuación diferencial fraccionaria [28, 38]. Por tanto, no es extraño que se adapten procesos iterativos para resolver ecuaciones no lineales por medio de derivadas fraccionarias de diferentes órdenes, y ver cuál es el efecto resultante en la convergencia de dicho método.

Supongamos que pretendemos calcular un cero de una función no lineal de la forma $f(x) = 0$. Podemos resolver este problema por medios analíticos, en caso de ser posible, o podemos aproximar la solución por medio de métodos numéricos. En la mayoría de los casos, éste último será más adecuado, debido al bajo coste computacional que suponen. Los métodos numéricos más usados son los llamados de *punto fijo*; escalares o vectoriales, punto a punto o multipunto, con o sin derivadas, con o sin memoria. Entre los esquemas clásicos conocidos podemos mencionar: Newton-Raphson, Secante, Steffensen, Traub, Ostrowski, Chun, etc. [34, 36, 42]

El objetivo de esta Tesis Doctoral es el diseño de nuevos procedimientos iterativos con derivadas fraccionales

rias (o su aproximación) que al menos igualen a los métodos clásicos en términos de orden de convergencia, mediante la introducción de las derivadas fraccionarias de Riemann-Liouville, de Caputo y conformable (o sus aproximaciones). También, proponemos estudiar la estabilidad de estos esquemas con el uso de planos de convergencia, y planos dinámicos en algunos casos. Finalmente, pretendemos diseñar una técnica general que nos permita obtener la versión fraccionaria conformable (o versión con derivada conformable o su aproximación) de cualquier procedimiento iterativo clásico para problemas no lineales.

En el Capítulo 2, se exponen los conceptos previos que serán necesarios para el desarrollo de los siguientes capítulos: Se presentan los conceptos básicos relacionados con métodos de punto fijo, se muestran los esquemas clásicos que trataremos en esta memoria, y finalmente se introducen las herramientas del cálculo fraccionario que serán necesarias para el diseño de procedimientos iterativos fraccionarios.

En el Capítulo 3, se diseñan métodos fraccionarios (o esquemas con derivadas fraccionarias) de tipo Newton-Raphson escalares con las derivadas de Caputo, de Riemann Liouville y la conformable. También diseñamos esquemas fraccionarios de Newton-Raphson escalares de mayor orden. Finalmente, realizamos el análisis de convergencia de dichos procedimientos y estudiamos su estabilidad.

En el Capítulo 4, se diseña la versión vectorial del método de Newton-Raphson conformable visto en el Capítulo 3. Antes, es necesario definir nuevos conceptos y establecer nuevos resultados que serán necesarios para el dersarrollo de este esquema. Finalmente, realizamos el análisis de convergencia y estudiamos su estabilidad.

En el Capítulo 5, se diseñan procedimientos fraccionarios de tipo Traub escalares con derivadas de Caputo y de Riemann-Liouville. También se diseña una técnica general para obtener la versión fraccionaria conformable escalar de cualquier método clásico, y se usa esta técnica para diseñar algunos esquemas conformables multipunto escalares: de tipos Traub, Chun-Kim, Ostrowski y Chun. Por último, se realiza el análisis de convergencia y se estudia la estabilidad de tales procedimientos.

En el Capítulo 6, se diseñan métodos fraccionarios libres de derivadas escalares de tipos Steffensen y Secante (el cual tiene memoria), donde es necesario la aproximación de derivadas conformables. Aquí se usa la técnica general propuesta en el Capítulo 5 para obtener la versión conformable de cada esquema. Finalmente, realizamos el análisis de convergencia y se estudia la estabilidad de dichos procedimientos.

En el Capítulo 7, se presentan las conclusiones y líneas futuras de investigación. Finalmente, se incluyen los anexos y la bibliografía utilizada.

Capítulo 2

Conceptos previos

"Las matemáticas poseen no sólo la verdad, sino cierta belleza suprema. Una belleza fría y austera, como la de una escultura"

Bertrand Russell (1872 - 1970)

Los métodos iterativos para la resolución de problemas no lineales que vamos a tratar son los llamados *"de punto fijo"* [36, 42]. En este capítulo presentamos los conceptos previos necesarios para trabajar con este tipo de métodos numéricos. También presentamos algunos métodos numéricos clásicos, los cuales utilizaremos para la deducción de sus respectivas variantes fraccionarias en capítulos posteriores. Luego se introducen las herramientas necesarias del cálculo fraccionario [2, 18, 19, 21, 27, 31, 33, 37, 43], y finalmente una herramienta de la dinámica real desarrollada en [29], con la cual realizaremos el análisis de estabilidad de los métodos propuestos.

2.1. Conceptos básicos

En esta sección, introducimos algunos conceptos necesarios para el análisis de convergencia de métodos iterativos clásicos (escalares y vectoriales) para resolver ecuaciones y sistemas no lineales.

2.1.1. Métodos iterativos de punto fijo

Supongamos que $f : I \subseteq \mathbb{R} \to \mathbb{R}$ es una función continua, y que es suficientemente derivable en el entorno de una raíz $\bar{x} \in \mathbb{R}$. Si $\bar{x}$ es una raíz de f, entonces se cumple que $f(\bar{x}) = 0$, es decir, $\bar{x}$ es una solución de la ecuación

$$f(x) = 0. \tag{2.1}$$

Podemos aproximar la raíz $\bar{x}$ por medio de algún método iterativo de la forma

$$x_{k+1} = \phi(x_k), \quad k = 0, 1, 2, \ldots, \tag{2.2}$$

donde la información obtenida en x_{k+1} depende de la determinada en x_k, partiendo de una estimación inicial x_0. A la función ϕ se la conoce como función iterada, o función de punto fijo, y puede ser clasificada como función punto a punto o multipunto, con o sin memoria. En una función de punto fijo con memoria, la información que se obtiene en x_{k+1} depende de la determinada en las estimaciones x_k, x_{k-1}, x_{k-2}, etc.

La ecuación (2.2) es una función iterada punto a punto sin memoria, como la mayoría de los métodos que veremos en esta memoria. Al menos una solución $\bar{x}$ de $f : [a, b] \to [a, b]$ será punto fijo de ϕ, y se cumple que $\phi(\bar{x}) = \bar{x}$ [36, 42].

Traub demuestra que para un problema de punto fijo $\phi(x) = x$, ϕ continua en un intervalo $[a, b]$, existe una solución $\bar{x}$, $a \leq \bar{x} \leq b$, tal que $\phi(\bar{x}) = \bar{x}$ (Lema 2-1, [42]), e imponiendo la condición de Lipschitz demuestra que ésta es única (Lema 2-2, [42]). También demuestra que la sucesión generada por la ecuación (2.2) converge a esta solución (Teorema 2-1, [42]).

Todos los métodos que veremos en este y los siguientes capítulos son funciones iteradas de punto fijo, o métodos iterativos de punto fijo, es decir, en éstos se cumple que $\phi(\bar{x}) = \bar{x}$.

2.1.2. Orden de convergencia

Supongamos que la sucesión de iterados $x_0, x_1, \ldots, x_n, \ldots$ converge a la raíz $\bar{x}$, es decir, $\lim_{k \to \infty} x_k = \bar{x}$. Denotemos por $e_{k+1} = x_{k+1} - \bar{x}$ y $e_k = x_k - \bar{x}$ a los errores cometidos en la *k+1-ésima* y *k-ésima* iteración, respectivamente. Si existen un $p \in \mathbb{R}$, $p > 1$, y una constante $C > 0$ tal que

$$\lim_{k \to \infty} \frac{|e_{k+1}|}{|e_k|^p} = \lim_{k \to \infty} \frac{|x_{k+1} - \bar{x}|}{|x_k - \bar{x}|^p} = C, \tag{2.3}$$

entonces p es llamado el orden de convergencia, y C es la constante de error asintótica de dicha sucesión [36, 42]. Si $p = 1$, se dice que la sucesión $\{x_k\}$ converge linealmente si $0 < C < 1$, o que tiene orden de convergencia lineal, si $p = 2$, se dice que tiene orden de convergencia cuadrático, y así sucesivamente.

Para denotar el orden de convergencia, usaremos la notación asintótica O; de esta forma, una expresión equivalente a la ecuación (2.3) es

$$x_{k+1} - \bar{x} = O((x_k - \bar{x})^p), \tag{2.4}$$

donde se dice que $x_{k+1} - \bar{x}$ es del orden de $(x_k - \bar{x})^p$. De forma equivalente, podemos reescribir la ecuación (2.4) en términos del error:

$$e_{k+1} = O(e_k^p). \tag{2.5}$$

La ecuación (2.5) es la llamada ecuación del error.

Traub también establece que si una función de punto fijo ϕ es continua en el entorno de la solución $\bar{x}$, y $e_k = x_k - \bar{x}$, entonces ϕ es de orden p si y sólo si

$$\phi(\bar{x}) = \bar{x}; \quad \phi^{(j)}(\bar{x}) = 0, \quad j = 1, 2, \ldots, p-1; \quad \phi^{(p)}(\bar{x}) \neq 0$$

(Teorema 2-2, [42]). Además,

$$\lim_{k \to \infty} \frac{e_{k+1}}{e_k^p} = \frac{\phi^{(p)}(\bar{x})}{p!}. \tag{2.6}$$

La conjetura de Kung y Traub (ver [26]) establece una cota superior para el orden de convergencia de un método de punto fijo ϕ, sin memoria, en función de su cantidad de evaluaciones funcionales d por iteración. Según esta conjetura, el orden de convergencia de ϕ no puede ser superior a 2^{d-1}. Si un método de punto fijo tiene orden de convergencia 2^{d-1}, se dice que este método es óptimo.

En el caso de los métodos iterativos con memoria, la ecuación (2.3) no puede ser usada para determinar el orden de convergencia. Ortega y Rheinboldt introdujeron una generalización de orden de convergencia en $\mathbb{R}^n$ (ver [34, 36]), llamado R-orden, con la cual podemos determinar el orden de convergencia de métodos con memoria. Veamos primero el concepto de R-factor:

Definición 2.1.1 *Sea ϕ un método iterativo convergente a algún límite β, y sea $\{x_k\}$ una sucesión arbitraria en $\mathbb{R}^n$ que converge a β. Entonces, el R-factor de la sucesión $\{x_k\}$ es*

$$R_m(x) = \begin{cases} \lim\limits_{k\to\infty} \sup ||x_k - \beta||^{1/k}, & \text{para } m = 1, \\ \lim\limits_{k\to\infty} \sup ||x_k - \beta||^{1/m^k}, & \text{para } m > 1. \end{cases} \tag{2.7}$$

Podemos ahora definir R-orden de la siguiente manera:

Definición 2.1.2 *El R-orden de convergencia de un método iterativo ϕ en el punto β es*

$$O_R(\phi, \beta) = \begin{cases} +\infty, & \text{si } R_m(\phi, \beta) = 0 \ \ \forall m \in [1, +\infty), \\ \text{ínf}\{m \in [1, +\infty) : R_m(\phi, \beta) = 1\}, & \text{en otro caso.} \end{cases} \tag{2.8}$$

El siguiente resultado nos proporciona una relación entre las raíces del polinomio característico y el R-orden de convergencia de un método iterativo con memoria:

Teorema 2.1.1 *Sea ϕ un método iterativo con memoria que genera la sucesión $\{x_k\}$ de aproximaciones a la raíz $\bar{x}$, y supongamos que la sucesión $\{x_k\}$ converge a $\bar{x}$. Si existe una constante no nula η, y números no negativos t_i, $i \in [0, m]$, tal que se cumple la desigualdad*

$$|e_{k+1}| \leq \eta \prod_{i=0}^{m} |e_{k-i}|^{t_i},$$

entonces el R-orden de convergencia del método iterativo ϕ satisface la siguiente desigualdad

$$O_R(\phi, \bar{x}) \geq s^*,$$

donde s^* es la única raíz positiva del polinomio

$$s^{m+1} - \sum_{i=0}^{m} t_i s^{m-i} = 0. \tag{2.9}$$

Una herramienta necesaria para el análisis de convergencia de métodos iterativos escalares es la serie de Taylor, la cual aproxima una función como un polinomio de grado n mediante serie de potencias; mientras mayor sea el grado del polinomio, mejor será la aproximación. El siguiente resultado nos proporciona la serie de Taylor clásica en una variable.

Teorema 2.1.2 *(Teorema de Taylor, [9]) Si una función $y = f(x)$ es continuamente diferenciable $n-1$ veces en el intervalo $[\gamma, \gamma + h]$, y si también existe la n-ésima derivada en el interior del intervalo, entonces el desarrollo de Taylor de f en torno a γ es*

$$f(\gamma + h) = f(\gamma) + \frac{h}{1!} f'(\gamma) + \frac{h^2}{2!} f''(\gamma) + \cdots + \frac{h^{n-1}}{(n-1)!} f^{(n-1)}(\gamma) + \frac{h^n}{n!} f^{(n)}(\gamma + \Theta h), \tag{2.10}$$

con $0 < \Theta < 1$.

Podemos escribir el desarrollo de Taylor de una función escalar f alrededor de la solución $\bar{x} \in \mathbb{R}$ de $f(x) = 0$ de la siguiente manera:

$$f(x_k) = f'(\bar{x}) \left[e_k + C_2 e_k^2 + C_3 e_k^3 + \cdots + C_p e_k^p \right] + O\left({e_k}^{p+1} \right), \tag{2.11}$$

donde $e_k = x_k - \bar{x}$, y $C_j = \dfrac{f^{(j)}(\bar{x})}{j! f'(\bar{x})}$, $j \geq 2$, siendo $f'(\bar{x})$ no nula.

Para el análisis de convergencia de métodos iterativos vectoriales podemos encontrar en [4, 16] la siguiente notación:

Definición 2.1.3 *Sea $F : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ suficientemente Frèchet-diferenciable en D. La q-ésima derivada de F en $u \in \mathbb{R}^n$, $q \in \mathbb{N}$, $q \geq 1$, es la función q-lineal $F^{(q)}(u) : \mathbb{R}^n \times \cdots \times \mathbb{R}^n \to \mathbb{R}^n$ tal que $F^{(q)}(u)(v_1, \ldots, v_q) \in \mathbb{R}^n$. Se puede observar que:*

1. *$F^{(q)}(u)(v_1, \ldots, v_{q-1}, \cdot) \in \mathcal{L}(\mathbb{R}^n)$, siendo $\mathcal{L}(\mathbb{R}^n)$ el espacio de la aplicación lineal de $\mathbb{R}^n \to \mathbb{R}^n$.*
2. *$F^{(q)}(u)(v_{\sigma_1}, \ldots, v_{\sigma_q}) = F^{(q)}(u)(v_1, \ldots, v_q)$, para cualquier permutación $\sigma \in \{1, \ldots, q\}$.*

Con estas propiedades podemos usar la siguiente notación:

1. *$F^{(q)}(u)(v_1, \ldots, v_q) = F^{(q)}(u) v_1 \cdots v_q$.*
2. *$F^{(q)}(u) v^{q-1} F^{(p)}(u) v^p = F^{(q)}(u) F^{(p)}(u) v^{q+p-1}$.*

Con estas expresiones podemos escribir el desarrollo de Taylor de una función vectorial F alrededor de la solución $\tilde{x} \in \mathbb{R}^n$ de $F(x) = \hat{0}$ de la siguiente manera:

$$F\left(x^{(k)}\right) = F'(\tilde{x}) \left[e^{(k)} + \hat{C}_2 {e^{(k)}}^2 + \hat{C}_3 {e^{(k)}}^3 + \cdots + \hat{C}_p {e^{(k)}}^p \right] + O\left({e^{(k)}}^{p+1} \right), \tag{2.12}$$

donde $e^{(k)} = x^{(k)} - \tilde{x}$, y $\hat{C}_q = \dfrac{1}{q!} \left[F'(\tilde{x}) \right]^{-1} F^{(q)}(\tilde{x})$, $q \geq 2$, siendo la matriz Jacobiana $F'(\tilde{x})$ no singular. Notemos que ${e^{(k)}}^p$ es $\underbrace{\left(e^{(k)}, e^{(k)}, \cdots, e^{(k)} \right)}_{\text{p componentes}}$.

Sabiendo que la constante de error asintótica C de un método iterativo $\phi(x)$ de orden p se define como [42]

$$C = \lim_{x \to \bar{x}} \frac{\phi(x) - \bar{x}}{(x - \bar{x})^p},$$

el siguiente resultado nos permite calcular la constante de error asintótica de un esquema iterativo de orden p si conocemos la constante de error asintótica de otro procedimiento iterativo de orden p.

Teorema 2.1.3 *(Teorema 2-8, [42]) Sean $\phi_1(x)$, $\phi_2(x)$ de orden p cuya solución $\bar{x}$ es de multiplicidad m. Sea*

$$G(x) = \frac{\phi_2(x) - \phi_1(x)}{(x - \bar{x})^p}, \quad x \neq \bar{x}.$$

Sean C_1, C_2 las constantes de error asintóticas de ϕ_1 y ϕ_2, respectivamente. Entonces,

$$C_2 = C_1 + \lim_{x \to \bar{x}} G(x).$$

La tesis del Teorema 2.1.3 la confirmamos en la mayoría de los métodos que se proponen en los siguientes capítulos de esta memoria. En el caso vectorial (como se verá en el Capítulo 4) introducimos normas, es decir, consideramos $||\phi(x) - \tilde{x}||$, $||\phi_2(x) - \phi_1(x)||$ y $||x - \tilde{x}||^p$.

2.2. Algunos métodos iterativos clásicos

En esta sección, introducimos los esquemas iterativos clásicos que se usan en esta memoria, y en los próximos capítulos diseñamos sus variantes fraccionarias. En adelante, cuando hagamos referencia a procedimientos iterativos clásicos nos referimos a métodos con derivadas (o su aproximación) de orden entero, es decir, de orden 1,2, etc., mientras que cuando hagamos referencia a esquemas iterativos fraccionarios nos referimos a procedimientos con derivadas (o su aproximación) de orden fraccionario, es decir, de orden $\alpha \in \mathbb{R}$, $\alpha \in (0,1]$. Abordaremos primero los métodos de tipo Newton-Raphson (la versión escalar, y después la versión vectorial), luego veremos los esquemas multipunto (de Traub, Chun-Kim, Ostrowski y Chun), y finalmente los procedimientos libres de derivadas (el de Steffensen y el de la Secante); teniendo en cuenta que el método de la Secante es el único esquema con memoria que trataremos.

2.2.1. Método de Newton-Raphson escalar

El esquema de Newton-Raphson es el procedimiento de punto fijo más conocido. En este método se evalúa tanto la función como su primera derivada en cada iterado, y su orden de convergencia es cuadrático (es un método óptimo). Este es un esquema punto a punto y sin memoria. El procedimiento iterativo de este método viene dado por (ver [36, 42])

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots \tag{2.13}$$

Aunque existen en la literatura muchas formas de diseñar el esquema clásico de Newton-Raphson, usemos el desarrollo en serie de Taylor (2.10) para deducir la expresión iterativa de este método. Sea $f : \mathbb{R} \to \mathbb{R}$ una función continua de clase C^2. $f(x)$ puede ser estimada por medio del polinomio de Taylor de primer grado alrededor de la raíz $\bar{x}$,

$$f(x) \approx f(\bar{x}) + f'(\bar{x})(x - \bar{x}) = f'(\bar{x})(x - \bar{x}).$$

El término $x - \bar{x}$ puede ser despejado como

$$x - \bar{x} \approx \frac{f(x)}{f'(\bar{x})},$$

y se puede obtener una nueva estimación x_{k+1} de $\bar{x}$, asumiendo a la derecha de la aproximación que $x = x_k$ y $f'(\bar{x}) \approx f'(x_k)$, nos queda el siguiente esquema iterativo

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots$$

El siguiente resultado proporciona una demostración del orden de convergencia del método de Newton-Raphson escalar.

Teorema 2.2.1 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo un cero $\bar{x} \in \mathbb{R}$ de f. Supongamos que $f'(x)$ es continua y no nula en $\bar{x}$. Si una estimación inicial x_0 es suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del método de Newton-Raphson (ver ecuación (2.13)) converge cuadráticamente a $\bar{x}$, siendo su ecuación del error:*

$$e_{k+1} = C_2 e_k^2 + O\left(e_k^3\right). \tag{2.14}$$

Demostración. Usando el Teorema 2.1.1, el desarrollo en serie de Taylor de $f(x_k)$ y $f'(x_k)$ alrededor de $\bar{x}$ se puede expresar como

$$f(x_k) = f'(\bar{x})\left[e_k + C_2 e_k^2\right] + O\left(e_k^3\right), \tag{2.15}$$

y

$$f'(x_k) = f'(\bar{x})\left[1 + 2C_2 e_k\right] + O\left(e_k^3\right),$$

respectivamente, donde $C_j = \dfrac{f^{(j)}(\bar{x})}{j! f'(\bar{x})}$, para $j \geq 2$.

De esta manera,

$$\frac{f(x_k)}{f'(x_k)} = e_k - C_2 e_k^2 + O\left(e_k^3\right).$$

Sean $x_{k+1} = e_{k+1} + \bar{x}$ y $x_k = e_k + \bar{x}$, entonces

$$e_{k+1} + \bar{x} = e_k + \bar{x} - e_k + C_2 e_k^2 + O\left(e_k^3\right).$$

Por tanto,

$$e_{k+1} = C_2 e_k^2 + O\left(e_k^3\right).$$

Esto completa la prueba. □

2.2.2. Método de Newton-Raphson vectorial

El método de Newton-Raphson escalar se puede extender a su versión vectorial, cuyo orden de convergencia también es cuadrático; el concepto de optimalidad de un método para sistemas no está definido. El esquema iterativo es (ver [36, 42])

$$x^{(k+1)} = x^{(k)} - \left[F'\left(x^{(k)}\right)\right]^{-1} F\left(x^{(k)}\right), \quad k = 0, 1, 2, \ldots, \tag{2.16}$$

donde $F'\left(x^{(k)}\right)$ es la matriz Jacobiana asociada a la función multidimensional no lineal, evaluada en el iterado $x^{(k)}$.

El siguiente resultado proporciona una demostración del orden de convergencia del método de Newton-Raphson vectorial.

Teorema 2.2.2 *Sea $F : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ una función suficientemente diferenciable en el conjunto convexo abierto D, conteniendo un cero $\tilde{x} \in \mathbb{R}^n$ de la función vectorial $F(x)$. Supongamos que la matriz Jacobiana $F'(x)$ es continua y no singular en $\tilde{x}$. Si una estimación inicial $x^{(0)} \in \mathbb{R}^n$ es suficientemente cercana a $\tilde{x}$, entonces el orden de convergencia local del método de Newton-Raphson (ver ecuación (2.16)) es dos, y la correspondiente ecuación del error es:*

$$e^{(k+1)} = \hat{C}_2 {e^{(k)}}^2 + O\left({e^{(k)}}^3\right). \tag{2.17}$$

Demostración. Usando la ecuación (2.12), los desarrollos de Taylor de $F\left(x^{(k)}\right)$ y $F'\left(x^{(k)}\right)$ alrededor de $\tilde{x}$ se pueden expresar como

$$F\left(x^{(k)}\right) = F'(\tilde{x})\left[e^{(k)} + \hat{C}_2 {e^{(k)}}^2\right] + O\left({e^{(k)}}^3\right) \tag{2.18}$$

y

$$F'\left(x^{(k)}\right) = F'(\tilde{x})\left[I + 2\hat{C}_2 e^{(k)}\right] + O\left({e^{(k)}}^2\right),$$

respectivamente, donde $\hat{C}_q = \frac{1}{q!}\left[F'(\tilde{x})\right]^{-1} F^{(q)}(\tilde{x})$ para $q \geq 2$.

Podemos plantear el desarrollo de Taylor de $\left[F'\left(x^{(k)}\right)\right]^{-1}$ como

$$\left[F'\left(x^{(k)}\right)\right]^{-1} = \left[I + X_2 e^{(k)}\right]\left[F'(\tilde{x})\right]^{-1} + O\left({e^{(k)}}^2\right),$$

donde X_2 es una variable desconocida tal que $\left[F'\left(x^{(k)}\right)\right]^{-1} F'\left(x^{(k)}\right) = I_n$, siendo I_n una matriz identidad de orden n. Entonces,

$$\left(2\hat{C}_2 + X_2\right) e^{(k)} = \hat{0},$$

de donde resulta que

$$X_2 = -2\hat{C}_2.$$

De este modo,

$$\left[F'\left(x^{(k)}\right)\right]^{-1} = \left[I - 2\hat{C}_2 e^{(k)}\right]\left[F'(\tilde{x})\right]^{-1} + O\left({e^{(k)}}^2\right).$$

Por tanto,

$$\left[F'\left(x^{(k)}\right)\right]^{-1} F\left(x^{(k)}\right) = e^{(k)} - \hat{C}_2 {e^{(k)}}^2 + O\left({e^{(k)}}^3\right).$$

Sean $x^{(k+1)} = e^{(k+1)} + \tilde{x}$ y $x^{(k)} = e^{(k)} + \tilde{x}$,

$$e^{(k+1)} + \tilde{x} = \tilde{x} + e^{(k)} - e^{(k)} + \hat{C}_2 {e^{(k)}}^2 + O\left({e^{(k)}}^3\right).$$

Finalmente,

$$e^{(k+1)} = \hat{C}_2 {e^{(k)}}^2 + O\left({e^{(k)}}^3\right).$$

Esto completa la prueba. □

En los siguientes métodos se proporcionan sólo los enunciados de los resultados sobre la convergencia de los restantes métodos conocidos que aparecerán a lo largo de la memoria.

2.2.3. Método de Traub escalar

En (Teorema 8-1, [42]), Traub demuestra que, sin considerar la estructura de las funciones iteradas involucradas, y para una multiplicidad $m = 1$ del cero $\bar{x}$, si una función de punto fijo $\phi(x)$ es de orden p, entonces la función iterada

$$\psi(x) = \phi(x) - \frac{f[\phi(x)]}{f'(x)} \tag{2.19}$$

es de orden $p + 1$.

Observación 2.2.1 *Cuando $\phi(x) = x$, tenemos el método de Newton-Raphson. Cuando $\phi(x)$ es el esquema de Newton-Raphson, tenemos el procedimiento de Traub que se describe a continuación.*

El método de Traub es un esquema multipunto y sin memoria que usa al procedimiento de Newton-Raphson como predictor. Al igual que el método de Newton-Raphson (2.13), el esquema de Traub usa la primera derivada de la función no lineal, pero su orden de convergencia es cúbico (no es un método óptimo). El esquema iterativo viene dado por (ver [36, 42])

$$x_{k+1} = y_k - \frac{f(y_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots, \tag{2.20}$$

donde

$$y_k = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots$$

El siguiente resultado enuncia el orden de convergencia del método de Traub.

Teorema 2.2.3 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo un cero $\bar{x} \in \mathbb{R}$ de f. Supongamos que $f'(x)$ es continua y no nula en $\bar{x}$. Si una estimación inicial x_0 es suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del método de Traub (ver ecuación (2.20)) converge cúbicamente a la raíz, siendo su ecuación del error:*

$$e_{k+1} = 2C_2^2 e_k^3 + O\left(e_k^4\right). \tag{2.21}$$

2.2.4. Método de Chun-Kim escalar

El método de Chun y Kim es un procedimiento multipunto y sin memoria que usa al esquema de Newton-Raphson como predictor. Al igual que el método de Newton-Raphson (2.13), el procedimiento de Chun y Kim usa la primera derivada de la función no lineal, pero su orden de convergencia es cúbico (no es un método óptimo). El esquema iterativo viene dado por (ver [14, 36])

$$x_{k+1} = x_k - \frac{1}{2}\left[3 - \frac{f'(y_k)}{f'(x_k)}\right]\frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots, \tag{2.22}$$

donde

$$y_k = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots$$

El siguiente resultado enuncia el orden de convergencia del método de Chun y Kim.

Teorema 2.2.4 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo un cero $\bar{x} \in \mathbb{R}$ de f. Supongamos que $f'(x)$ es continua y no nula en $\bar{x}$. Si una estimación inicial x_0 es suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del método de Chun y Kim (ver ecuación (2.22)) converge cúbicamente a la raíz, siendo su ecuación del error:*

$$e_{k+1} = \left(2C_2^2 + \frac{1}{2}C_3\right) e_k^3 + O\left(e_k^4\right). \tag{2.23}$$

2.2.5. Método de Ostrowski escalar

El método de Ostrowski es un esquema multipunto y sin memoria que usa al procedimiento de Newton-Raphson como predictor. Al igual que el método de Traub (2.20), el esquema de Ostrowski usa la primera derivada de la función no lineal, pero su orden de convergencia es cuatro (es un método óptimo). Este procedimiento pertenece a una familia uniparamétrica de métodos de King (ver [25, 36]). El esquema iterativo de esta familia viene dado por

$$x_{k+1} = y_k - \frac{f(x_k) + \kappa f(y_k)}{f(x_k) + (\kappa - 2)f(y_k)} \frac{f(y_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots, \tag{2.24}$$

siendo

$$y_k = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots$$

Cuando $\kappa = 0$, obtenemos el método de Ostrowski con el siguiente esquema iterativo (ver [36, 42]):

$$x_{k+1} = y_k - \frac{f(x_k)}{f(x_k) - 2f(y_k)} \frac{f(y_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots, \tag{2.25}$$

donde

$$y_k = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots$$

El siguiente resultado enuncia el orden de convergencia del método de Ostrowski.

Teorema 2.2.5 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo un cero $\bar{x} \in \mathbb{R}$ de f. Supongamos que $f'(x)$ es continua y no nula en $\bar{x}$. Si una estimación inicial x_0 es suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del método de Ostrowski (ver ecuación (2.25)) aproxima la raíz con orden de convergencia cuatro, siendo su ecuación del error:*

$$e_{k+1} = \left(C_2^3 - C_2 C_3\right) e_k^4 + O\left(e_k^5\right). \tag{2.26}$$

2.2.6. Método de Chun escalar

El método de Chun, al igual que el de Ostrowski (2.23), es también un esquema multipunto y sin memoria, y que usa al procedimiento de Newton-Raphson como predictor. Éste también usa la primera derivada de la

función no lineal, y su orden de convergencia es también cuatro (es un método óptimo). Este procedimiento también pertenece a una familia uniparamétrica de métodos de King (ver [25, 36]). Cuando $\kappa = 2$ en (2.22), obtenemos el método de Chun con el siguiente esquema iterativo (ver [36]):

$$x_{k+1} = y_k - \frac{f(x_k) + 2f(y_k)}{f(x_k)} \frac{f(y_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots, \tag{2.27}$$

donde

$$y_k = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots$$

El siguiente resultado enuncia el orden de convergencia del método de Chun.

Teorema 2.2.6 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo un cero $\bar{x} \in \mathbb{R}$ de f. Supongamos que $f'(x)$ es continua y no nula en $\bar{x}$. Si una estimación inicial x_0 es suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del método de Chun (ver ecuación (2.27)) aproxima la raíz con orden de convergencia cuatro, siendo su ecuación del error:*

$$e_{k+1} = \left(5C_2^3 - C_2C_3\right) e_k^4 + O\left(e_k^5\right). \tag{2.28}$$

2.2.7. Método de Steffensen escalar

El método de Steffensen, al igual que el de Newton-Raphson es de orden de convergencia cuadrático (es un método óptimo) y sin memoria, pero con la ventaja de que es libre de derivadas. El esquema iterativo viene dado por (ver [36, 42])

$$x_{k+1} = x_k - \frac{[f(x_k)]^2}{f(x_k + f(x_k)) - f(x_k)}, \quad k = 0, 1, 2, \ldots \tag{2.29}$$

El siguiente resultado enuncia el orden de convergencia del método de Steffensen.

Teorema 2.2.7 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo un cero $\bar{x} \in \mathbb{R}$ de f. Si una estimación inicial x_0 es suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del método de Steffensen (ver ecuación (2.29)) aproxima la raíz con orden de convergencia dos, siendo su ecuación del error:*

$$e_{k+1} = (1 + f'(\bar{x}))\, C_2 e_k^2 + O\left(e_k^3\right). \tag{2.30}$$

2.2.8. Método de la Secante escalar

El método de la Secante, al igual que el de Steffensen es un procedimiento libre de derivadas pero con memoria, y con orden de convergencia superlineal (no es un método óptimo). El esquema iterativo viene dado por (ver [36, 42])

$$x_{k+1} = x_k - \frac{(x_k - x_{k-1}) f(x_k)}{f(x_k) - f(x_{k-1})}, \quad k = 0, 1, 2, \ldots \tag{2.31}$$

El siguiente resultado enuncia el orden de convergencia del método de la Secante.

Teorema 2.2.8 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo un cero $\bar{x} \in \mathbb{R}$ de f. Si las estimaciones iniciales x_0 y x_{-1} son suficientemente cercanas a $\bar{x}$, entonces el orden de convergencia local del método de la Secante (ver ecuación (2.31) aproxima la raíz con orden de convergencia 1.618, siendo su ecuación del error:*

$$e_{k+1} = C_2 e_k e_{k-1} + O\left(e_k e_{k-1}^2\right). \tag{2.32}$$

Usando el Teorema 2.1.1, el polinomio característico que se obtiene es $s^2 - s - 1 = 0$, cuya única raíz positiva es $s \approx 1.618$.

2.3. Cálculo fraccionario y planos de convergencia/estabilidad

En esta sección, primero vamos a introducir algunas funciones especiales del cálculo fraccionario, como la función Gamma y la función de Mittag-Leffler [2]. También se va a tratar el concepto de derivada fraccionaria, en algunas de sus variantes que se utilizarán más adelante en la memoria: derivada fraccionaria de Caputo y de Riemann-Liouville [7, 33, 37], y derivada fraccionaria conformable [1, 23]. Incluiremos también las series de Taylor fraccionarias que usaremos [1, 7, 22, 32], el teorema generalizado del binomio [19] y el coeficiente binomial fraccionario [2]. También introduciremos el concepto de producto/potencia de Hadamard, necesario para el método de Newton-Raphson vectorial fraccionario que veremos en esta memoria [20]. Por último, trataremos algunos conceptos de la dinámica discreta real, donde el foco de atención estará en los planos de convergencia [29], los cuales serán una de las principales herramientas en los siguientes capítulos para analizar la dependencia de la convergencia de los métodos de la estimación inicial empleada.

2.3.1. Algunas funciones especiales

Una función especial es una generalización de una o más funciones elementales. A continuación presentamos la función Gamma y la de Mittag-Leffler.

La función Gamma es una generalización de la función factorial al plano complejo. Para todo $z \in \mathbb{C}\backslash\{0, -1, -2, \dots\}$, la función Gamma, denotada por $\Gamma(z)$, se define como

$$\Gamma(z) = \begin{cases} \displaystyle\int_0^\infty t^{z-1} e^{-t} dt, & \text{si } Re(z) > 0, \\ \dfrac{\Gamma(z+1)}{z}, & \text{si } Re(z) \leq 0, \ \ z \neq 0, -1, -2, \dots \end{cases} \tag{2.33}$$

En esta memoria usaremos la función Gamma restringida al conjunto de los números reales. Algunas propiedades de ésta son [2]

$$\Gamma(n) = (n-1)!, \ \ n \in \mathbb{N}, \tag{2.34}$$

$$\Gamma(z+1) = z\Gamma(z), \tag{2.35}$$

$$\Gamma(1/2) = \sqrt{\pi}. \tag{2.36}$$

En la Figura 2.1 se puede ver la gráfica de la función Gamma (ver fichero MATLAB en el Anexo A.3).

La función de Mittag-Leffler, denotada por $E_\mu(z)$, es una generalización de algunas funciones trascendentes (exponencial, trigonométricas, etc.). Esta función depende de un parámetro real μ, y se define como

$$E_\mu(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\mu k + 1)}, \quad \mu > 0, \ \ \mu \in \mathbb{R}, \ \ z \in \mathbb{C}. \tag{2.37}$$

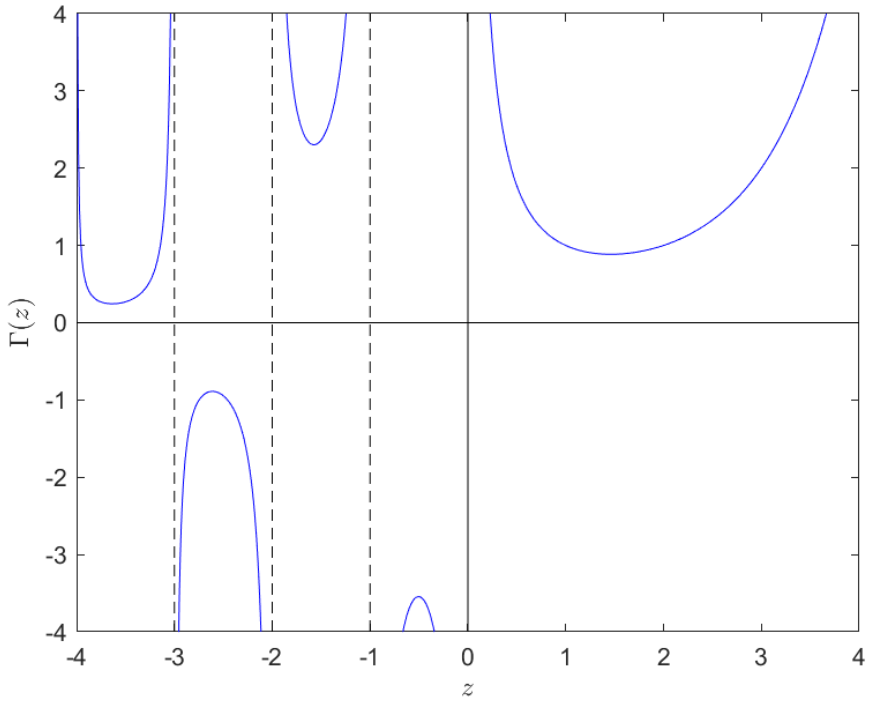


Figura 2.1: Función Gamma para valores reales de z

Esta función generalizada para dos parámetros, μ y σ, denotada por $E_{\mu,\sigma}(z)$, se define como

$$E_{\mu,\sigma}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\mu k + \sigma)}, \quad \mu, \sigma > 0, \;\; \mu, \sigma \in \mathbb{R}, \;\; z \in \mathbb{C}. \tag{2.38}$$

Observación 2.3.1 *La función de Mittag-Leffler también se denota por $E_{\mu,1}(z)$, es decir, es un caso particular de la ecuación (2.38) cuando $\sigma = 1$.*

Algunas propiedades de la función de Mittag-Leffler son [2]

$$E_0(z) = \sum_{k=o}^{\infty} z^k = \frac{1}{1-z}, \;\; |z| < 1, \tag{2.39}$$

$$E_1(z) = \sum_{k=o}^{\infty} \frac{z^k}{k!} = e^z, \tag{2.40}$$

$$E_2\left(-z^2\right) = cos(z). \tag{2.41}$$

En la Figura 2.2 se pueden ver las gráficas de la función de Mittag-Leffler para $\mu = 1, 1.5, 2$ (ver fichero MATLAB en el Anexo A.4).

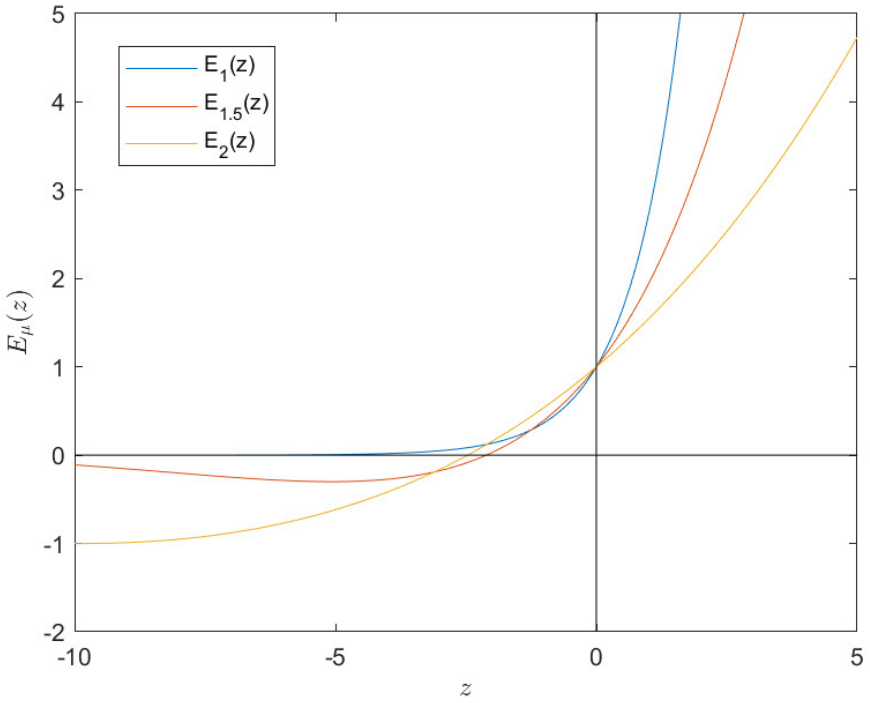


Figura 2.2: Función de Mittag-Leffler para valores reales de z

2.3.2. Operadores fraccionarios

Un operador fraccionario, $D^u f(t), u \in \mathbb{R}$, será llamado operador fraccionario diferencial o integral si cumple con las siguientes condiciones [18, 21, 43]:

1. $D^n f(t) = \dfrac{d^n}{dt^n} f(t), \quad n \in \mathbb{N}.$
   Es decir, si $u = n \in \mathbb{N}$, coincide con la *n-ésima* derivada clásica.
2. $D^{-n} f(t) = \underbrace{\int \int \cdots \int}_{\text{n veces}} f(t) dt dt \cdots dt, \quad n \in \mathbb{N}.$
   Es decir, si $u = n \in \mathbb{N}$, coincide con la *n-ésima* integral clásica.
3. $D^0 f(t) = f(t).$
   Es decir, si $u = 0$, coincide con la propia función, al igual que en el cálculo clásico.
4. $D^u \left[D^v f(t)\right] = D^{u+v} f(t), \quad u, v \in \mathbb{R}.$
   Es decir, cumple con la propiedad de semigrupo de los operadores integrodiferenciales fraccionarios. Esto significa que el operador D es conmutativo y aditivo.

Una vez establecidas las condiciones de un operador fraccionario, el operador de Riemann-Liouville, o derivada fraccionaria de Riemann-Liouville se define como:

Definición 2.3.1 *Sean $\alpha > 0$, α, a, $t \in \mathbb{R}$, entonces*

$$D^{\alpha}_{a^+} f(t) := \begin{cases} \dfrac{1}{\Gamma(m-\alpha)} \dfrac{d^m}{dt^m} \displaystyle\int_a^t \frac{f(\tau)}{(t-\tau)^{\alpha+1-m}} d\tau, & m-1 < \alpha < m \in \mathbb{N}, \\ \dfrac{d^m}{dt^m} f(t), & \alpha = m, \end{cases} \tag{2.42}$$

es la derivada fraccionaria de Riemann-Liouville de orden α, centrada en a, $a < t$, y Γ es la función Gamma. El siguiente resultado nos proporciona el desarrollo en serie de Taylor de una función f utilizando derivadas fraccionarias de este tipo.

Teorema 2.3.1 *([22]) Asumamos que la fución continua $f : \mathbb{R} \to \mathbb{R}$ tiene derivadas fraccionarias de orden $k\alpha$, para cualquier entero positivo k y cualquier α, $0 < \alpha \leq 1$, entonces se cumple la siguiente igualdad,*

$$f(x+h) = \sum_{k=0}^{\infty} \frac{h^{\alpha k}}{\Gamma(\alpha k + 1)} D_{a^+}^{\alpha k} f(x), \tag{2.43}$$

donde $D_{a^+}^{\alpha k} f(x)$ es la derivada de Riemann-Liouville de orden αk de $f(x)$.

A continuación definimos el operador diferencial de Caputo, o la derivada fraccionaria de Caputo, denotada por ${}_cD_{a^+}^{\alpha} f(t)$:

Definición 2.3.2 *Sean $\alpha > 0$, α, a, $t \in \mathbb{R}$, entonces*

$${}_cD_{a^+}^{\alpha} f(t) \coloneqq \begin{cases} \dfrac{1}{\Gamma(m-\alpha)} \displaystyle\int_a^t \frac{f^{(m)}(\tau)}{(t-\tau)^{\alpha+1-m}} d\tau, & \alpha \notin \mathbb{N}, \\ \dfrac{d^m}{dt^m} f(t), & \alpha = m \in \mathbb{N} \cup \{0\}, \end{cases} \tag{2.44}$$

es la derivada fraccionaria de Caputo de orden α, centrada en a, $a < t$, y Γ es la función Gamma. Para los propósitos de esta memoria usaremos $m = 1$ y $a = 0$ (tanto para la derivada de Riemann-Liouville, como para la de Caputo). El siguiente resultado nos proporciona el desarrollo en serie de Taylor de una función f utilizando derivadas fraccionarias de este tipo.

Teorema 2.3.2 *([7, 32]) Supongamos que ${}_cD_{a^+}^{j\alpha} f(x) \in C([a,b])$ para $j = 1, 2, \ldots, n+1$ donde $\alpha \in (0, 1]$, entonces tenemos*

$$f(x) = \sum_{i=0}^{n} {}_cD_{a^+}^{i\alpha} f(a) \frac{(x-a)^{i\alpha}}{\Gamma(i\alpha+1)} + {}_cD_{a^+}^{(n+1)\alpha} f(\xi) \frac{(x-a)^{(n+1)\alpha}}{\Gamma((n+1)\alpha+1)}, \tag{2.45}$$

con $a \leq \xi \leq x$, para todo $x \in (a, b]$ donde ${}_cD_a^{n\alpha} = {}_cD_a^{\alpha} \cdot {}_cD_a^{\alpha} \cdots {}_cD_a^{\alpha}$ (n veces).

Algunas propiedades de las derivadas de Riemann-Liouville y de Caputo se resumen a continuación:

(I) ${}_cD^{\alpha} f(t) \neq D^{\alpha} f(t)$ (éstas no coinciden)

(II) ${}_cD^{\alpha} K = 0$, donde K es una constante

(III) $D^{\alpha} K \neq 0$, donde K es una constante

(IV) ${}_cD^{\alpha}(\lambda f(t) + g(t)) = \lambda\, {}_cD^{\alpha} f(t) + {}_cD^{\alpha} g(t)$ (linealidad)

(V) $D^{\alpha}(\lambda f(t) + g(t)) = \lambda D^{\alpha} f(t) + D^{\alpha} g(t)$ (linealidad)

(VI) ${}_cD^{u}\, {}_cD^{v} f(t) = {}_cD^{u+v} f(t) = {}_cD^{v+u} f(t) = {}_cD^{v}\, {}_cD^{u} f(t)$ (conmutatividad)

(VII) $D^{u} D^{v} f(t) = D^{u+v} f(t) = D^{v+u} f(t) = D^{v} D^{u} f(t)$ (conmutatividad)

(VIII) ${}_cD^{u} D^{v} f(t) \neq D^{v}\, {}_cD^{u} f(t)$ (no conmutan entre sí)

(IX) $cD^{\alpha}f(t) = D^{\alpha}f(t) - \sum_{k=0}^{m-1} \frac{t^{k-\alpha}}{\Gamma(k+1-\alpha)} f^{(k)}(0)$

En la Tabla 2.1 se resumen las derivadas fraccionarias de algunas funciones básicas:

$f(t)$	$cD^{\alpha}f(t)$	$D^{\alpha}f(t)$
$K(constante)$	0	$\frac{K}{\Gamma(1-\alpha)}t^{-\alpha}$
t^p	$\frac{\Gamma(p+1)}{\Gamma(p-\alpha+1)}t^{p-\alpha}, \quad m-1<\alpha<m, p\in\mathbb{R}$	Igual a $cD^{\alpha}t^p$
$e^{\lambda t}$	$\lambda^m t^{m-\alpha}E_{1,m-\alpha+1}(\lambda t)$	Igual a $cD^{\alpha}e^{\lambda t}$
$\sin\lambda t$	$-\frac{1}{2}i(i\lambda)^m t^{m-\alpha}\left(E_{1,m-\alpha+1}(i\lambda t)-(-1)^m E_{1,m-\alpha+1}(-i\lambda t)\right)$	Igual a $cD^{\alpha}\sin\lambda t$
$\cos\lambda t$	$\frac{1}{2}i(i\lambda)^m t^{m-\alpha}\left(E_{1,m-\alpha+1}(i\lambda t)+(-1)^m E_{1,m-\alpha+1}(-i\lambda t)\right)$	Igual a $cD^{\alpha}\cos\lambda t$

Tabla 2.1: Comparación entre algunas derivadas de Caputo y Riemann-Liouville

Observación 2.3.2 *En la Propiedad (IX) se puede ver que cuando $f(t)=K$, $cD^{\alpha}K$ se anula debido a que $D^{\alpha}K = \frac{K}{\Gamma(1-\alpha)}t^{-\alpha}$.*

La derivada fraccionaria conformable es la definición más natural de derivadas fraccionarias. Ésta no requiere la evaluación de funciones especiales (como la función Gamma o de Mittag-Leffler), lo cual implica un bajo coste computacional comparado con las derivadas de Riemann-Liouville o de Caputo. A continuación se muestra una definición de ésta, denotada como $(T_{\alpha}^{a}f)(x)$.

Definición 2.3.3 *La derivada fraccionaria conformable por la izquierda (ver [1, 23]) centrada en a de una función $f:[a,\infty)\to\mathbb{R}$ de orden $\alpha\in(0,1]$, $\alpha, a, x\in\mathbb{R}$, $a<x$, es definida como*

$$(T_{\alpha}^{a}f)(x) = \lim_{\varepsilon\to 0}\frac{f(x+\varepsilon(x-a)^{1-\alpha})-f(x)}{\varepsilon}. \tag{2.46}$$

Si este límite existe, se dice que f es α-diferenciable. Además, si f es diferenciable, entonces $(T_{\alpha}^{a}f)(x) = (x-a)^{1-\alpha}f'(x)$. Si f es α-diferenciable en (a,b), para algún $b\in\mathbb{R}$, $(T_{\alpha}^{a}f)(a) = \lim_{x\to a^{+}}(T_{\alpha}^{a}f)(x)$.

La derivada fraccionaria conformable preserva la propiedad de derivadas no fraccionarias, $T_{\alpha}^{a}C=0$, siendo C una constante. El siguiente resultado proporciona una serie de Taylor de $f(x)$ con derivada fraccionaria conformable.

Teorema 2.3.3 *([1]) Sea $f(x)$ una función infinitamente α-diferenciable para $\alpha\in(0,1]$, en el entorno de x_0 con derivada conformable centrada en x_0. La serie de potencias conformable para $f(x)$ es:*

$$f(x) = \sum_{k=0}^{\infty}\frac{(T_{\alpha}^{x_0}f)^{(k)}(x_0)(x-x_0)^{k\alpha}}{\alpha^k k!}, \quad x_0<x<x_0+R^{1/\alpha}, R>0, \tag{2.47}$$

donde $(T_{\alpha}^{x_0}f)^{(k)}(x_0)$ es la derivada fraccionaria conformable aplicada k veces.

En (2.47), el punto donde se centra la derivada conformable y se evalúa la misma es x_0; sin embargo, es posible definir una serie de Taylor para $f(x)$, donde las derivadas conformables están centradas en un punto diferente de el punto en el que son evaluadas, lo que es más conveniente para nuestros propósitos.

Teorema 2.3.4 *([40]) Sea f una función infinitamente α-diferenciable para $\alpha \in (0,1]$, en el entorno de b con derivada conformable centrada en a. La serie de potencias conformable para $f(x)$ es:*

$$f(x) = f(b) + \frac{(T_\alpha^a f)(b)\delta_1}{\alpha} + \frac{(T_\alpha^a f)^{(2)}(b)\delta_2}{2\alpha^2} + R_2(x,b,a), \tag{2.48}$$

siendo $\delta_1 = H^\alpha - L^\alpha$, $\delta_2 = H^{2\alpha} - L^{2\alpha} - 2L^\alpha\delta_1$, ..., y $H = x - a$, $L = b - a$.

Es fácil probar que $\delta_2 = \delta_1^2$, $\delta_3 = \delta_1^3$, etc. Por tanto, la serie de Tayor (2.48) se puede escribir como

$$f(x) = f(b) + \frac{(T_\alpha^a f)(b)\delta_1}{\alpha} + \frac{(T_\alpha^a f)^{(2)}(b)\delta_1^2}{2\alpha^2} + R_2(x,b,a), \tag{2.49}$$

En [6] se puede encontrar una definición de derivada parcial conformable como se muestra a continuación:

Definición 2.3.4 *Sea $f : \mathbb{R}^n \to \mathbb{R}$ una función en n variables, $x_1, \ldots, x_n$, la derivada parcial conformable de f de orden $\alpha \in (0,1]$ en $x_i > a = 0$ es definida como:*

$$\frac{\partial_0^\alpha}{\partial x_i^\alpha} f(x_1, \ldots, x_n) = \lim_{\epsilon \to 0} \frac{f(x_1, \ldots, x_i + \epsilon x_i^{1-\alpha}, \ldots, x_n) - f(x_1, \ldots, x_n)}{\epsilon}. \tag{2.50}$$

Más adelante se verá que a puede ser considerado no nulo.

Utilizando la definición previa de derivada parcial conformable, en [6] también es definida la matriz Jacobiana conformable como:

Definición 2.3.5 *Sean f, g funciones en dos variables, x e y, y sus respectivas derivadas parciales existen y son continuas, donde $x > a_1$ e $y > a_2$, siendo $a = (a_1, a_2) = (0,0) = \hat{0}$, entonces la matriz Jacobiana conformable está dada por:*

$$\begin{pmatrix} \frac{\partial_0^\alpha f}{\partial x^\alpha} & \frac{\partial_0^\alpha f}{\partial y^\alpha} \\ \frac{\partial_0^\alpha g}{\partial x^\alpha} & \frac{\partial_0^\alpha g}{\partial y^\alpha} \end{pmatrix} = \begin{pmatrix} x^{1-\alpha}\frac{\partial f}{\partial x} & y^{1-\alpha}\frac{\partial f}{\partial y} \\ x^{1-\alpha}\frac{\partial g}{\partial x} & y^{1-\alpha}\frac{\partial g}{\partial y} \end{pmatrix}. \tag{2.51}$$

Esto se puede extender directamente a dimensiones superiores, y como se verá más adelante, a puede ser considerado no nulo.

2.3.3. Teorema del binomio y coeficiente binomial generalizados

El teorema del binomio de Newton para una potencia $(x+y)^n$, $x, y \in \mathbb{R}$, $n \in \mathbb{N}$, se puede generalizar para una potencia [19]

$$(x+y)^r = \sum_{k=0}^{+\infty} \binom{r}{k} x^{r-k} y^k, \quad k \in \{0\} \cup \mathbb{N},\ r \in \mathbb{R}, \tag{2.52}$$

donde $\binom{r}{k}$ es el coeficiente binomial fraccionario, el cual es una generalización del coeficiente binomial clásico $\binom{n}{k}$. Éste se define en [2] como

$$\binom{r}{k} = \frac{\Gamma(r+1)}{k!\Gamma(r-k+1)}, \tag{2.53}$$

donde $r \in \mathbb{R}$ y $k \in \{0\} \cup \mathbb{N}$. Notemos que en el caso clásico $r = n \in \mathbb{N}$, y $\Gamma(n+1) = n!$.

2.3.4. Producto/potencia de Hadamard

Otro concepto que utilizaremos es el producto de Hadamard [20]:

Definición 2.3.6 *Sean $A = (a_{ij})_{m\times n}$ y $B = (b_{ij})_{m\times n}$ matrices de $m \times n$. El producto de Hadamard se define como $A \odot B \coloneqq (a_{ij}b_{ij})_{m\times n}$.*

Observación 2.3.3 *Un concepto análogo al producto de Hadamard es la potencia de Hadamard, donde $A^{\odot r} = \underbrace{A \odot A \odot \cdots \odot A}_{r \text{ veces}}$, siendo $r \in \mathbb{R}$.*

2.3.5. Planos de convergencia

Los planos de convergencia son una herramienta de la dinámica real que nos permite estudiar la dependencia, en términos de convergencia, de un método iterativo (que depende de un solo parámetro) a las estimaciones iniciales (véase [29]).

Un plano de convergencia asocia cada punto del plano con una estimación inicial y un valor del parámetro. Éste se construye considerando los valores de las estimaciones iniciales en el eje horizontal, y los valores del parámetro en el eje vertical. Cada punto del mallado corresponde, por tanto, a un método concreto de la familia paramétrica de métodos iterativos y a la estimación inicial con el que se itera dicho esquema. Se le asigna el color negro si dicho método no converge a la solución del problema en un máximo de iteraciones, y otros colores en función de la solución a la que haya convergido.

Para estudiar la estabilidad de los métodos que se proponen en esta memoria usaremos los planos de convergencia. Como ejemplo, veamos la construcción de un plano de convergencia con el uso del método de Traub (2.22), pero con un parámetro $\theta \in (0, 1]$ en el esquema, como se muestra a continuación:

$$x_{k+1} = y_k - \theta\frac{f(y_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots, \tag{2.54}$$

donde

$$y_k = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots$$

Consideremos el problema $f(x) = x^2 - 1 = 0$, cuyas raíces son $\bar{x}_1 = -1$ y $\bar{x}_2 = 1$. El correspondiente plano de convergencia se muestra en la Figura 2.3 (ver fichero MATLAB en el Anexo A.5):

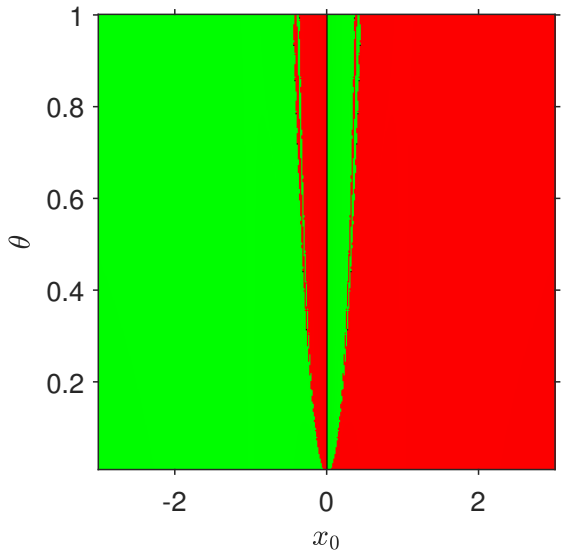


Figura 2.3: Método de Traub paramétrico

En la Figura 2.3 se pueden observar los pares ordenados (x_0, θ) que convergieron a $\bar{x}_1$ (en verde) y a $\bar{x}_2$ (en rojo).

Hemos visto los conceptos teóricos, los métodos clásicos que se trabajan en esta memoria, y las herramientas del cálculos fraccionario sobre los que nos apoyaremos para el diseño de nuevos métodos fraccionarios.

En el siguiente capítulo, veremos las variantes fraccionarias de tipo Newton-Raphson escalares que se han diseñado hasta el momento, haremos el análisis de convergencia de cada uno, y estudiaremos la estabilidad de dichos métodos.

Capítulo 3

Métodos iterativos fraccionarios escalares de tipo Newton-Raphson

"Algún matemático dijo que el verdadero placer no reside en el descubrimiento de la verdad, sino en su búsqueda"

Leo Tolstoy (1828 - 1910)

En este capítulo se tratan los primeros métodos iterativos de tipo Newton-Raphson con derivadas fraccionarias desarrollados en [3, 8]. También se presenta el diseño, análisis de convergencia y estudio de la estabilidad de nuevas variantes fraccionarias de tipo Newton-Raphson con derivadas de Caputo, Riemann-Liouville y conformable[1,2] . Finalmente, se proponen variantes de tipo Newton-Raphson con derivada conformable de orden de convergencia mayor a dos[3] .

3.1. Estado del arte

Estamos aproximando la solución $\bar{x} \in \mathbb{R}$ de la ecuación no lineal $f(x) = 0$ con $f : I \subseteq \mathbb{R} \to \mathbb{R}$. La resolución de problemas no lineales por medio de métodos con derivadas fraccionarias (o métodos fraccionarios) es una línea de investigación relativamente nueva. Una primera variante del esquema de Newton-Raphson con derivada fraccionaria ha sido implementada en [8] con el procedimiento iterativo

$$x_{k+1} = x_k - \frac{f(x_k)}{D^{\alpha}_{0^+} f(x_k)}, \quad k = 0, 1, 2, \ldots, \tag{3.1}$$

para $0 < \alpha < 2$, donde se sustituye la derivada de orden entero (la derivada clásica) por una de orden fraccionario de Riemann-Liouville. $D^{\alpha}_{0^+}$ es la derivada de Riemann-Liouville (2.42) de orden α y centrada en cero, $x_k > 0$, $\forall k$. En dicho artículo se hacen pruebas numéricas con polinomios reales que sólo tienen raíces complejas, pero no se realiza un análisis de convergencia de dicho método. Con pruebas numéricas, se

[1] Resultados parciales de este capítulo forman parte del artículo *Multipoint Fractional Iterative Methods with (2α+1)th-Order of Convergence for Solving Nonlinear Problems* publicado en Mathematics, MDPI [10], del Capítulo 5 del libro *Fractional-Order Modeling of Dynamic Systems with Application in Optimization, Signal Processing, and Control* publicado en Academic Press, Elsevier [38], y presentados en el congreso CEDYA CMA 2020, Gijón, España

[2] Resultados parciales de este capítulo forman parte del artículo *An optimal and low computational cost fractional Newton-type method for solving nonlinear equations* publicado en Applied Mathematics Letters, Elsevier [11], y presentados en los congresos MME&HB 2021, Valencia, España, e ICRAPAM 2021, Bodrum/Mugla, Turquía

[3] Los resultados de este capítulo forman parte del manuscrito *On a new technique to generate optimal multipoint fractional methods for solving nonlinear equations* enviado para su posible publicación a Journal of Mathematical Analysis and Applications, Elsevier [13]

busca mostrar que este esquema puede converger rápidamente a raíces complejas de polinomios reales con estimaciones iniciales reales, lo cual es una ventaja ante el procedimiento clásico de Newton-Raphson, el cual por lo general no converge a raíces complejas partiendo de una estimación inicial real. Aunque estos autores propusieron otras variantes de este mismo método, no se mencionan en esta memoria debido a que en estos casos tampoco se proporciona el orden de convergencia de tales esquemas.

Otras variantes fraccionarias del procedimiento de Newton-Raphson han sido desarrolladas en [3] con los métodos iterativos

$$x_{k+1} = x_k - \Gamma(\alpha+1)\frac{f(x_k)}{cD^{\alpha}_{0^+}f(x_k)}, \quad k = 0, 1, 2, \ldots, \tag{3.2}$$

donde $cD^{\alpha}_{0^+}$ es la derivada de Caputo (2.44) de orden α y centrada en cero, $x_k > 0$, $\forall k$, y Γ es la función Gamma (2.33), la cual es un parámetro amortiguador. Se prueba la convergencia de orden 2α, siendo

$$e^{\alpha}_{k+1} = \frac{\Gamma(2\alpha+1) - \Gamma^2(\alpha+1)}{\Gamma^2(\alpha+1)} C_2^{CA} e_k^{2\alpha} + O\left(e_k^{3\alpha}\right),$$

donde $\Gamma^n(\cdot) = (\Gamma(\cdot))^n$, $C_j^{CA} = \dfrac{\Gamma(\alpha+1)}{\Gamma(j\alpha+1)} \dfrac{cD^{j\alpha}_{0^+}f(\bar{x})}{cD^{\alpha}_{0^+}f(\bar{x})}$, y

$$x_{k+1} = x_k - \Gamma(\alpha+1)\frac{f(x_k)}{D^{\alpha}_{0^+}f(x_k)}, \quad k = 0, 1, 2, \ldots, \tag{3.3}$$

donde $D^{\alpha}_{0^+}$ es la derivada de Riemann-Liouville (2.42) de orden α y centrada en cero, $x_k > 0$, $\forall k$, y con ecuación del error

$$e^{\alpha}_{k+1} = \frac{\Gamma(2\alpha+1) - \Gamma^2(\alpha+1)}{\Gamma^2(\alpha+1)} C_2^{RL} e_k^{2\alpha} + O\left(e_k^{3\alpha}\right),$$

siendo $C_j^{RL} = \dfrac{\Gamma(\alpha+1)}{\Gamma(j\alpha+1)} \dfrac{D^{j\alpha}_{0^+}f(\bar{x})}{D^{\alpha}_{0^+}f(\bar{x})}$.

Se puede deducir de las respectivas ecuaciones del error que el orden de convergencia de estos métodos depende del orden α de la derivada, el cual es 2α en cada caso. En estos esquemas se introduce a $\Gamma(\alpha+1)$ como parámetro de amortiguamiento, lo cual permite, con el uso del desarrollo en series de Taylor de orden fraccionario (2.43) y (2.45), demostrar que el orden de convergencia es al menos 2α en cada caso. En estos procedimientos se escoge $a = 0$, ya que sólo así se puede obtener un orden de convergencia teórico que no sea lineal. Cuando $\alpha = 1$, se obtiene el método clásico de Newton-Raphson, y también su respectiva ecuación del error clásica en cada caso. En [3] se hacen pruebas numéricas, se estudia la dependencia de las estimaciones iniciales mediante la visualización de planos de convergencia, y se puede apreciar cómo estos esquemas convergen a múltiples raíces con una misma estimación inicial para $0 < \alpha \leq 1$. También se puede ver que el procedimiento de Newton-Raphson con derivada de Caputo muestra mejores propiedades numéricas que con derivada de Riemann-Liouville en términos de amplitud de las cuencas de atracción de las raíces.

En la siguiente sección, presentamos el diseño y análisis de convergencia de nuevas variantes fraccionarias de tipo Newton-Raphson con derivadas de Caputo y Riemann-Liouville.

3.2. Métodos de tipo Newton-Raphson con derivadas de Caputo y Riemann-Liouville

En [3] los autores desarrollaron dos esquemas de Newton-Raphson de orden 2α, en los cuales introdujeron un parámetro amortiguador y se sustituye la derivada clásica por las derivadas fraccionarias de Caputo y Riemann-Liouville. A continuación se muestra el diseño de nuevos métodos fraccionarios de tipo Newton-Raphson de orden $\alpha+1$ con estas mismas derivadas, los cuales fueron publicados en la revista *Mathematics* (ver [10]).

3.2.1. Deducción de los métodos

Para construir una nueva versión del esquema de Newton-Raphson con derivada fraccionaria de Caputo, usamos el desarrollo de Taylor (2.45) de la función no lineal cuya raíz estamos buscando. Sea $f:\mathbb{R}\to\mathbb{R}$ una función continua que tiene derivadas fraccionarias de Caputo de orden α y 2α, $0<\alpha\leq 1$, y centradas en cero, $x_k>0$, $\forall k$. Entonces, aplicando el Teorema 2.3.2, $f(x)$ puede ser estimada con un polinomio de Taylor de primer grado alrededor de la raíz $\bar{x}$,

$$f(x)\approx f(\bar{x})+cD^{\alpha}_{0^+}f(\bar{x})\frac{(x-\bar{x})^{\alpha}}{\Gamma(\alpha+1)}=cD^{\alpha}_{0^+}f(\bar{x})\frac{(x-\bar{x})^{\alpha}}{\Gamma(\alpha+1)}.$$

Por lo tanto, el término $x-\bar{x}$ puede ser despejado como

$$x-\bar{x}\approx\left(\Gamma(\alpha+1)\frac{f(x)}{cD^{\alpha}_{0^+}f(\bar{x})}\right)^{1/\alpha},$$

y se puede obtener una nueva estimación x_{k+1} de $\bar{x}$, asumiendo $x=x_k$ y $cD^{\alpha}_{0^+}f(\bar{x})\approx cD^{\alpha}_{0^+}f(x_k)$, por medio del esquema iterativo

$$x_{k+1}=x_k-\left(\Gamma(\alpha+1)\frac{f(x_k)}{cD^{\alpha}_{0^+}f(x_k)}\right)^{1/\alpha},\quad k=0,1,2,\ldots$$

Denotemos este procedimiento por NeCA. De manera análoga, pero usando el desarrollo de Taylor (2.43) del Teorema 2.3.1, podemos deducir una versión fraccionaria con derivada de Riemann-Liouville:

$$x_{k+1}=x_k-\left(\Gamma(\alpha+1)\frac{f(x_k)}{D^{\alpha}_{0^+}f(x_k)}\right)^{1/\alpha},\quad k=0,1,2,\ldots$$

Denotemos este método por NeRL. Al igual que en los esquemas (3.2) y (3.3), escogemos $a=0$, ya que sólo así es posible obtener un orden de convergencia teórico que no sea lineal. Cuando $\alpha=1$, se obtiene el procedimiento clásico de Newton-Raphson en cada caso.

3.2.2. Análisis de convergencia

En el siguiente resultado se establecen las condiciones que aseguran la convergencia del método NeCA con desarrollos de Taylor (véase (2.45)).

Teorema 3.2.1 *Sea $f:I\subseteq\mathbb{R}\to\mathbb{R}$ una función con derivada fraccionaria de Caputo de orden αk, para todo entero positivo k y para cualquier $\alpha\in(0,1]$, en el intervalo I conteniendo el cero $\bar{x}$ de $f(x)$. Supongamos*

que $cD_{0^+}^{\alpha} f(x)$ *es continua y no nula en* $\bar{x}$*. Si una aproximación inicial* x_0 *es lo suficientemente cercana a* $\bar{x}$*, entonces el orden de convergencia local del método de Newton-Raphson fraccionario con derivada de Caputo (NeCA)*

$$x_{k+1} = x_k - \left(\Gamma(\alpha+1)\frac{f(x_k)}{cD_{0^+}^{\alpha} f(x_k)}\right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots \tag{3.4}$$

es al menos $\alpha + 1$*, siendo* $0 < \alpha \leq 1$*, y la ecuación del error es*

$$e_{k+1} = \frac{\Gamma(2\alpha+1) - \Gamma^2(\alpha+1)}{\alpha\Gamma^2(\alpha+1)} C_2^{CA} e_k^{\alpha+1} + O\left(e_k^{2\alpha+1}\right),$$

siendo $(\Gamma(\cdot))^n$ *denotada por* $\Gamma^n(\cdot)$*, donde* $e_k = x_k - \bar{x}$ *y* $C_j^{CA} = \dfrac{\Gamma(\alpha+1)}{\Gamma(j\alpha+1)} \dfrac{cD_{0^+}^{j\alpha} f(\bar{x})}{cD_{0^+}^{\alpha} f(\bar{x})}$*,* $j \geq 2$*, tal que* $x_k > 0$*,* $\forall k$*.*

Demostración. Por medio del Teorema 2.3.2, el desarrollo de Taylor de $f(x)$ y su derivada de Caputo $cD_{0^+}^{\alpha} f(x_k)$ alrededor de $\bar{x}$ pueden ser expresadas por

$$f(x_k) = \frac{cD_{0^+}^{\alpha} f(\bar{x})}{\Gamma(\alpha+1)} \left[e_k^{\alpha} + C_2^{CA} e_k^{2\alpha} + C_3^{CA} e_k^{3\alpha}\right] + O\left(e_k^{4\alpha}\right)$$

y

$$cD_{0^+}^{\alpha} f(x_k) = \frac{cD_{0^+}^{\alpha} f(\bar{x})}{\Gamma(\alpha+1)} \left[\Gamma(\alpha+1) + \frac{\Gamma(2\alpha+1)}{\Gamma(\alpha+1)} C_2^{CA} e_k^{\alpha} + \frac{\Gamma(3\alpha+1)}{\Gamma(2\alpha+1)} C_3^{CA} e_k^{2\alpha}\right] + O\left(e_k^{3\alpha}\right),$$

respectivamente, siendo

$C_j^{CA} = \dfrac{\Gamma(\alpha+1)}{\Gamma(j\alpha+1)} \dfrac{cD_{0^+}^{j\alpha} f(\bar{x})}{cD_{0^+}^{\alpha} f(\bar{x})}$ para $j \geq 2$.

De este modo,

$$\frac{f(x_k)}{cD_{0^+}^{\alpha} f(x_k)} = \frac{1}{\Gamma(\alpha+1)} e_k^{\alpha} + \frac{\Gamma^2(\alpha+1) - \Gamma(2\alpha+1)}{\Gamma^3(\alpha+1)} C_2^{CA} e_k^{2\alpha} + O\left(e_k^{3\alpha}\right),$$

de donde se obtiene

$$\Gamma(\alpha+1)\frac{f(x_k)}{cD_{0^+}^{\alpha} f(x_k)} = e_k^{\alpha} + \frac{\Gamma^2(\alpha+1) - \Gamma(2\alpha+1)}{\Gamma^2(\alpha+1)} C_2^{CA} e_k^{2\alpha} + O\left(e_k^{3\alpha}\right).$$

Usando el Teorema generalizado del binomio (2.52) y el coeficiente binomial fraccionario (2.53),

$$\begin{aligned}\left(\Gamma(\alpha+1)\frac{f(x_k)}{cD_{0^+}^{\alpha} f(x_k)}\right)^{1/\alpha} &= \left(e_k^{\alpha} + \frac{\Gamma^2(\alpha+1) - \Gamma(2\alpha+1)}{\Gamma^2(\alpha+1)} C_2^{CA} e_k^{2\alpha} + O\left(e_k^{3\alpha}\right)\right)^{1/\alpha} \\ &= e_k + \frac{\Gamma(\alpha+1)}{1!\Gamma(\alpha)} e_k^{1-\alpha} \frac{\Gamma^2(\alpha+1) - \Gamma(2\alpha+1)}{\Gamma^2(\alpha+1)} C_2^{CA} e_k^{2\alpha} + O\left(e_k^{2\alpha+1}\right).\end{aligned}$$

Como $\Gamma(1/\alpha+1)=\dfrac{1}{\alpha}\Gamma(1/\alpha)$ (ver propiedad (2.35)), y simplificando

$$\left(\Gamma(\alpha+1)\frac{f(x_k)}{cD^{\alpha}_{0^+}f(x_k)}\right)^{1/\alpha}=e_k+\frac{\Gamma^2(\alpha+1)-\Gamma(2\alpha+1)}{\alpha\Gamma^2(\alpha+1)}C_2^{CA}e_k^{\alpha+1}+O\left(e_k^{2\alpha+1}\right).$$

Sean $x_{k+1}=e_{k+1}+\bar{x}$ y $x_k=e_k+\bar{x}$,

$$e_{k+1}+\bar{x}=e_k+\bar{x}-e_k+\frac{\Gamma(2\alpha+1)-(\Gamma(\alpha+1))^2}{\alpha\Gamma^2(\alpha+1)}C_2^{CA}e_k^{\alpha+1}+O\left(e_k^{2\alpha+1}\right).$$

Por lo tanto,

$$e_{k+1}=\frac{\Gamma(2\alpha+1)-\Gamma^2(\alpha+1)}{\alpha\Gamma^2(\alpha+1)}C_2^{CA}e_k^{\alpha+1}+O\left(e_k^{2\alpha+1}\right).$$

Esto completa la prueba. □

Observación 3.2.1 *Hemos podido ver en la demostración del Teorema 3.2.1 que es necesario introducir los parámetros $\Gamma(\alpha+1)$ y $1/\alpha$ para asegurar el orden de convergencia.*

En el siguiente resultado se plantean las condiciones que aseguran la convergencia del procedimiento NeRL de manera similar, pero usando el desarrollo de Taylor (2.43). En este caso se omite la prueba, ya que es totalemte análoga a la demostración del Teorema 3.2.1.

Teorema 3.2.2 *Sea $f: I\subseteq\mathbb{R}\to\mathbb{R}$ una función con derivada fraccionaria de Riemann-Liouville de orden αk, para todo entero positivo k y para cualquier $\alpha\in(0,1]$, en el intervalo I conteniendo el cero $\bar{x}$ de $f(x)$. Supongamos que $D^{\alpha}_{0^+}f(x)$ es continua y no nula en $\bar{x}$. Si una aproximación inicial x_0 es lo suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del método de Newton-Raphson fraccionario con derivada de Riemann-Liouville (NeRL)*

$$x_{k+1}=x_k-\left(\Gamma(\alpha+1)\frac{f(x_k)}{D^{\alpha}_{0^+}f(x_k)}\right)^{1/\alpha},\quad k=0,1,2,\ldots \tag{3.5}$$

es al menos $\alpha+1$, siendo $0<\alpha\leq 1$, y la ecuación del error es

$$e_{k+1}=\frac{\Gamma(2\alpha+1)-\Gamma^2(\alpha+1)}{\alpha\Gamma^2(\alpha+1)}C_2^{RL}e_k^{\alpha+1}+O\left(e_k^{2\alpha+1}\right),$$

siendo $(\Gamma(\cdot))^n$ denotada por $\Gamma^n(\cdot)$, donde $e_k=x_k-\bar{x}$ y $C_j^{RL}=\dfrac{\Gamma(\alpha+1)}{\Gamma(j\alpha+1)}\dfrac{D^{j\alpha}_{0^+}f(\bar{x})}{D^{\alpha}_{0^+}f(\bar{x})}$, $j\geq 2$, tal que $x_k>0$, $\forall k$.

Corolario 3.2.1 *Como consecuencia de los Teoremas 3.2.1 y 3.2.2, los métodos de tipo Newton-Raphson fraccionarios NeCA y NeRL tienen orden de convergencia al menos lineal cuando $\alpha\approx 0$, y cuadrático cuando $\alpha=1$; éste último coincide con el caso del esquema clásico de Newton-Raphson.*

Al igual que en los procedimientos (3.2) y (3.3), cuando $\alpha=1$ se obtiene la ecuación del error del método clásico de Newton-Raphson en las ecuaciones del error de los esquemas NeCA y NeRL.

En la siguiente sección, presentamos el diseño y análisis de convergencia de una nueva variante fraccionaria de tipo Newton-Raphson con derivada conformable.

3.3. Método de tipo Newton-Raphson con derivada conformable

El uso de las derivadas de Caputo y Riemann-Liouville requieren la evalucación de funciones especiales, como la función Gamma y/o la de Mittag-Leffler (2.37), cuyo cálculo supone un alto coste computacional. En la teoría, el orden de convergencia de NeCA y NeRL tiende a ser cuadrático cuando $\alpha \approx 1$, pero en la práctica, el Orden de Convergencia Computacional Aproximado (ACOC, ver [17]) es lineal si $\alpha \neq 1$. También, en estos métodos las derivadas deben estar centradas en cero para poder obterner un orden de convergencia deseado; esto tampoco es una limitación con el uso de derivadas conformables. A continuación se presenta el diseño de un nuevo esquema fraccionario de tipo Newton-Raphson de orden cuadrático con derivada conformable que publicamos en la revista *Applied Mathematics Letters* (ver [11]).

3.3.1. Deducción del método

Para obtener un procedimiento iterativo con derivada conformable (2.46), consideremos el desarrollo de Taylor (2.49) de la función no lineal cuya raíz estaremos buscando. Sea $f : \mathbb{R} \to \mathbb{R}$ una función continua con derivadas fraccionarias conformables aplicadas una y dos veces, $0 < \alpha \leq 1$, y centradas en a, $a \in \mathbb{R}$, $a < x$. Entonces, aplicando el Teorema 2.3.4, $f(x)$ puede ser estimada con un polinomio de Taylor de primer grado alrededor de la raíz $\bar{x}$,

$$f(x) \approx f(\bar{x}) + \frac{(T_\alpha^a f)(\bar{x})\delta_1}{\alpha}. \tag{3.6}$$

Sabiendo que $f(\bar{x}) = 0$, y $\delta_1 = H^\alpha - L^\alpha$, siendo $H = x - a$ y $L = b - a$ $(b = \bar{x})$, (3.6) se puede escribir como

$$f(x) \approx \frac{(T_\alpha^a f)(\bar{x})}{\alpha} \left[(x-a)^\alpha - (\bar{x}-a)^\alpha\right].$$

Ahora, $(\bar{x} - a)^\alpha$ se puede despejar como

$$(\bar{x}-a)^\alpha \approx (x-a)^\alpha - \alpha \frac{f(x)}{(T_\alpha^a f)(\bar{x})}.$$

Así, de $(\bar{x} - a)^\alpha$, $\bar{x}$ queda despejada como

$$\bar{x} \approx a + \left((x-a)^\alpha - \alpha \frac{f(x)}{(T_\alpha^a f)(\bar{x})}\right)^{1/\alpha}.$$

Considerando los iterados x_k y x_{k+1} como aproximaciones de la raíz $\bar{x}$, se obtiene el método fraccionario conformable de tipo Newton-Raphson:

$$x_{k+1} = a + \left((x_k - a)^\alpha - \alpha \frac{f(x_k)}{(T_\alpha^a f)(x_k)}\right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots$$

Denotemos este esquema por NeCO.

Observación 3.3.1 *Este es el primer procedimiento fraccionario óptimo de acuerdo a la conjetura de Kung y Traub (ver [26]).*

3.3.2. Análisis de convergencia

En el siguiente resultado se establecen las condiciones que aseguran la convergencia del método NeCO con desarrollos de Taylor.

Teorema 3.3.1 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo un cero $\bar{x}$ de $f(x)$. Sea $(T_\alpha^a f)(x)$ la derivada fraccionaria conformable de $f(x)$ centrada en a, de orden α, para cualquier $\alpha \in (0, 1]$. Supongamos que $(T_\alpha^a f)(x)$ es continua y no nula en $\bar{x}$. Si una aproximación inicial x_0 es lo suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del método de tipo Newton-Raphson con derivada conformable (NeCO)*

$$x_{k+1} = a + \left((x_k - a)^\alpha - \alpha \frac{f(x_k)}{(T_\alpha^a f)(x_k)} \right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots$$

es al menos 2, siendo $0 < \alpha \leq 1$, y la ecuación del error es

$$e_{k+1} = \alpha (\bar{x} - a)^{\alpha - 1} C_2^{CO} e_k^2 + O\left(e_k^3\right),$$

siendo $C_j^{CO} = \dfrac{1}{j! \alpha^{j-1}} \dfrac{(T_\alpha^a f)^{(j)}(\bar{x})}{(T_\alpha^a f)(\bar{x})}$ para $j \geq 2$, tal que $a < x_k$, $\forall k$.

Demostración. Usando la serie de Taylor (2.49) de $f(x)$ en x_k en el entorno de $\bar{x}$, y considerando $x_k = e_k + \bar{x}$, el desarrollo de $f(x_k)$ se puede expresar como

$$\begin{aligned} f(x_k) &= (T_\alpha^a f)(\bar{x}) \left[\delta_1 + C_2^{CO} \delta_1^2 + C_3^{CO} \delta_1^3 \right] + O\left(e_k^4\right) \\ &= (T_\alpha^a f)(\bar{x}) \Big[\left((e_k + \bar{x} - a)^\alpha - (\bar{x} - a)^\alpha \right) + C_2^{CO} \left((e_k + \bar{x} - a)^\alpha - (\bar{x} - a)^\alpha \right)^2 \\ &+ C_3^{CO} \left((e_k + \bar{x} - a)^\alpha - (\bar{x} - a)^\alpha \right)^3 \Big] + O\left(e_k^4\right), \end{aligned}$$

siendo $C_j^{CO} = \dfrac{1}{j! \alpha^{j-1}} \dfrac{(T_\alpha^a f)^{(j)}(\bar{x})}{(T_\alpha^a f)(\bar{x})}$ para $j \geq 2$.

Usando el Teorema del binomio (2.52) y el coeficiente binomial (2.53),

$$\begin{aligned} f(x_k) &= (T_\alpha^a f)(\bar{x}) \left[\left(\alpha (\bar{x} - a)^{\alpha - 1} \right) e_k + \left(\frac{\alpha}{2} (\alpha - 1)(\bar{x} - a)^{\alpha - 2} + \alpha^2 (\bar{x} - a)^{2\alpha - 2} C_2^{CO} \right) e_k^2 \right. \\ &+ \left(\frac{\alpha}{6} (\alpha - 1)(\alpha - 2)(\bar{x} - a)^{\alpha - 3} + \alpha^2 (\alpha - 1)(\bar{x} - a)^{2\alpha - 3} C_2^{CO} \right. \\ &+ \left. \left. \alpha^3 (\bar{x} - a)^{3\alpha - 3} C_3^{CO} \right) e_k^3 \right] + O\left(e_k^4\right). \end{aligned}$$

Sabiendo que $(T_\alpha^a f)(x) = (x - a)^{1 - \alpha} f'(x)$, y usando de nuevo el Teorema del binomio y el coeficiente binomial generalizados, el desarrollo de $(T_\alpha^a f)(x_k)$ es

$$\begin{aligned} (T_\alpha^a f)(x_k) &= (T_\alpha^a f)(\bar{x}) \left[\left(\alpha + \left(2\alpha^2 (\bar{x} - a)^{\alpha - 1} C_2^{CO} \right) e_k \right. \right. \\ &+ \left(2\alpha(\alpha - 1)(\alpha - 2)(\bar{x} - a)^{-2} + \alpha^2 (\alpha - 1)(\bar{x} - a)^{\alpha - 2} C_2^{CO} + 3\alpha^3 (\bar{x} - a)^{2\alpha - 2} C_3^{CO} \right. \\ &- \left. \left. \alpha (1 - \alpha)^2 (\bar{x} - a)^{-2} + \frac{\alpha^2}{2} (\alpha - 1)(\bar{x} - a)^{-2} \right) e_k^2 \right] + O\left(e_k^3\right). \end{aligned}$$

El cociente $\dfrac{f(x_k)}{(T_\alpha^a f)(x_k)}$ puede expandirse como

$$\frac{f(x_k)}{(T_\alpha^a f)(x_k)} = (\bar{x}-a)^{\alpha-1}e_k + \left(\frac{1}{2}(\alpha-1)(\bar{x}-a)^{\alpha-2} - \alpha(\bar{x}-a)^{2\alpha-2}C_2^{CO}\right)e_k^2 + O\left(e_k^3\right),$$

y multiplicando por α, resulta

$$\alpha\frac{f(x_k)}{(T_\alpha^a f)(x_k)} = \alpha(\bar{x}-a)^{\alpha-1}e_k + \left(\frac{1}{2}\alpha(\alpha-1)(\bar{x}-a)^{\alpha-2} - \alpha^2(\bar{x}-a)^{2\alpha-2}C_2^{CO}\right)e_k^2 + O\left(e_k^3\right).$$

Restando esta expresión a $(x_k-a)^\alpha$, se tiene

$$(x_k-a)^\alpha - \alpha\frac{f(x_k)}{(T_\alpha^a f)(x_k)} = (\bar{x}-a)^\alpha + \alpha^2(\bar{x}-a)^{2\alpha-2}C_2^{CO}e_k^2 + O\left(e_k^3\right).$$

De nuevo, usando el Teorema del binomio y el coeficiente binomial generalizados, se obtiene

$$\left((x_k-a)^\alpha - \alpha\frac{f(x_k)}{(T_\alpha^a f)(x_k)}\right)^{1/\alpha} = \bar{x}-a+\alpha(\bar{x}-a)^{\alpha-1}C_2^{CO}e_k^2 + O\left(e_k^3\right).$$

Por lo tanto,

$$a+\left((x_k-a)^\alpha - \alpha\frac{f(x_k)}{(T_\alpha^a f)(x_k)}\right)^{1/\alpha} = \bar{x}+\alpha(\bar{x}-a)^{\alpha-1}C_2^{CO}e_k^2 + O\left(e_k^3\right).$$

Sea $x_{k+1} = e_{k+1} + \bar{x}$,

$$e_{k+1}+\bar{x} = \bar{x}+\alpha(\bar{x}-a)^{\alpha-1}C_2^{CO}e_k^2 + O\left(e_k^3\right).$$

Finalmente, la ecuación del error es

$$e_{k+1} = \alpha(\bar{x}-a)^{\alpha-1}C_2^{CO}e_k^2 + O\left(e_k^3\right). \tag{3.7}$$

Esto completa la prueba del orden cuadrático del método con independencia del orden de la derivada conformable. □

Observación 3.3.2 *Se puede mostrar que con las reglas del producto y la cadena establecidas en [1], la constante de error asintótica de la ecuación del error (3.7) se puede expresar como*

$$\begin{aligned}
\alpha(\bar{x}-a)^{\alpha-1}C_2^{CO} &= \alpha(\bar{x}-a)^{\alpha-1}\frac{1}{2\alpha}\frac{(T_\alpha^a f)^{(2)}(\bar{x})}{(T_\alpha^a f)(\bar{x})} \\
&= \frac{\alpha(\bar{x}-a)^{\alpha-1}}{2\alpha}\left[\frac{(\bar{x}-a)^{2-2\alpha}f''(\bar{x}) + (1-\alpha)(\bar{x}-a)^{1-2\alpha}f'(\bar{x})}{(\bar{x}-a)^{1-\alpha}f'(\bar{x})}\right] \\
&= \frac{1}{2}\left[\frac{f''(\bar{x})}{f'(\bar{x})} + \frac{1-\alpha}{\bar{x}-a}\right] \\
&= C_2 + \frac{1}{2}\frac{1-\alpha}{\bar{x}-a},
\end{aligned} \tag{3.8}$$

donde $C_j = \dfrac{1}{j!}\dfrac{f^{(j)}(\bar{x})}{f'(\bar{x})}$ *para* $j \geq 2$*, siendo esta la constante de error asintótica de los esquemas con derivada clásica (los procedimientos clásicos). En este caso,* $j = 2$.

Por tanto, como consecuencia del Teorema 3.3.1 y la observación 3.3.2, en el siguiente resultado se plantean, de forma alternativa, las condiciones que aseguran la convergencia del método NeCO, pero ahora con desarrollos de Taylor usando la serie con derivadas enteras clásica (2.10).

Corolario 3.3.1 *Sea* $f : I \subseteq \mathbb{R} \to \mathbb{R}$ *una función suficientemente diferenciable en el intervalo* I *conteniendo un cero* $\bar{x}$ *de* $f(x)$*. Sea* $(T_\alpha^a f)(x)$ *la derivada fraccionaria conformable de* $f(x)$ *centrada en* a*, de orden* α*, para cualquier* $\alpha \in (0, 1]$*. Supongamos que* $(T_\alpha^a f)(x)$ *es continua y no nula en* $\bar{x}$*. Si una aproximación inicial* x_0 *es lo suficientemente cercana a* $\bar{x}$*, entonces el orden de convergencia local del método de tipo Newton-Raphson con derivada conformable (NeCO)*

$$x_{k+1} = a + \left((x_k - a)^\alpha - \alpha \frac{f(x_k)}{(T_\alpha^a f)(x_k)} \right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots \tag{3.9}$$

es al menos 2*, siendo* $0 < \alpha \leq 1$*, y la ecuación del error es*

$$e_{k+1} = \left(C_2 + \frac{1}{2}\frac{1-\alpha}{\bar{x}-a} \right) e_k^2 + O\left(e_k^3\right),$$

donde $C_j = \dfrac{f^{(j)}(\bar{x})}{j! f'(\bar{x})}$*, para* $j \geq 2$*, tal que* $a < x_k$*,* $\forall k$.

Demostración. Usando la serie de Taylor (2.10) de $f(x)$ en x_k en el entorno de $\bar{x}$, y considerando $x_k = e_k + \bar{x}$, el desarrollo de $f(x_k)$ se puede expresar como

$$f(x_k) = f'(\bar{x}) \left[e_k + C_2 e_k^2 + C_3 e_k^3 \right] + O\left(e_k^4\right),$$

siendo $C_j = \dfrac{f^{(j)}(\bar{x})}{j! f'(\bar{x})}$, para $j \geq 2$.

Sabiendo que $(T_\alpha^a f)(x) = (x-a)^{1-\alpha} f'(x)$, y usando el Teorema del binomio y el coeficiente binomial generalizados, el desarrollo de $(T_\alpha^a f)(x_k)$ es

$$\begin{aligned} (T_\alpha^a f)(x_k) &= f'(\bar{x}) \left[(\bar{x}-a)^{1-\alpha} + \left((1-\alpha)(\bar{x}-a)^{-\alpha} + 2(\bar{x}-a)^{1-\alpha} C_2 \right) e_k \right. \\ &+ \left. \left(\frac{1}{2}\alpha(\alpha-1)(\bar{x}-a)^{-1-\alpha} + 2(1-\alpha)(\bar{x}-a)^{-\alpha} C_2 + 3(\bar{x}-a)^{1-\alpha} C_3 \right) e_k^2 \right] + O\left(e_k^3\right). \end{aligned}$$

El cociente $\dfrac{f(x_k)}{(T_\alpha^a f)(x_k)}$ se calcula como

$$\frac{f(x_k)}{(T_\alpha^a f)(x_k)} = (\bar{x}-a)^{\alpha-1} e_k + \left((\alpha-1)(\bar{x}-a)^{\alpha-2} - (\bar{x}-a)^{\alpha-1} C_2 \right) e_k^2 + O\left(e_k^3\right),$$

y multiplicando por α, se obtiene

$$\alpha \frac{f(x_k)}{(T_\alpha^a f)(x_k)} = \alpha(\bar{x}-a)^{\alpha-1} e_k + \alpha \left((\alpha-1)(\bar{x}-a)^{\alpha-2} - (\bar{x}-a)^{\alpha-1} C_2 \right) e_k^2 + O\left(e_k^3\right).$$

Sustrayendo esta expresión a $(x_k - a)^\alpha$, resulta

$$(x_k-a)^\alpha - \alpha\frac{f(x_k)}{(T_\alpha^a f)(x_k)} = (\bar{x}-a)^\alpha + \alpha\left((\bar{x}-a)^{\alpha-1}C_2 + \frac{1}{2}(1-\alpha)(\bar{x}-a)^{\alpha-2}\right)e_k^2 + O\left(e_k^3\right).$$

Usando de nuevo el Teorema del binomio y el coeficiente binomial generalizados,

$$\left((x_k-a)^\alpha - \alpha\frac{f(x_k)}{(T_\alpha^a f)(x_k)}\right)^{1/\alpha} = \bar{x}-a+\left(C_2+\frac{1}{2}\frac{1-\alpha}{\bar{x}-a}\right)e_k^2 + O\left(e_k^3\right).$$

Por tanto,

$$a+\left((x_k-a)^\alpha - \alpha\frac{f(x_k)}{(T_\alpha^a f)(x_k)}\right)^{1/\alpha} = \bar{x}+\left(C_2+\frac{1}{2}\frac{1-\alpha}{\bar{x}-a}\right)e_k^2 + O\left(e_k^3\right).$$

Sea $x_{k+1} = e_{k+1} + \bar{x}$,

$$e_{k+1}+\bar{x} = \bar{x}+\left(C_2+\frac{1}{2}\frac{1-\alpha}{\bar{x}-a}\right)e_k^2 + O\left(e_k^3\right).$$

Finalmente,

$$e_{k+1} = \left(C_2+\frac{1}{2}\frac{1-\alpha}{\bar{x}-a}\right)e_k^2 + O\left(e_k^3\right). \tag{3.10}$$

Esto completa la prueba. □

Observación 3.3.3 *De acuerdo con la Observación 3.3.2 y las ecuaciones del error (3.7) y (3.10), se concluye que las ecuaciones del error obtenidas con las series de Taylor (2.49) y (2.10) son equivalentes.*

En ambas ecuaciones del error, (3.7) y (3.10), cuando $\alpha = 1$ se obtiene la ecuación del error del esquema clásico de Newton-Raphson, y en (3.9) se obtiene el procedimiento clásico de Newton-Raphson.

Observación 3.3.4 *Dado que el método NeCO y el de Newton-Raphson clásico tienen el mismo oden de convergencia, se comprueba el Teorema 2.1.3.*

En adelante, usaremos sólo los desarrollos de Taylor clásicos para realizar el análisis de convergencia de todos los procedimientos con derivadas conformables (o su aproximación) que se presentan en esta memoria, con excepción de la versión vectorial del método NeCO, el cual se verá en el siguiente capítulo.

En la siguiente sección, presentamos el diseño y análisis de convergencia de nuevas variantes de tipo Newton-Raphson con derivadas de mayor orden.

3.4. Métodos de tipo Newton-Raphson conformables de mayor orden

Para aumentar el orden de convergencia de un esquema iterativo en una unidad, la constante de error asintótica de la ecuación del error de dicho procedimiento debe anularse. Como ejemplo, consideremos el método NeCO, en cuya ecuación del error la constante de error asintótica se anula si se cumple que

$$C_2 = \frac{1}{2}\frac{\alpha - 1}{\bar{x} - a}.$$

A continuación presentamos el diseño de tres esquemas fraccionarios basados en NeCO, dos de orden tres y uno de orden cuatro.

3.4.1. Deducción de los métodos

Antes de diseñar cada procedimiento necesitamos la ecuación del error (3.10) hasta orden cuatro. De forma análoga a como se procedió en la demostración del Corolario 3.3.1, pero considerando más términos en los desarrollos de Taylor, la ecuación del error de NeCO hasta orden cuatro es

$$\begin{aligned} e_{k+1} &= \left(C_2 + \frac{1-\alpha}{2(\bar{x}-a)}\right) e_k^2 + \left(2C_3 - 2C_2^2 + \frac{(\alpha-1)C_2}{\bar{x}-a} + \frac{(1-\alpha)(\alpha-2)}{3(\bar{x}-a)^2}\right) e_k^3 \\ &+ \left(3C_3 - 7C_2C_3 + 4C_2^3 + \frac{(1-\alpha)(5C_2^2 - C_3)}{2(\bar{x}-a)} + \frac{(2\alpha^2 - 5\alpha + 3)C_2}{2(\bar{x}-a)^2} + \frac{(1-\alpha)(2\alpha^2 - 7\alpha + 7)}{8(\bar{x}-a)^3}\right) e_k^4 \\ &+ O\left(e_k^5\right). \end{aligned} \tag{3.11}$$

Para el primer método, queremos encontrar el valor de α que anula el término que multiplica a e_k^2 en (3.11), de modo que

$$\begin{aligned} \alpha &= 1 + 2(\bar{x} - a)C_2 \\ &= 1 + 2(\bar{x} - a)\frac{1}{2}\frac{f''(\bar{x})}{f'(\bar{x})} \\ &= 1 + (\bar{x} - a)\frac{f''(\bar{x})}{f'(\bar{x})}. \end{aligned}$$

Considerando a α_k y al iterado x_k como aproximaciones de α y de la raíz $\bar{x}$, respectivamente,

$$\alpha_k = 1 + (x_k - a)\frac{f''(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots \tag{3.12}$$

Al sustituir el valor de α_k (3.12) en el esquema NeCO, obtenemos un nuevo procedimiento, el cual denotaremos por NeL3:

$$x_{k+1} = a + \left((x_k - a)^{\alpha_k} - \alpha_k \frac{f(x_k)}{(T_{\alpha_k}^a f)(x_k)}\right)^{1/\alpha_k}, \quad k = 0, 1, 2, \ldots,$$

donde α_k está definido en (3.12).

Para el segundo método, queremos encontrar el valor de a que anula el término que multiplica a e_k^2 en (3.11), de modo que

$$\begin{aligned} a &= \bar{x} + \frac{1-\alpha}{2C_2} \\ &= \bar{x} + \frac{1}{2}\frac{2(1-\alpha)f'(\bar{x})}{f''(\bar{x})} \\ &= \bar{x} + (1-\alpha)\frac{f'(\bar{x})}{f''(\bar{x})}. \end{aligned}$$

Considerando a a_k y al iterado x_k como aproximaciones de a y de la raíz $\bar{x}$, respectivamente,

$$a_k = x_k + (1-\alpha)\frac{f'(x_k)}{f''(x_k)}, \quad k = 0, 1, 2, \ldots \tag{3.13}$$

Al sustituir el valor de a_k (3.13) en el esquema NeCO, obtenemos un nuevo procedimiento, el cual denotaremos por NeA3:

$$x_{k+1} = a_k + \left((x_k - a_k)^\alpha - \alpha\frac{f(x_k)}{(T_\alpha^{a_k} f)(x_k)}\right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots$$

donde a_k está definido en (3.13).

Para el tercer método, queremos encontrar los valores de α y a que simultáneamente anulan los términos que multiplican a e_k^2 y e_k^3 en (3.11). Resolviendo dicho sistema no lineal para α y a, tenemos que

$$\begin{aligned}
\alpha &= 1 + \frac{2C_2^2}{2C_2^2 - 3C_3} \\
&= 1 + \frac{2\left(\dfrac{f''(\bar{x})}{2f'(\bar{x})}\right)^2}{2\left(\dfrac{f''(\bar{x})}{2f'(\bar{x})}\right)^2 - 3\left(\dfrac{f'''(\bar{x})}{6f'(\bar{x})}\right)} \\
&= 1 + \frac{\left(\dfrac{f''(\bar{x})}{f'(\bar{x})}\right)^2}{\left(\dfrac{f''(\bar{x})}{f'(\bar{x})}\right)^2 - \dfrac{f'''(\bar{x})}{f'(\bar{x})}} \\
&= 1 + \frac{f''(\bar{x})^2}{f''(\bar{x})^2 - f'(\bar{x})f'''(\bar{x})}.
\end{aligned}$$

Considerando a α_k y al iterado x_k como aproximaciones de α y de la raíz $\bar{x}$, respectivamente,

$$\alpha_k = 1 + \frac{f''(x_k)^2}{f''(x_k)^2 - f'(x_k)f'''(x_k)}, \quad k = 0, 1, 2, \ldots \tag{3.14}$$

Y el valor de a es

$$\begin{aligned}
a &= \bar{x} + \frac{C_2}{3C_3 - 2C_2^2} \\
&= \bar{x} + \frac{\dfrac{f''(\bar{x})}{2f'(\bar{x})}}{3\left(\dfrac{f'''(\bar{x})}{6f'(\bar{x})}\right) - 2\left(\dfrac{f''(\bar{x})}{2f'(\bar{x})}\right)^2} \\
&= \bar{x} + \frac{\dfrac{f''(\bar{x})}{f'(\bar{x})}}{\dfrac{f'''(\bar{x})}{f'(\bar{x})} - \left(\dfrac{f''(\bar{x})}{f'(\bar{x})}\right)^2} \\
&= \bar{x} + \frac{f'(\bar{x})f''(\bar{x})}{f'(\bar{x})f'''(\bar{x}) - f''(\bar{x})^2}.
\end{aligned}$$

Considerando a a_k y al iterado x_k como aproximaciones de a y de la raíz $\bar{x}$, respectivamente,

$$a_k = x_k + \frac{f'(x_k)f''(x_k)}{f'(x_k)f'''(x_k) - f''(x_k)^2}, \quad k = 0, 1, 2, \ldots \tag{3.15}$$

Al sustituir los valores de α_k (3.14) y a_k (3.15) en el esquema NeCO, obtenemos un nuevo procedimiento, el cual denotaremos por NeLA4:

$$x_{k+1} = a_k + \left((x_k - a_k)^{\alpha_k} - \alpha_k \frac{f(x_k)}{(T^{a_k}_{\alpha_k} f)(x_k)} \right)^{1/\alpha_k}, \quad k = 0, 1, 2, \ldots$$

donde α_k y a_k están definidos en (3.14) y (3.15), respectivamente.

De esta forma, hemos diseñado esquemas punto a punto que mejoran el orden de convergencia del método de partida NeCO.

3.4.2. Análisis de convergencia

En el siguiente resultado se establecen las condiciones que aseguran la convergencia del método NeL3.

Teorema 3.4.1 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo un cero $\bar{x}$ de $f(x)$. Sea $(T^a_{\alpha_k} f)(x)$ la derivada fraccionaria conformable de $f(x)$ centrada en a, de orden α_k. Supongamos que $(T^a_{\alpha_k} f)(x)$ es continua y no nula en $\bar{x}$. Si una aproximación inicial x_0 es lo suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del método de tipo Newton-Raphson con derivada conformable (NeL3)*

$$x_{k+1} = a + \left((x_k - a)^{\alpha_k} - \alpha_k \frac{f(x_k)}{(T^a_{\alpha_k} f)(x_k)} \right)^{1/\alpha_k}, \quad k = 0, 1, 2, \ldots,$$

siendo

$$\alpha_k = 1 + (x_k - a)\frac{f''(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots,$$

es al menos 3, y la ecuación del error es

$$e_{k+1} = \frac{1}{3}\left(2C_2^2 - 3C_3 - \frac{C_2}{\bar{x} - a} \right) e_k^3 + O\left(e_k^4\right), \tag{3.16}$$

donde $C_j = \dfrac{f^{(j)}(\bar{x})}{j! f'(\bar{x})}$, para $j \geq 2$, tal que $a < x_k$, $\forall k$.

Demostración. Basta con sustituir α en la ecuación del error (3.11) por el desarrollo de Taylor de α_k (3.12). Los desarrollos de Taylor de $f'(x_k)$ y $f''(x_k)$ en el entorno de la raíz $\bar{x}$ se pueden expresar como

$$f'(x_k) = f'(\bar{x})\left[1 + 2C_2 e_k + 3C_3 e_k^2 + 4C_4 e_k^3\right] + O\left(e_k^4\right),$$

y

$$f''(x_k) = f'(\bar{x})\left[2C_2 + 6C_3 e_k + 12C_4 e_k^2\right] + O\left(e_k^3\right),$$

respectivamente, siendo $C_j = \dfrac{f^{(j)}(\bar{x})}{j!f'(\bar{x})}$, para $j \geq 2$.

El desarrollo del cociente $\dfrac{f''(x_k)}{f'(x_k)}$ se calcula como

$$\frac{f''(x_k)}{f'(x_k)} = 2C_2 + 2\left(3C_3 - 2C_2^2\right)e_k + 2\left(4C_2^3 - 9C_2C_3 + 6C_4\right)e_k^2 + O\left(e_k^3\right).$$

El producto $(x_k - a)\dfrac{f''(x_k)}{f'(x_k)} = (\bar{x} - a + e_k)\dfrac{f''(x_k)}{f'(x_k)}$ resulta

$$\begin{aligned}(x_k - a)\frac{f''(x_k)}{f'(x_k)} &= 2(\bar{x}-a)C_2 + 2\left((\bar{x}-a)\left(3C_3 - 2C_2^2\right) + C_2\right)e_k \\ &+ 2\left((\bar{x}-a)\left(4C_2^3 - 9C_2C_3 + 6C_4\right) + 3C_3 - 2C_2^2\right)e_k^2 + O\left(e_k^3\right).\end{aligned}$$

Por tanto,

$$\begin{aligned}\alpha_k = 1 + (x_k - a)\frac{f''(x_k)}{f'(x_k)} &= 1 + 2(\bar{x}-a)C_2 + 2\left((\bar{x}-a)\left(3C_3 - 2C_2^2\right) + C_2\right)e_k \\ &+ 2\left((\bar{x}-a)\left(4C_2^3 - 9C_2C_3 + 6C_4\right) + 3C_3 - 2C_2^2\right)e_k^2 + O\left(e_k^3\right),\end{aligned}$$

y sustituyendo α en (3.11) por el desarrollo de α_k , tenemos la ecuación del error

$$e_{k+1} = \frac{1}{3}\left(2C_2^2 - 3C_3 - \frac{C_2}{\bar{x}-a}\right)e_k^3 + O\left(e_k^4\right).$$

Esto completa la prueba. □

Observación 3.4.1 *El esquema NeL3 no es óptimo, ya que obtenemos orden de convergencia tres con $f''(x_k)$ como una nueva evaluación funcional.*

En el siguiente resultado se plantean las condiciones que aseguran la convergencia del procedimiento NeA3.

Teorema 3.4.2 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo un cero $\bar{x}$ de $f(x)$. Sea $(T_\alpha^{a_k} f)(x)$ la derivada fraccionaria conformable de $f(x)$ centrada en a_k, de orden α, para cualquier $\alpha \in (0,1)$. Supongamos que $(T_\alpha^{a_k} f)(x)$ es continua y no nula en $\bar{x}$. Si una aproximación inicial x_0 es lo suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del método de tipo Newton-Raphson con derivada conformable (NeA3)*

$$x_{k+1} = a_k + \left((x_k - a_k)^\alpha - \alpha\frac{f(x_k)}{(T_\alpha^{a_k} f)(x_k)}\right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots,$$

siendo

$$a_k = x_k + (1-\alpha)\frac{f'(x_k)}{f''(x_k)}, \quad k = 0, 1, 2, \ldots,$$

es al menos 3, *siendo* $0 < \alpha < 1$, *y la ecuación del error es*

$$e_{k+1} = \left(\frac{2}{3}\frac{(2-\alpha)C_2^2}{1-\alpha} - C_3\right) e_k^3 + O\left(e_k^4\right), \tag{3.17}$$

donde $C_j = \dfrac{f^{(j)}(\bar{x})}{j!f'(\bar{x})}$, *para* $j \geq 2$.

Demostración. Basta con sustituir a en la ecuación del error (3.11) por el desarrollo de Taylor de a_k (3.13). Los desarrollos de Taylor de $f'(x_k)$ y $f''(x_k)$ en el entorno de la raíz $\bar{x}$ se pueden expresar como

$$f'(x_k) = f'(\bar{x})\left[1 + 2C_2e_k + 3C_3e_k^2 + 4C_4e_k^3\right] + O\left(e_k^4\right),$$

y

$$f''(x_k) = f'(\bar{x})\left[2C_2 + 6C_3e_k + 12C_4e_k^2\right] + O\left(e_k^3\right),$$

respectivamente, siendo $C_j = \dfrac{f^{(j)}(\bar{x})}{j!f'(\bar{x})}$, para $j \geq 2$.

El cociente $\dfrac{f'(x_k)}{f''(x_k)}$ se calcula como

$$\frac{f'(x_k)}{f''(x_k)} = \frac{1}{2C_2} + \left(1 - \frac{3}{2}\frac{C_3}{C_2^2}\right)e_k + \frac{3}{2}\left(\frac{3C_3^2 - C_2^2C_3 - 2C_2C_4}{C_2^3}\right)e_k^2 + O\left(e_k^3\right).$$

El producto $(1-\alpha)\dfrac{f'(x_k)}{f''(x_k)}$ resulta

$$(1-\alpha)\frac{f'(x_k)}{f''(x_k)} = \frac{1-\alpha}{2C_2} + (1-\alpha)\left(1 - \frac{3}{2}\frac{C_3}{C_2^2}\right)e_k + \frac{3}{2}(1-\alpha)\left(\frac{3C_3^2 - C_2^2C_3 - 2C_2C_4}{C_2^3}\right)e_k^2 + O\left(e_k^3\right).$$

Por tanto,

$$\begin{aligned} a_k &= x_k + (1-\alpha)\frac{f'(x_k)}{f''(x_k)} = \bar{x} + e_k + (1-\alpha)\frac{f'(x_k)}{f''(x_k)} = \bar{x} + \frac{1-\alpha}{2C_2} \\ &+ \left(2 - \alpha - \frac{3}{2}\frac{(1-\alpha)C_3}{C_2^2}\right)e_k + \frac{3}{2}(1-\alpha)\left(\frac{3C_3^2 - C_2^2C_3 - 2C_2C_4}{C_2^3}\right)e_k^2 + O\left(e_k^3\right). \end{aligned}$$

Sustituyendo a en (3.11) por el desarrollo de a_k , tenemos

$$e_{k+1} = \left(\frac{2}{3}\frac{(2-\alpha)C_2^2}{1-\alpha} - C_3\right) e_k^3 + O\left(e_k^4\right),$$

donde $\alpha \neq 1$.

Esto completa la prueba. □

Observación 3.4.2 *El método NeA3 tampoco es óptimo, ya que obtenemos orden de convergencia tres con* $f''(x_k)$ *como una nueva evaluación funcional.*

Observación 3.4.3 *Nótese que cuando $\alpha = 1$ en el esquema NeA3 obtenemos el método clásico de Newton-Raphson, por lo que éste es de orden cuadrático en este caso.*

En el siguiente resultado se establecen las condiciones que aseguran la convergencia del procedimiento NeLA4.

Teorema 3.4.3 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo un cero $\bar{x}$ de $f(x)$. Sea $(T^{a_k}_{\alpha_k} f)(x)$ la derivada fraccionaria conformable de $f(x)$ centrada en a_k, de orden α_k. Supongamos que $(T^{a_k}_{\alpha_k} f)(x)$ es continua y no nula en $\bar{x}$. Si una aproximación inicial x_0 es lo suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del método de tipo Newton-Raphson con derivada conformable (NeLA4)*

$$x_{k+1} = a_k + \left((x_k - a_k)^{\alpha_k} - \alpha_k \frac{f(x_k)}{(T^{a_k}_{\alpha_k} f)(x_k)} \right)^{1/\alpha_k}, \quad k = 0, 1, 2, \ldots,$$

siendo

$$\alpha_k = 1 + \frac{f''(x_k)^2}{f''(x_k)^2 - f'(x_k) f'''(x_k)}, \quad k = 0, 1, 2, \ldots,$$

y

$$a_k = x_k + \frac{f'(x_k) f''(x_k)}{f'(x_k) f'''(x_k) - f''(x_k)^2}, \quad k = 0, 1, 2, \ldots,$$

es al menos 4, y la ecuación del error es

$$e_{k+1} = 2 \left(C_2 C_3 - 3 \frac{C_3^2}{C_2} + 2 C_4 \right) e_k^4 + O\left(e_k^5\right),$$

donde $C_j = \dfrac{f^{(j)}(\bar{x})}{j! f'(\bar{x})}$, para $j \geq 2$.

Demostración. Basta con sustituir α y a en la ecuación del error (3.11) por los desarrollos de Taylor de α_k (3.14) y a_k (3.15), respectivamente. Los desarrollos de Taylor de $f'(x_k)$, $f''(x_k)$ y $f'''(x_k)$ en el entorno de la raíz $\bar{x}$ se pueden expresar como

$$f'(x_k) = f'(\bar{x}) \left[1 + 2 C_2 e_k + 3 C_3 e_k^2 + 4 C_4 e_k^3 + 5 C_5 e_k^4 \right] + O\left(e_k^5\right),$$

$$f''(x_k) = f'(\bar{x}) \left[2 C_2 + 6 C_3 e_k + 12 C_4 e_k^2 + 20 C_5 e_k^3 \right] + O\left(e_k^4\right),$$

y

$$f'''(x_k) = f'(\bar{x}) \left[6 C_3 + 24 C_4 e_k + 60 C_5 e_k^2 \right] + O\left(e_k^3\right),$$

respectivamente, siendo $C_j = \dfrac{f^{(j)}(\bar{x})}{j! f'(\bar{x})}$, para $j \geq 2$.

El desarrollo de los siguientes cocientes se calcula como:

$$\begin{aligned}\frac{f''(x_k)}{f'(x_k)} &= 2C_2 + 2\left(3C_3 - 2C_2^2\right)e_k + 2\left(4C_2^3 - 9C_2C_3 + 6C_4\right)e_k^2 \\ &+ 2\left(24C_2^2C_3 - 8C_2^4 - 9C_3^2 - 16C_2C_4 + 10C_5\right)e_k^3 + O\left(e_k^4\right),\end{aligned}$$

y

$$\frac{f'''(x_k)}{f'(x_k)} = 6C_3 + 12(2C_4 - C_2C_3)e_k + 6\left(4C_2^2C_3 - 3C_3^2 - 8C_2C_4 + 10C_5\right)e_k^2 + O\left(e_k^3\right).$$

Así,

$$\left(\frac{f''(x_k)}{f'(x_k)}\right)^2 = 4C_2^2 + 8C_2\left(3C_3 - 2C_2^2\right)e_k + 12\left(4C_2^4 - 10C_2^2C_3 + 3C_3^2 + 4C_2C_4\right)e_k^2 + O\left(e_k^3\right).$$

Por lo que para α_k resulta

$$\begin{aligned}\alpha_k &= 1 + \frac{\left(\frac{f''(x_k)}{f'(x_k)}\right)^2}{\left(\frac{f''(x_k)}{f'(x_k)}\right)^2 - \left(\frac{f'''(x_k)}{f'(x_k)}\right)} = 1 + \frac{2C_2^2}{2C_2^2 - C_3} + \left(\frac{12C_2\left(C_2^2C_3 - 3C_3^2 + 2C_2C_4\right)}{\left(2C_2^2 - 3C_3\right)^2}\right)e_k \\ &+ \left(\frac{6\left(27C_3^4 + 16C_2^5C_4 - 48C_2^3C_3C_4 - 36C_2C_3^2C_4 + C_2^4\left(20C_5 - 6C_3^2\right) + C_2^2\left(9C_3^3 + 48C_4^2 - 30C_3C_5\right)\right)}{\left(2C_2^2 - 3C_3\right)^3}\right)e_k^2 \\ &+ O\left(e_k^3\right).\end{aligned}$$

Y parar a_k tenemos

$$\begin{aligned}a_k &= x_k + \frac{\left(\frac{f''(x_k)}{f'(x_k)}\right)}{\left(\frac{f'''(x_k)}{f'(x_k)}\right) - \left(\frac{f''(x_k)}{f'(x_k}\right)^2} = \bar{x} + \frac{C_2}{3C_3 - 2C_2^2} + \left(\frac{6C_2^2C_3 - 4C_2^4 + 9C_3^2 - 12C_2C_4}{\left(2C_2^2 - 3C_3\right)^2}\right)e_k \\ &+ \left(\frac{6\left(2C_2^5C_3 + 12C_2^4C_4 - 36C_2^2C_3C_4 - 9C_3^2C_4 - 5C_2^3\left(3C_3^2 - 2C_5\right) + 3C_2\left(9C_3^3 + 8C_4^2 - 5C_3C_5\right)\right)}{\left(2C_2^2 - 3C_3\right)^3}\right)e_k^2 \\ &+ O\left(e_k^3\right).\end{aligned}$$

Sustituyendo α y a en (3.11) por los desarrollos de α_k y a_k, respectivamente, tenemos

$$e_{k+1} = 2\left(C_2C_3 - \frac{3C_3^2}{C_2} + 2C_4\right)e_k^4 + O\left(e_k^5\right).$$

Esto completa la prueba. □

Observación 3.4.4 *Este esquema tampoco es óptimo, ya que obtenemos orden de convergencia cuatro con $f''(x_k)$ y $f'''(x_k)$ como nuevas evaluaciones funcionales.*

En la siguiente sección, realizamos diversas pruebas numéricas con algunas funciones no lineales. También se analiza la dependencia a las estimaciones iniciales de dichos métodos con el uso de planos de convergencia y/o planos dinámicos.

3.5. Pruebas numéricas

Para obtener los resultados mostrados en esta sección, hemos utilizado Matlab R2020a con aritmética de precisión doble, $|f(x_{k+1})| < 10^{-8}$ o $|x_{k+1} - x_k| < 10^{-8}$ como criterios de parada, y un máximo de 500 iteraciones. Para el cálculo de la función Gamma utilizamos el programa propuesto en [27] (ver fichero MATLAB en Anexo A.1). Para la función de Mittag-Leffler utilizamos el programa proporcionado por Igor Podlubny en MathWorks (ver fichero MATLAB en Anexo A.2). En cuanto a la precisión de estas funciones, la función *Gamma.m* del Anexo A.1 se calcula con 15 dígitos de precisión a lo largo del eje real y 13 en otras partes de $\mathbb{C}$, y la función *mlf.m* del Anexo A.2 para calcular la función de Mittag-Leffler tiene 9 cifras significativas de precisión. La función Gamma y la de Mittag-Leffler sólo se calculan para los métodos NeCA y NeRL, los cuales usan derivadas de Caputo y de Riemann-Liouville, respectivamente; la derivada conformable no necesita que se evalúen estas funciones. También usamos el Orden de Convergencia Computacional Aproximado (ACOC, por sus siglas en inglés)

$$ACOC = \rho = \frac{\ln(|x_{k+1} - x_k|/|x_k - x_{k-1}|)}{\ln(|x_k - x_{k-1}|/|x_{k-1} - x_{k-2}|)}, \quad k = 2, 3, 4, \ldots,$$

definido en [17], para verificar que el orden de convergencia teórico también se conserva en la práctica.

Vamos a probar cuatro funciones no lineales para hacer una comparación entre los esquemas diseñados en este capítulo (ver ficheros MATLAB en Anexos A.6 a A.11). Los procedimientos cuyo $\alpha \in (0, 1]$ no se recalcula en cada iteración (NeCA, NeRL, NeCO y NeA3) se comparan con el método clásico de Newton-Raphson (cuando $\alpha = 1$). En el caso de los esquemas cuyo a no se recalcula en cada iteración (NeCO, NeL3) se escoge $a = -10$ para asegurar que $a < x_k$, $\forall k$; para los procedimientos NeCA y NeRL (con derivadas de Caputo y Riemann-Liouville, respectivamente) a debe ser cero, ya que sólo así se puede demostrar la convergencia en la teoría. Para cada función de prueba calcularemos, además de su respectiva derivada fraccionaria, sus derivadas clásicas de orden uno, dos y tres, ya que serán necesarias para los métodos NeL3, NeA3 y NeLA4.

En cada tabla mostramos los resultados obtenidos para cada función de prueba con una misma estimación inicial x_0, con el uso de los seis esquemas diseñados en este capítulo.

Nuestra primera función es $f_1(x) = -12.84x^6 - 25.6x^5 + 16.55x^4 - 2.21x^3 + 26.71x^2 - 4.29x - 15.21$, con raíces $\bar{x}_1 = 0.82366 + 0.24769i$, $\bar{x}_2 = 0.82366 - 0.24769i$, $\bar{x}_3 = -2.62297$, $\bar{x}_4 = -0.584$, $\bar{x}_5 = -0.21705 + 0.99911i$ y $\bar{x}_6 = -0.21705 - 0.99911i$. Las derivadas necesarias para esta función son:

$$\begin{aligned}
{}_{c}D^{\alpha}_{0^+} f_1(x) &= -12.84\frac{\Gamma(7)}{\Gamma(7-\alpha)}x^{6-\alpha} - 25.6\frac{\Gamma(6)}{\Gamma(6-\alpha)}x^{5-\alpha} + 16.55\frac{\Gamma(5)}{\Gamma(5-\alpha)}x^{4-\alpha} \\
&\quad -2.21\frac{\Gamma(4)}{\Gamma(4-\alpha)}x^{3-\alpha} + 26.71\frac{\Gamma(3)}{\Gamma(3-\alpha)}x^{2-\alpha} - 4.29\frac{\Gamma(2)}{\Gamma(2-\alpha)}x^{1-\alpha}, \\
D^{\alpha}_{0^+} f_1(x) &= {}_{c}D^{\alpha}_{0^+} f_1(x) - 15.21\frac{1}{\Gamma(1-\alpha)}x^{-\alpha}, \\
f_1'(x) &= -77.04x^5 - 128x^4 + 66.2x^3 - 6.63x^2 + 53.42x - 4.29, \\
f_1''(x) &= -385.2x^4 - 512x^3 + 198.6x^2 - 13.26x + 53.42, \\
f_1'''(x) &= -1540.8x^3 - 1536x^2 + 397.2x - 13.26, \\
(T^a_\alpha f_1)(x) &= (x-a)^{1-\alpha} f_1'(x).
\end{aligned}$$

En la Tabla 3.1 podemos ver que en los esquemas NeCA y NeRL el número de iteraciones requerido es mayor cuando α disminuye, y que ρ es lineal cuando $\alpha \neq 1$; no se muestran resultados cuando éstos no convergen en 500 iteraciones. Podemos observar que en el procedimiento NeCO el número de iteraciones puede ser igual o menor que en el método clásico de Newton-Raphson (cuando $\alpha = 1$), y que ρ se mantiene en el orden teórico. También se puede ver que en el esquema NeA3, a pesar de que, en general, el número de iteraciones requerido es mayor que en el procedimiento NeCO, ρ tiende a ser cúbico; aquí se observa que, cuando $\alpha = 1$,

	Método NeCA					Método NeRL				
α	$\bar{x}$	$\lvert f_1(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_1(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_4$	4.36×10^{-11}	6.85×10^{-7}	13	2.01	$\bar{x}_4$	4.36×10^{-11}	6.85×10^{-7}	13	2.01
0.9	$\bar{x}_4$	2.85×10^{-6}	9.79×10^{-9}	41	0.98	$\bar{x}_3$	8.02×10^{-5}	8.16×10^{-9}	37	0.98
0.8	$\bar{x}_3$	5.94×10^{-4}	9.65×10^{-9}	119	0.99	$\bar{x}_2$	1.80×10^{-5}	9.92×10^{-9}	122	0.99
0.7	$\bar{x}_1$	1.10×10^{-4}	9.95×10^{-9}	401	1.00	$\bar{x}_4$	9.27×10^{-5}	9.98×10^{-9}	354	1.00
0.6	-	-	-	>500	-	-	-	-	>500	-
0.5	-	-	-	>500	-	-	-	-	>500	-
0.4	-	-	-	>500	-	-	-	-	>500	-
0.3	-	-	-	>500	-	-	-	-	>500	-
0.2	-	-	-	>500	-	-	-	-	>500	-
0.1	-	-	-	>500	-	-	-	-	>500	-

	Método NeCO					Método NeA3				
α	$\bar{x}$	$\lvert f_1(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_1(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_4$	4.36×10^{-11}	6.85×10^{-7}	13	2.01	$\bar{x}_4$	4.36×10^{-11}	6.85×10^{-7}	13	2.01
0.9	$\bar{x}_4$	8.48×10^{-10}	3.02×10^{-6}	12	2.01	$\bar{x}_4$	3.55×10^{-15}	1.02×10^{-9}	7	2.92
0.8	$\bar{x}_4$	1.07×10^{-14}	2.01×10^{-10}	12	2.00	$\bar{x}_3$	3.91×10^{-13}	3.58×10^{-8}	15	2.89
0.7	$\bar{x}_4$	1.69×10^{-10}	1.36×10^{-6}	10	1.99	$\bar{x}_6$	5.33×10^{-15}	5.35×10^{-10}	38	3.01
0.6	$\bar{x}_4$	1.87×10^{-11}	4.52×10^{-7}	11	2.01	$\bar{x}_4$	1.71×10^{-13}	8.73×10^{-6}	22	3.37
0.5	$\bar{x}_4$	2.24×10^{-13}	4.17×10^{-8}	12	2.00	$\bar{x}_3$	6.18×10^{-13}	6.70×10^{-10}	54	2.96
0.4	$\bar{x}_4$	1.17×10^{-13}	2.52×10^{-10}	13	2.00	$\bar{x}_3$	3.91×10^{-13}	1.69×10^{-7}	31	2.91
0.3	$\bar{x}_4$	2.59×10^{-12}	1.66×10^{-7}	13	2.01	$\bar{x}_4$	1.03×10^{-9}	1.88×10^{-4}	19	3.57
0.2	$\bar{x}_4$	1.10×10^{-9}	3.49×10^{-6}	13	2.02	$\bar{x}_3$	4.02×10^{-9}	8.18×10^{-5}	35	2.81
0.1	$\bar{x}_4$	6.54×10^{-13}	5.03×10^{-8}	12	2.00	$\bar{x}_4$	9.07×10^{-10}	1.94×10^{-4}	168	3.53

	Método NeL3					Método NeLA4				
α_k	$\bar{x}$	$\lvert f_1(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_1(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
-	$\bar{x}_3$	4.13×10^{-13}	4.10×10^{-6}	11	2.89	$\bar{x}_1$	6.53×10^{-15}	1.51×10^{-7}	3	4.00

Tabla 3.1: Resultados de los métodos fraccionarios de tipo Newton-Raphson escalares para $f_1(x)$, con estimación inicial $x_0 = 1$

la convergencia es cuadrática, como se establece en la Observación 3.4.3. En los métodos NeL3 y NeLA44 se observa que se requieren menos iteraciones que en los demás esquemas, y que el orden es tres y cuatro, respectivamente. En general, podemos ver que se puede obtener diferentes soluciones al variar el valor de α para una misma estimación inicial x_0, y que es posible conseguir raíces complejas partiendo de estimaciones iniciales reales.

Nuestra segunda función es $f_2(x) = ix^{1.8} - x^{0.9} - 16$, con raíces $\bar{x}_1 = 2.90807 - 4.24908i$ y $\bar{x}_2 = -3.85126 + 1.74602i$. Las derivadas necesarias para esta función son:

$$\begin{aligned}
{}_cD^{\alpha}_{0^+} f_2(x) &= i\frac{\Gamma(2.8)}{\Gamma(2.8-\alpha)}x^{1.8-\alpha} - \frac{\Gamma(1.9)}{\Gamma(1.9-\alpha)}x^{0.9-\alpha},\\
D^{\alpha}_{0^+} f_2(x) &= {}_cD^{\alpha}_{0^+} f_2(x) - 16\frac{1}{\Gamma(1-\alpha)}x^{-\alpha},\\
f_2'(x) &= 1.8ix^{0.8} - 0.9x^{-0.1},\\
f_2''(x) &= 1.44x^{-0.2} + 0.09x^{-1.1},\\
f_2'''(x) &= -0.288x^{-1.2} - 0.099x^{-2.1},\\
(T^a_{\alpha} f_2)(x) &= (x-a)^{1-\alpha} f_2'(x).
\end{aligned}$$

En la Tabla 3.2 se puede notar que en los procedimientos NeCA y NeRL el número de iteraciones que se requiere es mayor cuando α disminuye, y que ρ es lineal cuando $\alpha \neq 1$; no se muestran resultados cuando éstos no convergen en 500 iteraciones. También, en el método NeRL no se obtienen resultados para los

	Método NeCA					Método NeRL				
α	$\bar{x}$	$\lvert f_2(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_2(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_1$	3.30×10^{-14}	2.63×10^{-7}	8	2.00	$\bar{x}_1$	3.30×10^{-14}	2.63×10^{-7}	8	2.00
0.9	$\bar{x}_1$	3.41×10^{-7}	8.54×10^{-9}	53	0.98	$\bar{x}_1$	3.18×10^{-7}	8.49×10^{-9}	56	0.98
0.8	$\bar{x}_1$	3.21×10^{-6}	9.84×10^{-9}	209	1.00	$\bar{x}_2$	2.78×10^{-6}	9.74×10^{-9}	185	0.99
0.7	-	-	-	>500	-	-	-	-	>500	-
0.6	-	-	-	>500	-	-	-	-	>500	-
0.5	-	-	-	>500	-	-	-	-	>500	-
0.4	-	-	-	>500	-	-	-	-	>500	-
0.3	-	-	-	>500	-	-	-	-	-	-
0.2	-	-	-	>500	-	-	-	-	-	-
0.1	-	-	-	>500	-	-	-	-	-	-

	Método NeCO					Método NeA3				
α	$\bar{x}$	$\lvert f_2(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_2(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_1$	3.30×10^{-14}	2.63×10^{-7}	8	2.00	$\bar{x}_1$	3.31×10^{-14}	2.63×10^{-7}	8	2.00
0.9	$\bar{x}_1$	8.73×10^{-9}	1.27×10^{-4}	8	2.00	$\bar{x}_1$	7.16×10^{-15}	4.79×10^{-8}	8	3.01
0.8	$\bar{x}_1$	8.44×10^{-9}	1.22×10^{-4}	9	2.00	$\bar{x}_1$	2.22×10^{-15}	2.36×10^{-9}	8	2.93
0.7	$\bar{x}_1$	3.30×10^{-14}	4.55×10^{-9}	12	2.00	$\bar{x}_1$	3.70×10^{-9}	0.003	7	2.46
0.6	$\bar{x}_2$	3.66×10^{-15}	3.42×10^{-9}	11	2.00	$\bar{x}_1$	7.16×10^{-15}	1.97×10^{-7}	8	2.89
0.5	$\bar{x}_2$	1.64×10^{-11}	6.34×10^{-6}	8	2.00	$\bar{x}_1$	3.11×10^{-15}	1.13×10^{-7}	10	2.90
0.4	$\bar{x}_2$	9.57×10^{-15}	2.12×10^{-7}	6	2.01	$\bar{x}_1$	7.23×10^{-15}	2.63×10^{-9}	9	3.02
0.3	$\bar{x}_2$	1.85×10^{-14}	4.65×10^{-9}	6	2.00	$\bar{x}_1$	4.96×10^{-13}	1.84×10^{-4}	10	2.72
0.2	$\bar{x}_2$	2.10×10^{-10}	2.33×10^{-5}	5	1.99	$\bar{x}_2$	7.12×10^{-15}	4.59×10^{-8}	13	2.93
0.1	$\bar{x}_2$	1.07×10^{-11}	5.22×10^{-6}	5	2.02	$\bar{x}_1$	7.89×10^{-14}	1.97×10^{-5}	13	3.34

	Método NeL3					Método NeLA4				
α_k	$\bar{x}$	$\lvert f_2(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_2(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
-	$\bar{x}_2$	5.44×10^{-10}	0.0021	4	2.76	$\bar{x}_1$	1.07×10^{-14}	1.09×10^{-8}	4	3.01

Tabla 3.2: Resultados de los métodos fraccionarios de tipo Newton-Raphson escalares para $f_2(x)$, con estimación inicial $x_0 = 0.5$

valores de $\alpha = 0.3, 0.2, 0.1$ debido a que éste converge a puntos que no son solución de $f_2(x)$; esto sucede porque a pesar de que $|x_{k+1} - x_k| < 10^{-8}$, es decir, cumple con el criterio de parada, $|f(x_{k+1})| \gg 10^{-8}$. Se puede ver que en el esquema NeCO el número de iteraciones puede ser mayor, igual o menor que en el procedimiento clásico de Newton-Raphson, y que ρ se mantiene en el orden teórico. También podemos ver que en el método NeA3 el número de iteraciones requerido en general es mayor que en el esquema NeCO, y que ρ es cúbico; de nuevo, cuando $\alpha = 1$ ρ es cuadrático. En los procedimientos NeL3 y NeLA44 se puede observar que se requieren menos iteraciones que en los demás métodos, y que ρ es tres en NeL3; el orden de NeLA4 se comporta de manera cúbica para esta estimación inicial. En general, se puede ver que es posible obtener diferentes soluciones al variar el valor de α para una misma estimación inicial x_0, y que se pueden conseguir raíces complejas partiendo de estimaciones iniciales reales.

Nuestra tercera función es $f_3(x) = e^x - 1$, con única raíz real $\bar{x}_1 = 0$. Las derivadas necesarias para esta función son:

$$\begin{aligned}
cD^{\alpha}_{0^+} f_3(x) &= x^{1-\alpha} E_{1,2-\alpha}(x), \\
D^{\alpha}_{0^+} f_3(x) &= cD^{\alpha}_{0^+} f_3(x) - \frac{1}{\Gamma(1-\alpha)} x^{-\alpha}, \\
f_3'(x) &= e^x, \\
f_3''(x) &= e^x, \\
f_3'''(x) &= e^x, \\
(T^a_{\alpha} f_3)(x) &= (x-a)^{1-\alpha} f_3'(x).
\end{aligned}$$

	Método NeCA					Método NeRL				
α	$\bar{x}$	$\lvert f_3(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_3(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_1$	2.22×10^{-16}	2.08×10^{-8}	6	2.00	$\bar{x}_1$	2.22×10^{-16}	2.08×10^{-8}	6	2.00
0.9	$\bar{x}_1$	4.35×10^{-9}	4.19×10^{-8}	10	1.00	-	-	-	>500	-
0.8	$\bar{x}_1$	3.58×10^{-9}	1.66×10^{-8}	13	1.00	-	-	-	>500	-
0.7	$\bar{x}_1$	2.94×10^{-9}	8.70×10^{-9}	16	1.00	-	-	-	>500	-
0.6	$\bar{x}_1$	9.26×10^{-9}	1.96×10^{-8}	18	1.00	-	-	-	>500	-
0.5	$\bar{x}_1$	9.66×10^{-9}	1.55×10^{-8}	21	1.00	-	-	-	>500	-
0.4	$\bar{x}_1$	5.56×10^{-9}	7.07×10^{-9}	25	1.00	-	-	-	>500	-
0.3	$\bar{x}_1$	9.84×10^{-9}	1.01×10^{-8}	28	1.00	-	-	-	>500	-
0.2	$\bar{x}_1$	1.09×10^{-8}	9.21×10^{-9}	32	1.00	-	-	-	>500	-
0.1	$\bar{x}_1$	9.57×10^{-9}	6.68×10^{-9}	37	1.00	-	-	-	>500	-

	Método NeCO					Método NeA3				
α	$\bar{x}$	$\lvert f_3(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_3(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_1$	0	2.08×10^{-8}	6	2.00	$\bar{x}_1$	2.22×10^{-16}	2.08×10^{-8}	6	2.00
0.9	$\bar{x}_1$	1.78×10^{-15}	2.45×10^{-8}	6	2.00	$\bar{x}_1$	1.23×10^{-9}	9.05×10^{-4}	4	2.70
0.8	$\bar{x}_1$	3.55×10^{-15}	2.87×10^{-8}	6	2.00	$\bar{x}_1$	1.65×10^{-12}	1.25×10^{-4}	4	2.83
0.7	$\bar{x}_1$	0	3.35×10^{-8}	6	2.00	$\bar{x}_1$	9.55×10^{-15}	2.57×10^{-5}	4	2.89
0.6	$\bar{x}_1$	1.78×10^{-15}	3.91×10^{-8}	6	2.00	$\bar{x}_1$	1.39×10^{-17}	6.67×10^{-6}	4	2.93
0.5	$\bar{x}_1$	1.78×10^{-15}	4.54×10^{-8}	6	2.00	$\bar{x}_1$	1.11×10^{-16}	2.05×10^{-6}	4	2.95
0.4	$\bar{x}_1$	0	5.28×10^{-8}	6	2.00	$\bar{x}_1$	2.78×10^{-16}	7.13×10^{-7}	4	2.96
0.3	$\bar{x}_1$	3.55×10^{-15}	6.12×10^{-8}	6	2.00	$\bar{x}_1$	3.34×10^{-16}	2.74×10^{-7}	4	2.97
0.2	$\bar{x}_1$	3.55×10^{-15}	7.07×10^{-8}	6	2.00	$\bar{x}_1$	4.00×10^{-16}	1.14×10^{-7}	4	2.98
0.1	$\bar{x}_1$	5.33×10^{-15}	8.16×10^{-8}	6	2.00	$\bar{x}_1$	1.24×10^{-16}	5.06×10^{-8}	4	2.99

	Método NeL3					Método NeLA4				
α_k	$\bar{x}$	$\lvert f_3(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_3(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
-	$\bar{x}_1$	0	1.21×10^{-5}	3	3.11	-	-	-	-	-

Tabla 3.3: Resultados de los métodos fraccionarios de tipo Newton-Raphson escalares para $f_3(x)$, con estimación inicial $x_0 = 1.5$

En la Tabla 3.3 podemos observar que en el esquema NeCA el número de iteraciones es mayor cuando α disminuye, y que ρ es lineal cuando $\alpha \neq 1$; en el procedimiento NeRL no se obtivieron resultados porque éste no converge en 500 iteraciones para $\alpha \neq 1$. Podemos ver que en el método NeCO el número de iteraciones requerido iguala al del esquema de Newton-Raphson para cualquier α, y que ρ se mantiene en el orden teórico. Se puede observar que en el procedimiento NeA3 el número de iteraciones requerido es menor que en el método NeCO, y que ρ es cúbico; también, cuando $\alpha = 1$, ρ es cuadrático. En el procedimiento NeL3 se puede ver que se requieren menos iteraciones que en el método NeCO, y que ρ es tres. En el esquema NeLA4 no se obtuvieron resultados debido a que $f_3'(x) = f_3''(x) = f_3'''(x)$, lo cual provoca indeterminaciones en α_k (3.14) y a_k (3.15).

Finalmente, nuestra cuarta función es $f_4(x) = \sin 10x - 0.5x + 0.2$, con raíces reales $\bar{x}_1 = -1.4523$, $\bar{x}_2 = -1.3647$, $\bar{x}_3 = -0.87345$, $\bar{x}_4 = -0.6857$, $\bar{x}_5 = -0.27949$, $\bar{x}_6 = -0.021219$, $\bar{x}_7 = 0.31824$, $\bar{x}_8 = 0.64036$, $\bar{x}_9 = 0.91636$, $\bar{x}_{10} = 1.3035$, $\bar{x}_{11} = 1.5118$, $\bar{x}_{12} = 1.9756$ y $\bar{x}_{13} = 2.0977$. Las derivadas necesarias para esta función son:

	Método NeCA					Método NeRL				
α	$\bar{x}$	$\lvert f_4(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_4(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_{12}$	1.94×10^{-11}	7.01×10^{-7}	4	1.99	$\bar{x}_{12}$	1.94×10^{-11}	7.01×10^{-7}	4	1.99
0.9	$\bar{x}_{12}$	2.51×10^{-7}	8.57×10^{-9}	35	0.98	$\bar{x}_{12}$	2.56×10^{-7}	8.74×10^{-9}	35	0.98
0.8	$\bar{x}_{12}$	1.86×10^{-6}	9.65×10^{-9}	125	0.99	$\bar{x}_{12}$	1.91×10^{-6}	9.91×10^{-9}	125	0.99
0.7	$\bar{x}_{12}$	1.06×10^{-5}	9.99×10^{-9}	426	1.00	$\bar{x}_{12}$	1.07×10^{-5}	9.96×10^{-9}	431	1.00
0.6	-	-	-	>500	-	-	-	-	>500	-
0.5	-	-	-	>500	-	-	-	-	>500	-
0.4	-	-	-	>500	-	-	-	-	>500	-
0.3	-	-	-	>500	-	-	-	-	>500	-
0.2	-	-	-	>500	-	-	-	-	>500	-
0.1	-	-	-	>500	-	-	-	-	>500	-

	Método NeCO					Método NeA3				
α	$\bar{x}$	$\lvert f_4(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_4(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_{12}$	1.94×10^{-11}	7.01×10^{-7}	4	1.99	$\bar{x}_{12}$	1.94×10^{-11}	7.01×10^{-7}	4	1.99
0.9	$\bar{x}_{12}$	1.92×10^{-11}	6.98×10^{-7}	4	1.99	$\bar{x}_{12}$	2.78×10^{-16}	2.31×10^{-8}	4	2.80
0.8	$\bar{x}_{12}$	1.90×10^{-11}	6.96×10^{-7}	4	1.99	$\bar{x}_{12}$	2.78×10^{-16}	5.39×10^{-8}	4	2.92
0.7	$\bar{x}_{12}$	1.89×10^{-11}	6.93×10^{-7}	4	1.99	$\bar{x}_{12}$	2.78×10^{-16}	5.32×10^{-8}	4	3.14
0.6	$\bar{x}_{12}$	1.87×10^{-11}	6.90×10^{-7}	4	1.99	$\bar{x}_{12}$	3.01×10^{-9}	1.59×10^{-4}	3	7.91
0.5	$\bar{x}_{12}$	1.85×10^{-11}	6.87×10^{-7}	4	1.99	$\bar{x}_{12}$	8.16×10^{-12}	2.32×10^{-5}	3	4.38
0.4	$\bar{x}_{12}$	1.84×10^{-11}	6.85×10^{-7}	4	1.99	$\bar{x}_{12}$	4.60×10^{-13}	9.21×10^{-6}	3	3.93
0.3	$\bar{x}_{12}$	1.82×10^{-11}	6.82×10^{-7}	4	1.99	$\bar{x}_{12}$	6.80×10^{-14}	4.97×10^{-6}	3	3.74
0.2	$\bar{x}_{12}$	1.81×10^{-11}	6.79×10^{-7}	4	1.99	$\bar{x}_{12}$	1.58×10^{-14}	3.15×10^{-6}	3	3.63
0.1	$\bar{x}_{12}$	1.79×10^{-11}	6.76×10^{-7}	4	1.99	$\bar{x}_{12}$	3.61×10^{-15}	2.20×10^{-6}	3	3.55

	Método NeL3					Método NeLA4				
α_k	$\bar{x}$	$\lvert f_4(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_4(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
-	-	-	-	-	-	$\bar{x}_{13}$	1.72×10^{-15}	1.15×10^{-5}	3	4.33

Tabla 3.4: Resultados de los métodos fraccionarios de tipo Newton-Raphson escalares para $f_4(x)$, con estimación inicial $x_0 = 2$

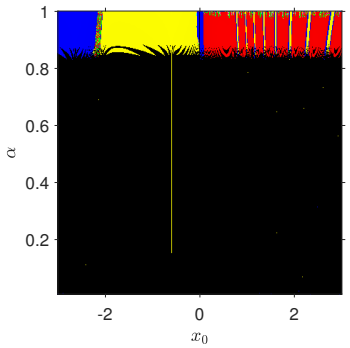
$$
\begin{aligned}
{}_cD^{\alpha}_{0^+} f_4(x) &= 5x^{1-\alpha}\left(E_{1,2-\alpha}(10ix) + E_{1,2-\alpha}(-10ix)\right) - \frac{0.5\Gamma(2)}{\Gamma(2-\alpha)}x^{1-\alpha},\\
D^{\alpha}_{0^+} f_4(x) &= {}_cD^{\alpha}_{0^+} f_4(x) + 0.2\frac{1}{\Gamma(1-\alpha)}x^{-\alpha},\\
f_4'(x) &= 10\cos 10x - 0.5,\\
f_4''(x) &= -100\sin 10x,\\
f_4'''(x) &= -1000\cos 10x,\\
(T^a_{\alpha} f_4)(x) &= (x-a)^{1-\alpha} f_4'(x).
\end{aligned}
$$

En la Tabla 3.4 podemos ver que en los procedimientos NeCA y NeRL el número de iteraciones es mayor cuando α disminuye, y que ρ es lineal cuando $\alpha \neq 1$; no se muestran resultados cuando éstos no convergen en 500 iteraciones. Se puede observar que en el método NeCO el número de iteraciones iguala al del esquema de Newton-Raphson para cualquier α, y que ρ se mantiene en el orden teórico. Se puede observar que en el procedimiento NeA3 el número de iteraciones que se requiere es menor que en el método NeCO cuando $\alpha < 0.7$, y que ρ puede ser mayor que tres; otra vez, cuando $\alpha = 1$ ρ es cuadrático. El esquema NeL3 falla debido a que el valor de x diverge al infinito. En el procedimiento NeLA4 se puede notar que se requiere una cantidad menor de iteraciones que en el método NeCO, y que ρ es aproximadamente cuatro.

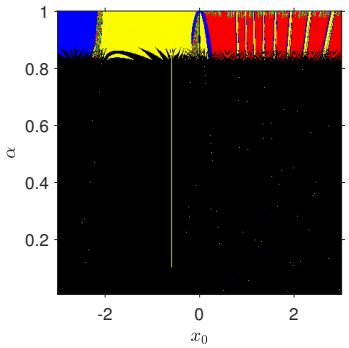
En la siguiente subsección, estudiamos la estabilidad de los esquemas fraccionarios de tipo Newton-Raphson que se han puesto a prueba, y visualizamos la dependencia de cada esquema respecto a las estimaciones iniciales.

3.5.1. Estabilidad numérica

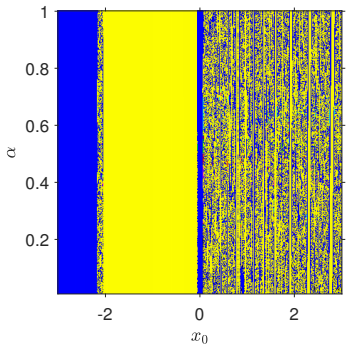
Ahora, vamos a analizar la dependencia respecto a las estimaciones iniciales de los procedimientos diseñados en este capítulo utilizando los planos de convergencia definidos en [29] para los métodos NeCA, NeRL, NeCO y NeA3 (ver ficheros MATLAB en Anexos A.12 a A.15), cuyo valor de α es fijo en cada iteración. En estos planos el eje de abscisas corresponde a la estimación inicial x_0, y el índice fraccionario α aparece en el eje de las ordenadas. Se utiliza una malla de 400×400 puntos. Los puntos que no se representan en negro corresponden a los pares de estimaciones iniciales y valores de α que convergen a una de las raíces con una tolerancia de 10^{-3}. Diferentes colores significan convergencia a diferentes raíces. Por lo tanto, cuando un punto se representa en negro, esto muestra que no se encuentra ninguna raíz en un máximo de 500 iteraciones. Además, para todos los planos de convergencia se calcula el porcentaje de pares convergentes (x_0, α), con el fin de comparar la eficiencia de los esquemas.



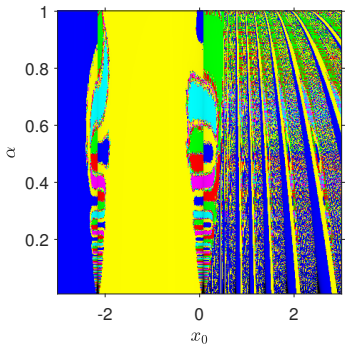
(a) NeCA, 14.59 % de convergencia



(b) NeRL, 15.46 % de convergencia



(c) NeCO, 99.93 % de convergencia



(d) NeA3, 99.76 % de convergencia

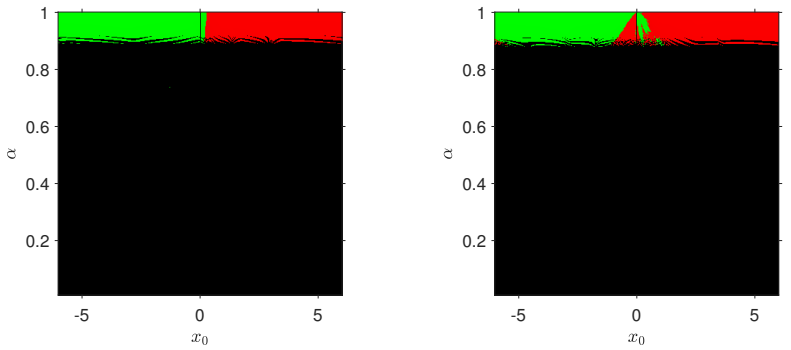
Figura 3.1: Planos de convergencia de métodos fraccionarios de tipo Newton-Raphson para $f_1(x)$

En la Figura 3.1, notamos que los procedimientos NeCA y NeRL tienen porcentajes de convergencia de alrededor de 15 %, mientras que los métodos NeCO y NeA3 alcanzan casi el 100 % de convergencia para las mismas estimaciones iniciales y los mismos valores de α; en cada caso se converge a todas las raíces.

En la Figura 3.2, observamos que los esquemas NeCA y NeRL tienen porcentajes de convergencia de alrededor de 10 %, mientras que los métodos NeCO y NeA3 alcanzan casi el 100 % de convergencia; de nuevo, en cada caso se converge a todas las raíces.

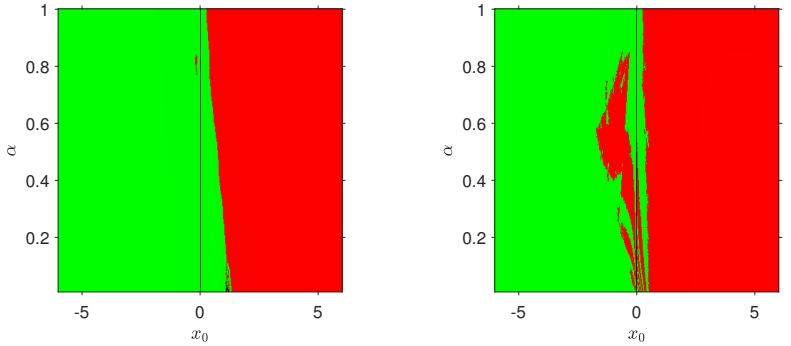
En la Figura 3.3, vemos que los esquemas NeCA, NeCO y NeA3 tienen porcentajes de convergencia de alrededor de 85 %, 73 % y 60 %, respectivamente, mientras que el procedimiento NeRL casi no alcanza a converger para las mismas estimaciones iniciales y los mismos valores de α. En el caso de NeCO y NeA3, considerando que $-10 \leq x_0 \leq 10$, y $a = -10$, resaltamos que la estabilidad podría ser ligeramente mejorada si escogemos $a < -10$.

En la Figura 3.4, podemos notar que los métodos NeCA y NeRL tienen porcentajes de convergencia de alrededor de 10 %, mientras que los esquemas NeCO y NeA3 convergen alrededor de un 50 % y 43 %, respec-



(a) NeCA, 9.88 % de convergencia

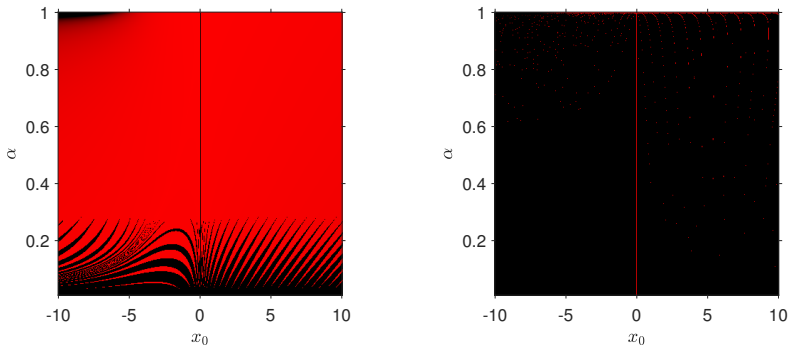
(b) NeRL, 10.65 % de convergencia



(c) NeCO, 99.74 % de convergencia

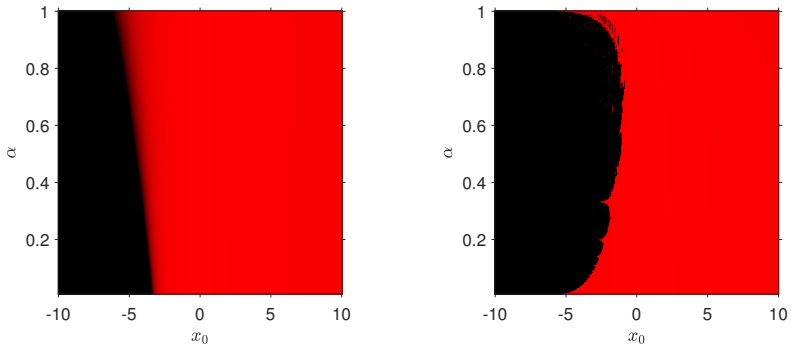
(d) NeA3, 99.75 % de convergencia

Figura 3.2: Planos de convergencia de métodos fraccionarios de tipo Newton-Raphson para $f_2(x)$



(a) NeCA, 85.52 % de convergencia

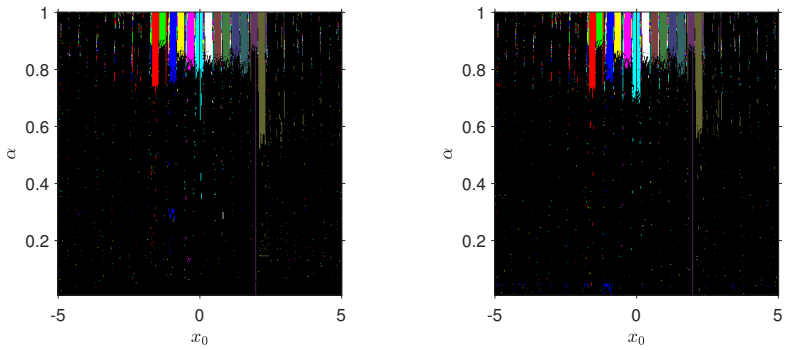
(b) NeRL, 0.96 % de convergencia



(c) NeCO, 73.19 % de convergencia

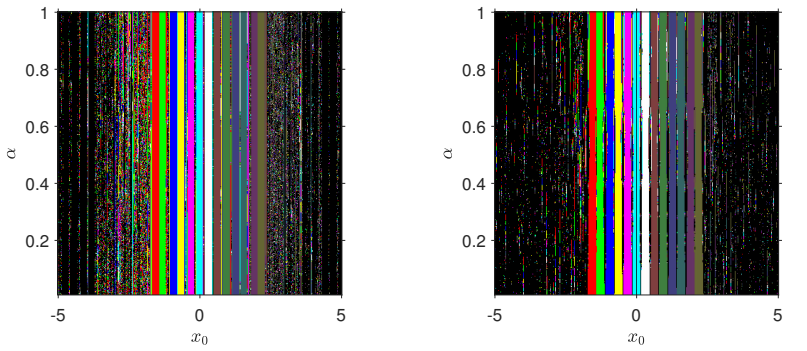
(d) NeA3, 59.26 % de convergencia

Figura 3.3: Planos de convergencia de métodos fraccionarios de tipo Newton-Raphson para $f_3(x)$



(a) NeCA, 9.10 % de convergencia

(b) NeRL, 9.27 % de convergencia



(c) NeCO, 58.25 % de convergencia

(d) NeA3, 42.96 % de convergencia

Figura 3.4: Planos de convergencia de métodos fraccionarios de tipo Newton-Raphson para $f_4(x)$

tivamente; de nuevo, en cada caso se converge a todas las raíces.

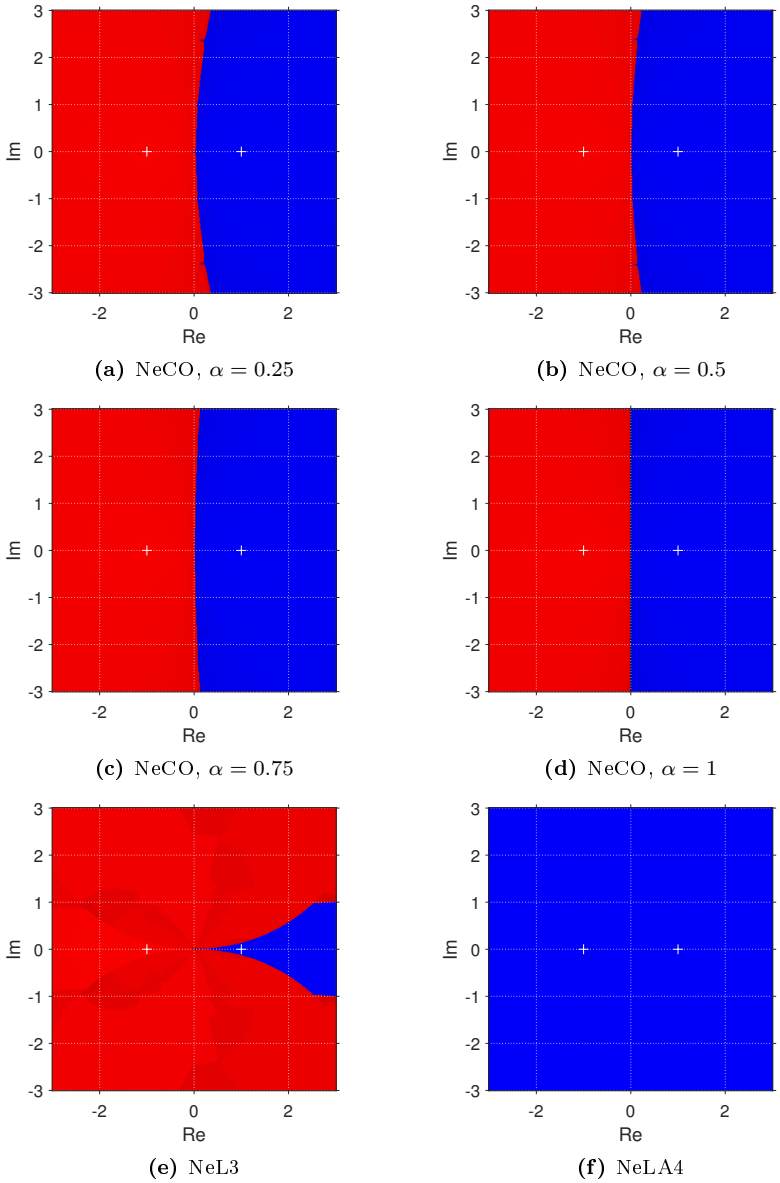
Para visualizar la dependencia respecto a las estimaciones iniciales de los procedimientos NeL3 y NeLA4, cuyo valor de α se recalcula en cada iteración, utilizaremos planos dinámicos (ver ficheros MATLAB en Anexos A.16 a A.18). Aunque el propósito de esta memoria no es hacer un estudio dinámico de los métodos, aprovecharemos para también observar cómo varían los planos dinámicos del esquema NeCO para diferentes valores de α. Estos planos se construyen considerando la parte real de la estimación inicial x_0 en el eje horizontal, y la parte imaginaria en el eje vertical. Como en los planos de convergencia, utilizamos una malla de 400×400, tolerancia de 10^{-3}, y un máximo de 500 iteraciones. En estos casos se consideran polinomios (reales o complejos) cuyas raíces son conocidas, las cuales se representan como "cruces blancas" en cada plano.

Nuestros polinomios con sus respectivas raíces son:

1. $p_1(z) = z^2 - 1$, con $\bar{z}_1 = -1$ y $\bar{z}_2 = 1$.
2. $p_2(z) = z^3 - 1$, con $\bar{z}_1 = 1$, $\bar{z}_2 = -1/2 + (\sqrt{3}/2)i$ y $\bar{z}_3 = -1/2 - (\sqrt{3}/2)i$.
3. $p_3(z) = z^2 - i$, con $\bar{z}_1 = \sqrt{2}/2 + (\sqrt{2}/2)i$ y $\bar{z}_2 = -\sqrt{2}/2 - (\sqrt{2}/2)i$.
4. $p_4(z) = z^3 + 4z^2 - 10$, con $\bar{z}_1 \approx 1.3652$, $\bar{z}_2 \approx -2.6826 + 0.3583i$ y $\bar{z}_3 \approx -2.6826 - 0.3583i$.

En general, se observa que con el método NeCO el tamaño de las cuencas de atracción de las raíces cambia ligeramente de tamaño con respecto al caso clásico (cuando $\alpha = 1$), sin zonas de no convergencia (zonas negras) como consecuencia.

Por otro lado, se puede ver que con los esquemas NeL3 y NeLA4 la tendencia es, en general, a converger a una sola raíz; en el caso del procedimiento NeL3 aparecen muchas zonas de no convergencia.



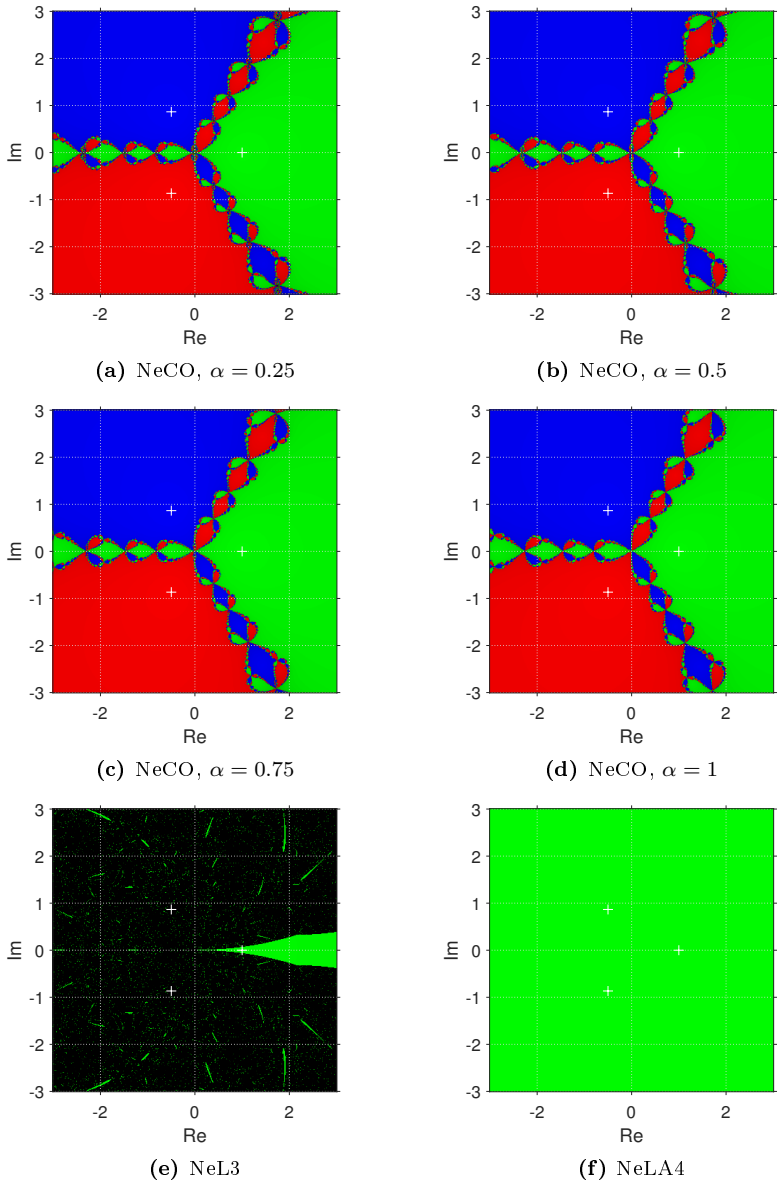
(a) NeCO, $\alpha = 0.25$ **(b)** NeCO, $\alpha = 0.5$

(c) NeCO, $\alpha = 0.75$ **(d)** NeCO, $\alpha = 1$

(e) NeL3 **(f)** NeLA4

Figura 3.5: Planos dinámicos de métodos fraccionarios de tipo Newton-Raphson para $p_1(z)$

En este capítulo, hemos visto el diseño, análisis de convergencia y estudio de la estabilidad de diferentes métodos fraccionarios de tipo Newton-Raphson escalares. Se ha observado que en los esquemas fraccionarios es posible alcanzar un orden de convergencia teórico que iguala al del caso clásico; algunos con orden de convergencia mayor que dos. En la práctica, hemos notado que se puede converger en menos iteraciones que en el caso clásico, y ρ puede ser ligeramente superior; también se ha observado, en general, que es posible converger a raíces complejas con estimaciones iniciales reales, y que se puede obtener una raíz diferente al cambiar el valor del índice fraccionario α. En el siguiente capítulo, vemos el diseño, análisis de convergencia y estudio de la estabilidad de la versión vectorial del método NeCO.

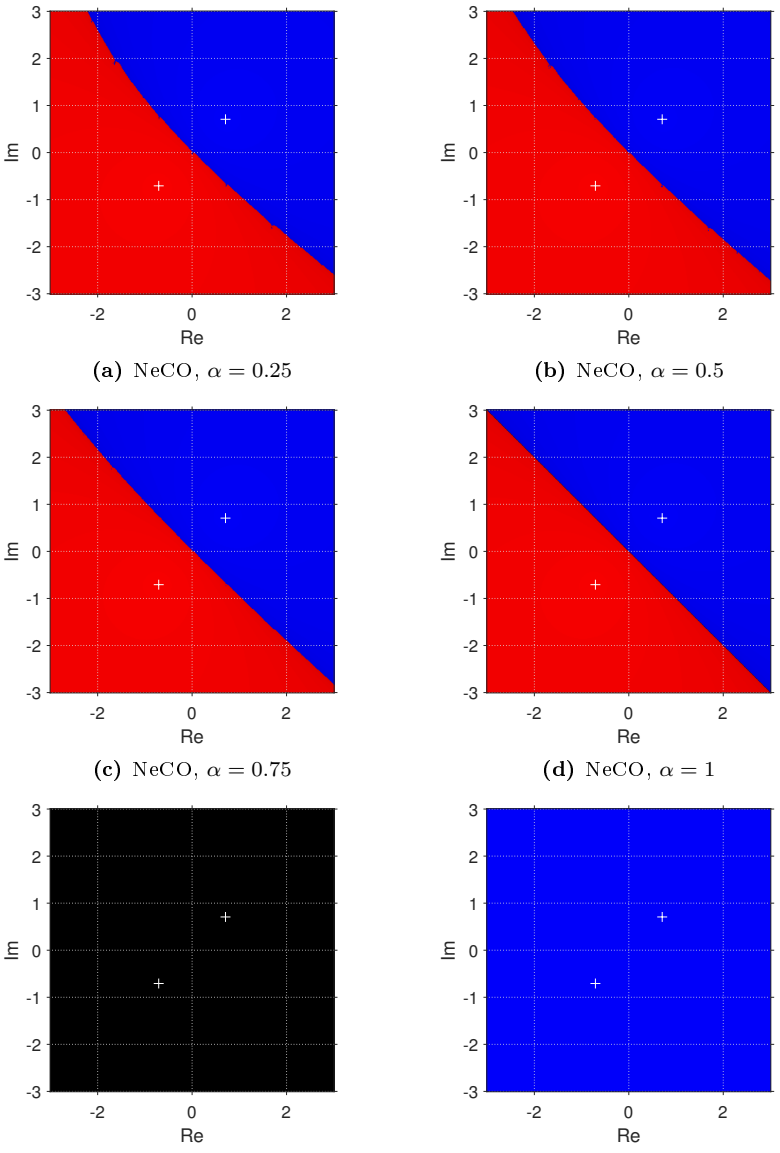


(a) NeCO, $\alpha = 0.25$ **(b)** NeCO, $\alpha = 0.5$

(c) NeCO, $\alpha = 0.75$ **(d)** NeCO, $\alpha = 1$

(e) NeL3 **(f)** NeLA4

Figura 3.6: Planos dinámicos de métodos fraccionarios de tipo Newton-Raphson para $p_2(z)$

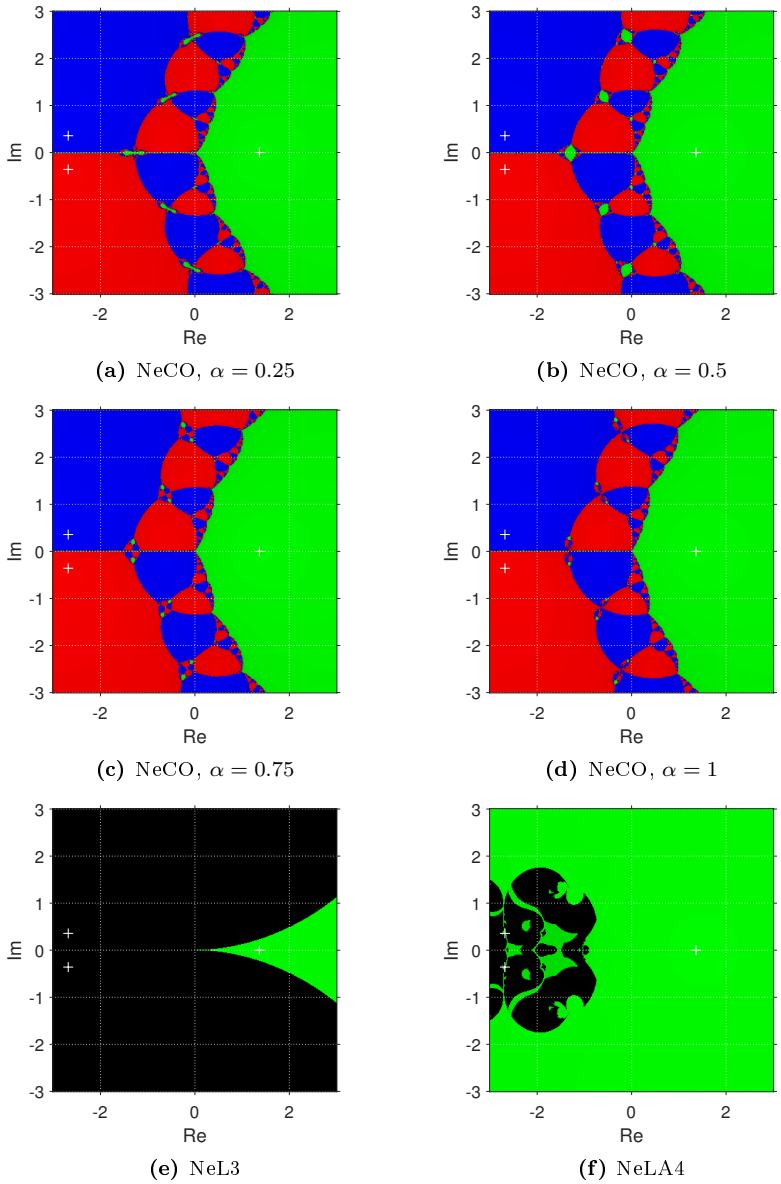


(a) NeCO, $\alpha = 0.25$ **(b)** NeCO, $\alpha = 0.5$

(c) NeCO, $\alpha = 0.75$ **(d)** NeCO, $\alpha = 1$

(e) NeL3 **(f)** NeLA4

Figura 3.7: Planos dinámicos de métodos fraccionarios de tipo Newton-Raphson para $p_3(z)$



(a) NeCO, $\alpha = 0.25$ **(b)** NeCO, $\alpha = 0.5$

(c) NeCO, $\alpha = 0.75$ **(d)** NeCO, $\alpha = 1$

(e) NeL3 **(f)** NeLA4

Figura 3.8: Planos dinámicos de métodos fraccionarios de tipo Newton-Raphson para $p_4(z)$

Capítulo 4

Método iterativo fraccionario vectorial de tipo Newton-Raphson

"Las matemáticas son el arte de dar el mismo nombre a diferentes cosas"

Henri Poincaré (1854 - 1912)

En este capítulo mostramos el diseño, análisis de convergencia y estudio de la estabilidad de una variante fraccionaria de tipo Newton-Raphson vectorial con matriz Jacobiana fraccionaria conformable[1] .

4.1. Estado del arte

Pretendemos estimar la solución $\tilde{x} \in \mathbb{R}^n$ del sistema no lineal $F(x) = \hat{0}$, donde $F : \mathbb{R}^n \to \mathbb{R}^n$, con funciones coordenadas $f_1, \ldots, f_n$. En [41] podemos encontrar el trabajo de unos autores que trasladaron directamente la expresión iterativa del método de Newton-Raphson clásico vectorial al caso fraccionario, donde numéricamente el orden sólo se mantiene en el caso del esquema clásico de Newton-Raphson, y el orden de convergencia teórico no se demuestra. En esta sección, utilizamos la propia notación de los autores en [41], donde primero definen el concepto de matriz Jacobiana fraccionaria de una función $F : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}^n$ como

$$F^{(\alpha)}(x) := \left(\partial_j^\alpha f_k(x)\right), \tag{4.1}$$

$x \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$, $j, k \in \mathbb{N}$, siendo

$$\partial_j^\alpha f_k(x) := \frac{\partial^\alpha}{\partial x_j^\alpha} f_k(x), \quad 1 \leq j, k \leq n, \tag{4.2}$$

donde $f_k : \mathbb{R}^n \to \mathbb{R}$, y el operador $\partial^\alpha / \partial x_j^\alpha$ denota cualquier derivada fraccionaria respecto a x_j, es decir, una derivada parcial fraccionaria, la cual satisface la siguiente condición de continuidad con respecto al orden α de la derivada

$$\lim_{\alpha \to 1} \frac{\partial^\alpha}{\partial x_j^\alpha} f_k(x) = \frac{\partial}{\partial x_j} f_k(x), \quad 1 \leq j, k \leq n. \tag{4.3}$$

[1] Los resultados de este capítulo forman parte del artículo *Generalized conformable fractional Newton-type method for solving nonlinear systems* publicado en Numerical Algorithms, Springer [12], y presentados en el congreso MME&HB 2022, Valencia, España

Por tanto, (4.1) satisface

$$\lim_{\alpha \to 1} F^{(\alpha)}(x) = F^{(1)}(x), \tag{4.4}$$

donde $F^{(1)}(x)$ es la matriz Jacobiana clásica de F. Con (4.1), los autores diseñan el esquema de tipo Newton-Raphson fraccionario para sistemas

$$x_{i+1} = x_i - \left(F^{(\alpha)}(x_i)\right)^{-1} F(x_i), \quad i = 0, 1, 2, \ldots, \tag{4.5}$$

cuyo análisis de convergencia no se proporciona. Para garantizar el orden de convergencia cuadrático del procedimiento clásico de Newton-Raphson para sistemas, los autores definen la función

$$\alpha_F(x) := \begin{cases} \alpha, & \text{si } ||F(x)|| \geq \delta \text{ y } ||x|| \neq 0, \\ 1, & \text{si } ||F(x)|| < \delta \text{ o } ||x|| = 0, \end{cases} \tag{4.6}$$

y sustituyen a α en (4.5) por (4.6), obteniendo el segundo método

$$x_{i+1} = x_i - \left(F^{(\alpha_F(x_i))}(x_i)\right)^{-1} F(x_i), \quad i = 0, 1, 2, \ldots, \tag{4.7}$$

donde la convergencia cuadrática está garantizada sólo si $||F(x)|| < \delta$, $\delta \in \mathbb{R}$, $\delta > 0$; obviamente, en este caso $\alpha_F(x) = 1$, por lo que se utiliza el esquema clásico de Newton-Raphson para sistemas cuando $||F(x)|| < \delta$.

Para las pruebas numéricas en [41], los autores utilizan $-2 < \alpha < 2, \alpha \neq -1, 0, 1$ para probar algunos sistemas no lineales de 2×2 y 3×3, y se puede observar que estos procedimientos pueden converger a diferentes raíces con una misma estimación inicial al variar el valor de α, aunque con velocidad de convergencia lineal.

Aunque los autores en [41] proponen más variantes fraccionarias de tipo Newton-Raphson para sistemas, asegurando tener orden de convergencia al menos lineal, no se mencionan en esta memoria porque para ninguno de estos métodos se realiza un análisis de convergencia, al igual que con (4.5) y (4.7).

En la siguiente sección, presentamos los conceptos y resultados que fueron necesarios para diseñar la versión vectorial del esquema NeCO visto en el Capítulo 3, es decir, el procedimiento fraccionario conformable de tipo Newton-Raphson para sistemas que publicamos en la revista *Numerical Algorithms* (ver [12]). Hasta donde sabemos, es el primer procedimiento iterativo vectorial con derivadas conformables cuyo análisis de convergencia ha sido llevado a cabo, además de su estabilidad.

4.2. Nuevos conceptos y resultados

Considerando que en la definición de derivada parcial conformable proporcionada en (2.50), $x_i \in (0, \infty)$, podemos extender la definición de derivada parcial conformable al caso $a \in \mathbb{R}$, donde $x_i \in (a, \infty)$, de la siguiente manera:

Definición 4.2.1 *Sea $f : \mathbb{R}^n \to \mathbb{R}$ una función en n variables, $x_1, \ldots, x_n$, la derivada parcial conformable de f de orden $0 < \alpha \leq 1$ para $x_i \in (a, \infty)$, $a \in \mathbb{R}$, se define como*

$$\frac{\partial_a^\alpha}{\partial x_i^\alpha} f(x_1, \ldots, x_n) = \lim_{\epsilon \to 0} \frac{f(x_1, \ldots, x_i + \epsilon(x_i - a)^{1-\alpha}, \ldots, x_n) - f(x_1, \ldots, x_n)}{\epsilon}. \tag{4.8}$$

En caso de que $x_i = a$, $\dfrac{\partial_a^\alpha}{\partial x_i^\alpha} f(x_1, \ldots, a, \ldots, x_n) = \lim\limits_{x_i \to a^+} \dfrac{\partial_a^\alpha}{\partial x_i^\alpha} f(x_1, \ldots, x_i, \ldots, x_n)$.

Esta derivada es lineal, y las reglas del producto, el cociente y de la cadena se satisfacen, analogamente a la derivada conformable definida en [1]. En el siguiente resultado establecemos una relación entre la derivada parcial clásica y la derivada parcial conformable:

Teorema 4.2.1 *Sea f una función diferenciable en n variables, $x_1, \ldots, x_n$, $x_i > a$, entonces,*

$$\frac{\partial_a^\alpha}{\partial x_i^\alpha} f(x_1, \ldots, x_n) = (x_i - a)^{1-\alpha} \frac{\partial}{\partial x_i} f(x_1, \ldots, x_n). \tag{4.9}$$

Demostración. Sea $h = \epsilon(x_i - a)^{1-\alpha}$, y por tanto $\epsilon = h(x_i - a)^{\alpha-1}$, entonces

$$\begin{aligned}
\frac{\partial_a^\alpha}{\partial x_i^\alpha} f(x_1, \ldots, x_n) &= \lim_{h \to 0} \frac{f(x_1, \ldots, x_i + h, \ldots, x_n) - f(x_1, \ldots, x_n)}{h(x_i - a)^{\alpha-1}} \\
&= (x_i - a)^{1-\alpha} \lim_{h \to 0} \frac{f(x_1, \ldots, x_i + h, \ldots, x_n) - f(x_1, \ldots, x_n)}{h} \\
&= (x_i - a)^{1-\alpha} \frac{\partial}{\partial x_i} f(x_1, \ldots, x_n).
\end{aligned}$$

Esto completa la prueba. □

También podemos definir la matriz Jacobiana conformable como en (2.51), pero para $x_1 \in (a_1, \infty), x_2 \in (a_2, \infty), \ldots, x_n \in (a_n, \infty)$, donde $x = (x_1, x_2, \ldots, x_n)$ y $\hat{a} = (a_1, a_2, \ldots, a_n)$:

Definición 4.2.2 *Sean $f_1, f_2, \ldots, f_n$ funciones coordenada de una función vectorial diferenciable $F : \mathbb{R}^n \to \mathbb{R}^n$ en las variables $x_1 > a_1, x_2 > a_2, \ldots, x_n > a_n$, donde $x = (x_1, x_2, \ldots, x_n)$ y $\hat{a} = (a_1, a_2, \ldots, a_n)$, tal que sus respectivas derivadas parciales existen y son continuas. Entonces, la matriz Jacobian conformable viene dada por*

$$\begin{aligned}
F_{\hat{a}}^{\alpha(1)}(x) &= \begin{pmatrix}
\dfrac{\partial_{a_1}^\alpha f_1}{\partial x_1^\alpha} & \dfrac{\partial_{a_2}^\alpha f_1}{\partial x_2^\alpha} & \cdots & \dfrac{\partial_{a_n}^\alpha f_1}{\partial x_n^\alpha} \\
\dfrac{\partial_{a_1}^\alpha f_2}{\partial x_1^\alpha} & \dfrac{\partial_{a_2}^\alpha f_2}{\partial x_2^\alpha} & \cdots & \dfrac{\partial_{a_n}^\alpha f_2}{\partial x_n^\alpha} \\
\vdots & \vdots & \ddots & \vdots \\
\dfrac{\partial_{a_1}^\alpha f_n}{\partial x_1^\alpha} & \dfrac{\partial_{a_2}^\alpha f_n}{\partial x_2^\alpha} & \cdots & \dfrac{\partial_{a_n}^\alpha f_n}{\partial x_n^\alpha}
\end{pmatrix} \\
&= \begin{pmatrix}
(x_1 - a_1)^{1-\alpha} \dfrac{\partial f_1}{\partial x_1} & (x_2 - a_2)^{1-\alpha} \dfrac{\partial f_1}{\partial x_2} & \cdots & (x_n - a_n)^{1-\alpha} \dfrac{\partial f_1}{\partial x_n} \\
(x_1 - a_1)^{1-\alpha} \dfrac{\partial f_2}{\partial x_1} & (x_2 - a_2)^{1-\alpha} \dfrac{\partial f_2}{\partial x_2} & \cdots & (x_n - a_n)^{1-\alpha} \dfrac{\partial f_2}{\partial x_n} \\
\vdots & \vdots & \ddots & \vdots \\
(x_1 - a_1)^{1-\alpha} \dfrac{\partial f_n}{\partial x_1} & (x_2 - a_2)^{1-\alpha} \dfrac{\partial f_n}{\partial x_2} & \cdots & (x_n - a_n)^{1-\alpha} \dfrac{\partial f_n}{\partial x_n}
\end{pmatrix}.
\end{aligned} \tag{4.10}$$

Para el análisis de convergencia de sistemas no lineales con el uso de serie de Taylor conformable, podemos usar la siguiente notación, análoga a la dada en la Definición 2.1.3:

Definición 4.2.3 *Sea $F : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ suficientemente α-diferenciable en D. La q-ésima derivada conformable de F en $u \in \mathbb{R}^n$ es la función $\alpha(q)$-lineal $F_{\hat{a}}^{\alpha(q)}(u) : \mathbb{R}^n \times \cdots \times \mathbb{R}^n \to \mathbb{R}^n$ tal que $F_{\hat{a}}^{\alpha(q)}(u)(v_1, \ldots, v_q) \in \mathbb{R}^n$. Se puede observar que:*

1. $F_{\hat{a}}^{\alpha(q)}(u)(v_1, \ldots, v_{q-1}, \cdot) \in \mathcal{L}(\mathbb{R}^n)$, *siendo* $\mathcal{L}(\mathbb{R}^n)$ *el espacio de las aplicaciones lineales de* $\mathbb{R}^n \to \mathbb{R}^n$.
2. $F_{\hat{a}}^{\alpha(q)}(u)(v_{\sigma_1}, \ldots, v_{\sigma_q}) = F_{\hat{a}}^{\alpha(q)}(u)(v_1, \ldots, v_q)$, *para cualquier permutación* $\sigma \in \{1, \ldots, q\}$.

Utilizando las propiedades anteriores, definimos la siguiente notación:

1. $F_{\hat{a}}^{\alpha(q)}(u)(v_1, \ldots, v_q) = F_{\hat{a}}^{\alpha(q)}(u)v_1 \cdots v_q$.
2. $F_{\hat{a}}^{\alpha(q)}(u)v^{q-1}F_{\hat{a}}^{\alpha(p)}(u)v^p = F_{\hat{a}}^{\alpha(q)}(u)F_{\hat{a}}^{\alpha(p)}(u)v^{q+p-1}$.

Con el propósito de definir una serie de Taylor conformable para una función vectorial, procedamos de manera similar al Teorema 4.1 de [1].

Teorema 4.2.2 *Supongamos que $F : \mathbb{R}^n \to \mathbb{R}^n$ es una función vectorial infinitamente α-diferenciable, para algún $\alpha \in (0, 1]$, en el entorno de un punto $t_0 \in \mathbb{R}^n$. Entonces, F puede desarrollarse en serie de Taylor conformable*

$$F(t) = \sum_{k=0}^{\infty} \frac{F_{t_0}^{\alpha(k)}(t_0)(t - t_0)^{k\alpha}}{\alpha^k k!}, \tag{4.11}$$

donde $F_{t_0}^{\alpha(k)}(t_0)$ significa la aplicación de la derivada conformable k veces.

Demostración. Sea $F(t) = K_0 + K_1(t - t_0)^{\alpha} + K_2(t - t_0)^{2\alpha} + K_3(t - t_0)^{3\alpha} + \cdots$. Entonces, $F(t_0) = K_0$.

Si aplicamos la derivada conformable una vez a F, y evaluamos en t_0, obtenemos $F_{t_0}^{\alpha(1)}(t_0) = K_1\alpha$, y así, $K_1 = \dfrac{F_{t_0}^{\alpha(1)}(t_0)}{\alpha}$.

Aplicando dos veces la derivada conformable a F, y evaluamos en t_0, obtenemos $F_{t_0}^{\alpha(2)}(t_0) = 2K_2\alpha^2$, y así, $K_2 = \dfrac{F_{t_0}^{\alpha(2)}(t_0)}{2\alpha^2}$.

Procediendo inductivamente,

$$K_n = \frac{F_{t_0}^{\alpha(n)}(t_0)}{\alpha^n n!}, \ n \in \mathbb{N}. \tag{4.12}$$

De esta manera, obtenemos (4.11). □

En consecuencia, $F(t)$ se puede escribir como

$$F(t) = F(t_0) + \frac{F_{t_0}^{\alpha(1)}(t_0)}{\alpha}(t - t_0)^{\alpha} + \frac{F_{t_0}^{\alpha(2)}(t_0)}{2\alpha^2}(t - t_0)^{2\alpha} + \ldots$$

Como podemos ver, las derivadas conformables están centradas en t_0, el cual es el mismo punto donde se evalúan. Esto es un problema que debemos evitar para diseñar el método fraccionario conformable de tipo Newton-Raphson vectorial.

Procediendo de forma similar al Teorema 4.1 de [40], podemos obtener una nueva serie de Taylor con el uso del Teorema 4.2.2, donde las derivadas conformables están centradas en algún punto $\hat{a} = (a_1, \ldots, a_n) \in \mathbb{R}^n$ distinto de otro punto $\hat{b} = (b_1, \ldots, b_n) \in \mathbb{R}^n$ donde se evalúan:

Teorema 4.2.3 *Sea $F : \mathbb{R}^n \to \mathbb{R}^n$ una función vectorial infinitamente α-diferenciable, para algún $\alpha \in (0, 1]$, en el entorno de un punto $b_i \in (a_i, \infty)$, $\forall i = 1, \ldots, n$, donde $\hat{a} = (a_1, \ldots, a_n) \in \mathbb{R}^n$ y $\hat{b} = (b_1, \ldots, b_n) \in \mathbb{R}^n$. Entonces, F puede desarrollarse en serie de Taylor conformable*

$$F(t) = F(\hat{b}) + \frac{F_{\hat{a}}^{\alpha(1)}(\hat{b})}{\alpha}\Delta + \frac{F_{\hat{a}}^{\alpha(2)}(\hat{b})}{2!\alpha^2}\Delta^2 + \cdots, \tag{4.13}$$

donde $\Delta = U^{\odot\alpha} - V^{\odot\alpha}$; $U = t - \hat{a}$, $V = \hat{b} - \hat{a}$, siendo $\odot$ el producto de Hadamard.

Demostración. Denotemos $t_0 = \hat{a}$ en (4.11),

$$F(t) = F(\hat{a}) + \frac{F_{\hat{a}}^{\alpha(1)}(\hat{a})}{\alpha}(t - \hat{a})^{\alpha} + \frac{F_{\hat{a}}^{\alpha(2)}(\hat{a})}{2\alpha^2}(t - \hat{a})^{2\alpha} + \cdots \tag{4.14}$$

Evaluando (4.14) en $\hat{b}$,

$$F(\hat{b}) = F(\hat{a}) + \frac{F_{\hat{a}}^{\alpha(1)}(\hat{a})}{\alpha}(\hat{b} - \hat{a})^{\alpha} + \frac{F_{\hat{a}}^{\alpha(2)}(\hat{a})}{2\alpha^2}(\hat{b} - \hat{a})^{2\alpha} + \cdots, \tag{4.15}$$

despejando $F(\hat{a})$, tenemos

$$F(\hat{a}) = F(\hat{b}) - \frac{F_{\hat{a}}^{\alpha(1)}(\hat{a})}{\alpha}(\hat{b} - \hat{a})^{\alpha} - \frac{F_{\hat{a}}^{\alpha(2)}(\hat{a})}{2\alpha^2}(\hat{b} - \hat{a})^{2\alpha} - \cdots \tag{4.16}$$

Si aplicamos la derivada conformable una y dos veces a F, centradas en $\hat{a}$, obtenemos, respectivamente,

$$F_{\hat{a}}^{\alpha(1)}(\hat{a}) = F_{\hat{a}}^{\alpha(1)}(\hat{b}) - \frac{F_{\hat{a}}^{\alpha(2)}(\hat{a})}{\alpha}(\hat{b} - \hat{a})^{\alpha} - \frac{F_{\hat{a}}^{\alpha(3)}(\hat{a})}{2\alpha^2}(\hat{b} - \hat{a})^{2\alpha} - \ldots \tag{4.17}$$

y

$$F_{\hat{a}}^{\alpha(2)}(\hat{a}) = F_{\hat{a}}^{\alpha(2)}(\hat{b}) - \frac{F_{\hat{a}}^{\alpha(3)}(\hat{a})}{\alpha}(\hat{b} - \hat{a})^{\alpha} - \frac{F_{\hat{a}}^{\alpha(4)}(\hat{a})}{2\alpha^2}(\hat{b} - \hat{a})^{2\alpha} - \ldots \tag{4.18}$$

Sustituyendo todas las derivadas evaluadas en $\hat{a}$ de (4.14), por sus expresiones en (4.16), (4.17) y (4.18) obtenemos (4.13), la cual se puede escribir como

$$\begin{aligned} F(t) &= F(\hat{b}) + \frac{F_{\hat{a}}^{\alpha(1)}(\hat{b})}{\alpha}\left[(t - \hat{a})^{\odot\alpha} - (\hat{b} - \hat{a})^{\odot\alpha}\right] + \frac{F_{\hat{a}}^{\alpha(2)}(\hat{b})}{2\alpha^2}\left[(t - \hat{a})^{\odot\alpha} - (\hat{b} - \hat{a})^{\odot\alpha}\right]^2 + \cdots, \\ &= F(\hat{b}) + \frac{F_{\hat{a}}^{\alpha(1)}(\hat{b})}{\alpha}\Delta + \frac{F_{\hat{a}}^{\alpha(2)}(\hat{b})}{2\alpha^2}\Delta^2 + \cdots, \end{aligned}$$

y esto completa la prueba. □

Observación 4.2.1 *Con estas expresiones, podemos escribir el desarrollo de Taylor conformable de F en el entorno de la raíz $\tilde{x}$, siendo la matriz Jacobiana conformable $F_{\hat{a}}^{\alpha(1)}(\tilde{x})$ no singular, como se muestra a continuación:*

$$F\left(x^{(k)}\right) = \frac{F_{\hat{a}}^{\alpha(1)}(\tilde{x})}{\alpha}\left[\Delta + \hat{C}_2^{CO}\Delta^2 + \hat{C}_3^{CO}\Delta^3 + \cdots + \hat{C}_p^{CO}\Delta^p\right] + O\left(e^{(k)p+1}\right), \tag{4.19}$$

donde $\Delta = U^{\odot\alpha} - V^{\odot\alpha}$; $U = x^{(k)} - \hat{a}$, $V = \tilde{x} - \hat{a}$, $e^{(k)} = x^{(k)} - \tilde{x}$, y siendo $\odot$ el producto de Hadamard, $\hat{C}_1^{CO} = I_n$, $\hat{C}_q^{CO} = \dfrac{1}{q!\alpha^{q-1}}\left[F_{\hat{a}}^{\alpha(1)}(\tilde{x})\right]^{-1} F_{\hat{a}}^{\alpha(q)}(\tilde{x})$, $q \geq 2$.

Observación 4.2.2 *Con el uso de la Definición 4.2.3, el Teorema 4.2.1 y la potencia de Hadamard obtenemos*

$$F_{\hat{a}}^{\alpha(1)}(x) = (x - \hat{a})^{\odot(1-\alpha)} F'(x), \tag{4.20}$$

y

$$F_{\hat{a}}^{\alpha(1)}(\hat{a}) = \lim_{x \to \hat{a}^+} (x - \hat{a})^{\odot(1-\alpha)} F'(x), \tag{4.21}$$

respectivamente, para una función vectorial F, siendo $F'(x)$ la matriz Jacobiana clásica. Notemos que en (4.21) $x \to \hat{a}^+$ significa que $x_i \to a_i^+$, $\forall i = 1, \ldots, n$, donde $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ y $\hat{a} = (a_1, \ldots, a_n) \in \mathbb{R}^n$.

Por otra parte, para realizar el análisis de convergencia, debemos proporcionar un resultado más: el Teorema del binomio de Newton generalizado para valores vectoriales y potencia fraccionaria. Para la definición del coeficiente binomial generalizado, véase [2].

Teorema 4.2.4 *Sean $x, y \in \mathbb{R}^n$, $r \in \mathbb{R}$, y sea $\odot$ la potencia/producto de Hadamard. El Teorema del binomio de Newton generalizado para valores vectoriales y potencia fraccionaria está dado por*

$$(x + y)^{\odot r} = \sum_{k=0}^{\infty} \binom{r}{k} x^{\odot(r-k)} \odot y^{\odot k}, \quad k \in \{0\} \cup \mathbb{N}, \tag{4.22}$$

siendo el coeficiente binomial generalizado

$$\binom{r}{k} = \frac{\Gamma(r+1)}{k!\Gamma(r-k+1)}, \quad k \in \{0\} \cup \mathbb{N}. \tag{4.23}$$

Demostración. Dado que la potencia/producto de Hadamard es una potencia/producto elemento a elemento, la prueba es análoga al caso clásico.

Fin de la prueba. □

En la siguiente sección, diseñamos el esquema conformable de tipo Newton-Raphson para la resolución de sistemas no lineales que publicamos en la revista *Numerical Algorithms* (ver [12]).

4.3. Deducción del método

Como procedimos en [11], consideremos la aproximación de la función F por medio de la serie de Taylor (4.13) hasta orden uno, evaluada en la raíz $\tilde{x}$, como se muestra:

$$F(x) \approx F(\tilde{x}) + \frac{F_{\hat{a}}^{\alpha(1)}(\tilde{x})}{\alpha}\Delta. \tag{4.24}$$

Como $F(\tilde{x}) = \hat{0}$ y $\Delta = U^{\odot\alpha} - V^{\odot\alpha}$; siendo $U = x - \hat{a}$ y $V = \tilde{x} - \hat{a}$,

$$F(x) \approx \frac{F_{\hat{a}}^{\alpha(1)}(\tilde{x})}{\alpha}\left[(x-\hat{a})^{\odot\alpha} - (\tilde{x}-\hat{a})^{\odot\alpha}\right]. \tag{4.25}$$

Multiplicando por la izquierda cada lado de (4.25), por $\alpha\left[F_{\hat{a}}^{\alpha(1)}(\tilde{x})\right]^{-1}$,

$$\alpha\left[F_{\hat{a}}^{\alpha(1)}(\tilde{x})\right]^{-1} F(x) \approx (x-\hat{a})^{\odot\alpha} - (\tilde{x}-\hat{a})^{\odot\alpha}. \tag{4.26}$$

De $(\tilde{x}-\hat{a})^{\odot\alpha}$, despejamos $\tilde{x}$, y así

$$\tilde{x} \approx \hat{a} + \left((x-\hat{a})^{\odot\alpha} - \alpha\left[F_{\hat{a}}^{\alpha(1)}(\tilde{x})\right]^{-1} F(x)\right)^{\odot 1/\alpha}. \tag{4.27}$$

Considerando los iterados $x^{(k)}$ y $x^{(k+1)}$ como aproximaciones de la raíz $\tilde{x}$, obtenemos el procedimiento de tipo Newton-Raphson conformable para sistemas no lineales, el cual denotamos por NvCO:

$$x^{(k+1)} = \hat{a} + \left[\left(x^{(k)}-\hat{a}\right)^{\odot\alpha} - \alpha\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1} F\left(x^{(k)}\right)\right]^{\odot 1/\alpha}, \quad k = 0,1,2,\ldots \tag{4.28}$$

En la siguiente sección, realizamos el análisis de convergencia del método NvCO con el uso de la serie de Taylor conformable (4.13), y con la clásica, comprobando que tiene convergencia cuadrática, independientemente del orden de la derivada fraccionaria.

4.4. Análisis de convergencia

En el siguiente resultado probamos la convergencia cuadrática del esquema NvCO con el uso de la serie de Taylor (4.13).

Teorema 4.4.1 *Sea $F : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ una función suficientemente diferenciable en un conjunto convexo abierto $D \subseteq \mathbb{R}^n$, conteniendo un cero $\tilde{x} \in \mathbb{R}^n$ de la función vectorial $F(x)$. Sea $F_{\hat{a}}^{\alpha(1)}(x)$ la matriz Jacobiana conformable de F centrada en $\hat{a} \in \mathbb{R}^n$, de orden α, para cualquier $\alpha \in (0,1]$. Supongamos que $F_{\hat{a}}^{\alpha(1)}(x)$ es continua y no singular en $\tilde{x}$. Si una aproximación inicial $x^{(0)} \in \mathbb{R}^n$ es lo suficientemente cercana a $\tilde{x}$, entonces el orden de convergencia local del procedimiento conformable de tipo Newton-Raphson vectorial (NvCO)*

$$x^{(k+1)} = \hat{a} + \left[\left(x^{(k)}-\hat{a}\right)^{\odot\alpha} - \alpha\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1} F\left(x^{(k)}\right)\right]^{\odot 1/\alpha}, \quad k = 0,1,2,\ldots$$

es al menos 2, y la ecuación del error es

$$e^{(k+1)} = \alpha\hat{C}_2^{CO}(\tilde{x}-\hat{a})^{\odot(\alpha-1)}{e^{(k)}}^2 + O\left({e^{(k)}}^3\right), \tag{4.29}$$

siendo $\hat{C}_q^{CO} = \dfrac{1}{q!\alpha^{q-1}} \left[F_{\hat{a}}^{\alpha(1)}(\tilde{x})\right]^{-1} F_{\hat{a}}^{\alpha(q)}(\tilde{x})$, $q = 2, 3, 4, \ldots$, *tal que* $\hat{a} < x^{(k)}$, $\forall k$.

Demostración. Con el uso de la notación dada en la Definición 4.2.3, el Teorema 4.2.3, y considerando $x^{(k)} = e^{(k)} + \tilde{x}$, el desarrollo de Taylor conformable de $F(x)$ en el entorno de $\tilde{x}$ es

$$\begin{aligned} F\left(x^{(k)}\right) &= \frac{F_{\hat{a}}^{\alpha(1)}(\tilde{x})}{\alpha}\left[\Delta + \hat{C}_2^{CO}\Delta^2\right] + O\left(e^{(k)^3}\right) \\ &= \frac{F_{\hat{a}}^{\alpha(1)}(\tilde{x})}{\alpha}\left[\left(\left(\tilde{x} - \hat{a} + e^{(k)}\right)^{\odot\alpha} - (\tilde{x} - \hat{a})^{\odot\alpha}\right)\right. \\ &+ \left.\hat{C}_2^{CO}\left(\left(\tilde{x} - \hat{a} + e^{(k)}\right)^{\odot\alpha} - (\tilde{x} - \hat{a})^{\odot\alpha}\right)^2\right] + O\left(e^{(k)^3}\right), \end{aligned}$$

siendo $\hat{C}_q^{CO} = \dfrac{1}{q!\alpha^{q-1}} \left[F_{\hat{a}}^{\alpha(1)}(\tilde{x})\right]^{-1} F_{\hat{a}}^{\alpha(q)}(\tilde{x})$; $q = 2, 3, 4, \ldots$

Usando el Teorema 4.2.4, y considerando la potencia de Hadamard,

$$\begin{aligned} F\left(x^{(k)}\right) &= \frac{F_{\hat{a}}^{\alpha(1)}(\tilde{x})}{\alpha}\left[\left(\alpha(\tilde{x} - \hat{a})^{\odot(\alpha-1)}\right)e^{(k)}\right. \\ &+ \left.\left(\frac{\alpha}{2}(\alpha-1)(\tilde{x} - \hat{a})^{\odot(\alpha-2)} + \alpha^2\hat{C}_2^{CO}(\tilde{x} - \hat{a})^{\odot(2\alpha-2)}\right)e^{(k)^2}\right] + O\left(e^{(k)^3}\right). \end{aligned}$$

Considerando (4.20), y usando de nuevo la Definición 4.2.3 y el Teorema 4.2.4, el desarrollo de la matriz Jacobiana conformable de $F\left(x^{(k)}\right)$ se puede expresar como

$$F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right) = \frac{F_{\hat{a}}^{\alpha(1)}(\tilde{x})}{\alpha}\left[\alpha I_n + \left(2\alpha^2\hat{C}_2^{CO}(\tilde{x} - \hat{a})^{\odot(\alpha-1)}\right)e^{(k)}\right] + O\left(e^{(k)^2}\right).$$

Podemos escribir el desarrollo de $\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1}$ como

$$\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1} = \left[\frac{1}{\alpha}I_n + \tilde{X}_2 e^{(k)}\right]\alpha\left[F_{\hat{a}}^{\alpha(1)}(\tilde{x})\right]^{-1} + O\left(e^{(k)^2}\right),$$

siendo $\tilde{X}_2$ una variable desconocida tal que $\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1} F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right) = I_n$, y así,

$$\left(2\alpha\hat{C}_2^{CO}(\tilde{x} - \hat{a})^{\odot(\alpha-1)} + \alpha\tilde{X}_2\right)e^{(k)} = \hat{0}.$$

Por tanto, para $\tilde{X}_2$,

$$\tilde{X}_2 = -2\hat{C}_2^{CO}(\tilde{x} - \hat{a})^{\odot(\alpha-1)}.$$

Entonces,

$$\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1} = \left[\frac{1}{\alpha}I_n + \left(-2\hat{C}_2^{CO}(\tilde{x} - \hat{a})^{\odot(\alpha-1)}\right)e^{(k)}\right]\alpha\left[F_{\hat{a}}^{\alpha(1)}(\tilde{x})\right]^{-1} + O\left(e^{(k)^2}\right),$$

y

$$\begin{aligned} -\alpha\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1}F\left(x^{(k)}\right) &= -\left(\alpha(\tilde{x}-\hat{a})^{\odot(\alpha-1)}\right)e^{(k)} \\ &+ \left(\alpha^2\hat{C}_2^{CO}(\tilde{x}-\hat{a})^{\odot(2\alpha-2)}\right. \\ &- \left.\frac{\alpha}{2}(\alpha-1)(\tilde{x}-\hat{a})^{\odot(\alpha-2)}\right)e^{(k)^2}+O\left(e^{(k)^3}\right). \end{aligned}$$

Entonces,

$$\begin{aligned} \left(x^{(k)}-\hat{a}\right)^{\odot\alpha}-\alpha\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1}F\left(x^{(k)}\right) &= (\tilde{x}-\hat{a})^{\odot\alpha}+\alpha^2\hat{C}_2^{CO}(\tilde{x}-\hat{a})^{\odot(2\alpha-2)}e^{(k)^2} \\ &+ O\left(e^{(k)^3}\right). \end{aligned}$$

Usando de nuevo el Teorema 4.2.4,

$$\begin{aligned} \left[\left(x^{(k)}-\hat{a}\right)^{\odot\alpha}-\alpha\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1}F\left(x^{(k)}\right)\right]^{\odot 1/\alpha} &= \tilde{x}-\hat{a}+\alpha\hat{C}_2^{CO}(\tilde{x}-\hat{a})^{\odot(\alpha-1)}e^{(k)^2} \\ &+ O\left(e^{(k)^3}\right). \end{aligned}$$

Sea $x^{(k+1)}=e^{(k+1)}+\tilde{x}$,

$$e^{(k+1)}+\tilde{x}=\hat{a}+\tilde{x}-\hat{a}+\alpha\hat{C}_2^{CO}(\tilde{x}-\hat{a})^{\odot(\alpha-1)}e^{(k)^2}+O\left(e^{(k)^3}\right).$$

Finalmente,

$$e^{(k+1)}=\alpha\hat{C}_2^{CO}(\tilde{x}-\hat{a})^{\odot(\alpha-1)}e^{(k)^2}+O\left(e^{(k)^3}\right).$$

Y esto completa la prueba. □

Como en (3.8), y considerando (4.20), en la ecuación del error (4.29) podemos obtener análogamente,

$$\alpha\hat{C}_2^{CO}(\tilde{x}-\hat{a})^{\odot(\alpha-1)}=\hat{C}_2+\frac{1}{2}(1-\alpha)(\tilde{x}-\hat{a})^{\odot(-1)},$$

siendo $\hat{C}_q=\frac{1}{q!}\left[F'(\tilde{x})\right]^{-1}F^{(q)}(\tilde{x})$ para $q=2,3,4,\ldots$, la cual es la constante de error asintótica clásica para una función vectorial F, y F' es la matriz Jacobiana clásica. Para este caso, $q=2$.

Por otra parte, en el siguiente resultado probamos el orden de convergencia cuadrático, de forma alternativa, del método NvCO con el uso de la serie de Taylor clásica.

Corolario 4.4.1 *Sea $F: D\subseteq\mathbb{R}^n\to\mathbb{R}^n$ una función suficientemente diferenciable en un conjunto convexo abierto $D\subseteq\mathbb{R}^n$, conteniendo un cero $\tilde{x}\in\mathbb{R}^n$ de la función vectorial $F(x)$. Sea $F_{\hat{a}}^{\alpha(1)}(x)$ la matriz Jacobiana conformable de F centrada en $\hat{a}\in\mathbb{R}^n$, de orden α, para cualquier $\alpha\in(0,1]$. supongamos que $F_{\hat{a}}^{\alpha(1)}(x)$ es continua y no singular en $\tilde{x}$. Si una aproximación inicial $x^{(0)}\in\mathbb{R}^n$ es lo suficientemente cercana a $\tilde{x}$, entonces el orden de convergencia local del método de Newton-Raphson conformable vectorial (NvCO)*

$$x^{(k+1)} = \hat{a} + \left[\left(x^{(k)} - \hat{a}\right)^{\odot\alpha} - \alpha\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1} F\left(x^{(k)}\right)\right]^{\odot 1/\alpha}, \quad k = 0, 1, 2, \ldots$$

es al menos 2, y la ecuación del error es

$$e^{(k+1)} = \left(\hat{C}_2 + \frac{1}{2}(1-\alpha)(\tilde{x} - \hat{a})^{\odot(-1)}\right) {e^{(k)}}^2 + O\left({e^{(k)}}^3\right), \tag{4.30}$$

siendo $\hat{C}_q = \frac{1}{q!}\left[F'(\tilde{x})\right]^{-1} F^{(q)}(\tilde{x})$ *para* $q = 2, 3, 4, \ldots$, *tal que* $\hat{a} < x^{(k)}$, $\forall k$.

Demostración. Usando la notación dada en la Definición 2.1.3, y considerando $x^{(k)} = e^{(k)} + \tilde{x}$, el desarrollo de Taylor de $F(x)$ en el entorno de $\tilde{x}$ es

$$F\left(x^{(k)}\right) = F'(\tilde{x})\left[e^{(k)} + \hat{C}_2 {e^{(k)}}^2\right] + O\left({e^{(k)}}^3\right),$$

siendo $\hat{C}_q = \frac{1}{q!}\left[F'(\tilde{x})\right]^{-1} F^{(q)}(\tilde{x})$ para $q = 2, 3, 4, \ldots$

Considerando (4.20) y el Teorema 4.2.4, y usando de nuevo la notación dada en la Definición 2.1.3, el desarrollo de la matriz Jacobiana conformable de $F\left(x^{(k)}\right)$ en términos de sus derivadas enteras es

$$F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right) = F'(\tilde{x})\left[(\tilde{x} - \hat{a})^{\odot(1-\alpha)} I_n + \left((1-\alpha)(\tilde{x} - \hat{a})^{\odot(-\alpha)} I_n + 2(\tilde{x} - \hat{a})^{\odot(1-\alpha)}\hat{C}_2\right) e^{(k)}\right] + O\left({e^{(k)}}^2\right).$$

Podemos escribir el desarrollo de $\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1}$ como

$$\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1} = \left[(\tilde{x} - \hat{a})^{\odot(\alpha-1)} I_n + \bar{X}_2 e^{(k)}\right]\left[F'(\tilde{x})\right]^{-1} + O\left({e^{(k)}}^2\right),$$

donde $\bar{X}_2$ es una variable desconocida tal que $\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1} F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right) = I_n$, y así,

$$\left((1-\alpha)(\tilde{x} - \hat{a})^{\odot(-1)} I_n + 2\hat{C}_2 + (\tilde{x} - \hat{a})^{\odot(1-\alpha)}\bar{X}_2\right) e^{(k)} = \hat{0}.$$

Resolviendo para $\bar{X}_2$,

$$\bar{X}_2 = (\alpha - 1)(\tilde{x} - \hat{a})^{\odot(\alpha-2)} I_n - 2(\tilde{x} - \hat{a})^{\odot(\alpha-1)}\hat{C}_2.$$

Entonces,

$$\begin{aligned}\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1} &= \left[(\tilde{x} - \hat{a})^{\odot(\alpha-1)} I_n + \left((\alpha-1)(\tilde{x} - \hat{a})^{\odot(\alpha-2)} I_n - 2(\tilde{x} - \hat{a})^{\odot(\alpha-1)}\hat{C}_2\right) e^{(k)}\right]\left[F'(\tilde{x})\right]^{-1} \\ &+ O\left({e^{(k)}}^2\right).\end{aligned}$$

Por lo tanto,

$$\begin{aligned} -\alpha\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1}F\left(x^{(k)}\right) &= -\alpha\left((\tilde{x}-\hat{a})^{\odot(\alpha-1)}\right)e^{(k)} \\ &+ \alpha\left((\tilde{x}-\hat{a})^{\odot(\alpha-1)}\hat{C}_2-(\alpha-1)(\tilde{x}-\hat{a})^{\odot(\alpha-2)}\right){e^{(k)}}^2+O\left({e^{(k)}}^3\right). \end{aligned}$$

Entonces,

$$\begin{aligned} \left(x^{(k)}-\hat{a}\right)^{\odot\alpha}-\alpha\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1}F\left(x^{(k)}\right) &= (\tilde{x}-\hat{a})^{\odot\alpha} \\ &+ \left(\alpha(\tilde{x}-\hat{a})^{\odot(\alpha-1)}\hat{C}_2-\frac{1}{2}\alpha(\alpha-1)(\tilde{x}-\hat{a})^{\odot(\alpha-2)}\right){e^{(k)}}^2 \\ &+ O\left({e^{(k)}}^3\right). \end{aligned}$$

Usando de nuevo el Teorema 4.2.4,

$$\left[\left(x^{(k)}-\hat{a}\right)^{\odot\alpha}-\alpha\left[F_{\hat{a}}^{\alpha(1)}\left(x^{(k)}\right)\right]^{-1}F\left(x^{(k)}\right)\right]^{\odot1/\alpha}=\tilde{x}-\hat{a}+\left(\hat{C}_2+\frac{1}{2}(1-\alpha)(\tilde{x}-\hat{a})^{\odot(-1)}\right){e^{(k)}}^2+O\left({e^{(k)}}^3\right).$$

Sea $x^{(k+1)}=e^{(k+1)}+\tilde{x}$,

$$e^{(k+1)}+\tilde{x}=\hat{a}+\tilde{x}-\hat{a}+\left(\hat{C}_2+\frac{1}{2}(1-\alpha)(\tilde{x}-\hat{a})^{\odot(-1)}\right){e^{(k)}}^2+O\left({e^{(k)}}^3\right).$$

Finalmente,

$$e^{(k+1)}=\left(\hat{C}_2+\frac{1}{2}(1-\alpha)(\tilde{x}-\hat{a})^{\odot(-1)}\right){e^{(k)}}^2+O\left({e^{(k)}}^3\right).$$

Esto completa la prueba. □

Observación 4.4.1 *Se confirma que las ecuaciones del error (4.29) y (4.30) son equivalentes.*

En la siguiente sección, realizamos diversas pruebas numéricas con algunos sistemas de ecuaciones no lineales. En todas las pruebas se hace una comparación con el método clásico de Newton-Raphson que se obtiene en NvCO cuando $\alpha=1$. También se analiza la dependencia respecto a las estimaciones iniciales con el uso de planos de convergencia.

4.5. Pruebas numéricas

Los siguientes resultados fueron obtenidos con el uso de Matlab R2020a con aritmética de precisión doble, $\|F(x^{(k+1)})\|<10^{-8}$ o $\|x^{(k+1)}-x^{(k)}\|<10^{-8}$ como criterios de parada, y un máximo de 500 iteraciones. Para cada prueba usamos $\hat{a}=(a_1,\ldots,a_n)=(-10,\ldots,-10)$. También usamos el Orden de Convergencia Computacional Aproximado (ACOC)

$$\hat{\rho}=ACOC=\frac{\ln(\|x^{(k+1)}-x^{(k)}\|/\|x^{(k)}-x^{(k-1)}\|)}{\ln(\|x^{(k)}-x^{(k-1)}\|/\|x^{(k-1)}-x^{(k-2)}\|)},\quad k=2,3,4,\ldots,$$

definido en [17], para comprobar que el orden de convergencia teórico se obtiene en la práctica. En cada tabla hemos usado la misma estimación inicial, y $\alpha \in (0, 1]$ (ver ficheros MATLAB en Anexos A.19 a A.21).

A partir de los datos de cada tabla, se proporcionan dos figuras (por ejemplo, las Figuras 4.1 y 4.2) con curvas de error para visualizar el error cometido ($\|x^{(k+1)} - x^{(k)}\|$) frente al número de iteraciones para diferentes valores de α (ver fichero MATLAB en Anexo A.22). Siguiendo con este mismo ejemplo, la Figura 4.1 muestra las curvas de error para todos los valores disponibles de α, mientras que la Figura 4.2 muestra las curvas de error para algunos valores de α con el objetivo de distinguir cada curva de otra; esto se mantiene para el resto de pares de figuras. En el caso en que se muestran las curvas de error para algunos valores de α, las curvas elegidas corresponden a valores de α con menos iteraciones, o a una elección arbitraria cuando el número de iteraciones sea el mismo. En cada caso, la curva correspondiente a $\alpha = 1$ es siempre elegida cuando sea posible con el fin de visualizar ambos métodos, el clásico (cuando $\alpha = 1$), y NvCO.

Nuestra primera función vectorial de prueba es $F_1(x, y) = (x^2 - 2x - y + 0.5, x^2 + 4y^2 - 4)^T$ con raíces reales y complejas $\tilde{x}_1 \approx (-0.2222, 0.9938)^T$, $\tilde{x}_2 \approx (1.9007, 0.3112)^T$ y $\tilde{x}_3 \approx (1.1608 - 0.6545i, -0.9025 - 0.2104i)^T$. La matriz Jacobiana conformable de $F_1(x, y)$ es

$$F_{\hat{a}}^{\alpha(1)}{}_1(x, y) = \begin{pmatrix} (x - a_1)^{1-\alpha}(2x - 2) & (y - a_2)^{1-\alpha}(-1) \\ (x - a_1)^{1-\alpha}(2x) & (y - a_2)^{1-\alpha}(8y) \end{pmatrix},$$

siendo $a = (a_1, a_2) = (-10, -10)$.

Método NvCO					
α	$\tilde{x}$	$\|F_1(x^{(k+1)})\|$	$\|x^{(k+1)} - x^{(k)}\|$	iter	$\hat{\rho}$
1	-	-	-	> 500	-
0.9	$\tilde{x}_3$	5.40×10^{-11}	5.86×10^{-6}	54	2.00
0.8	$\tilde{x}_3$	9.77×10^{-9}	7.87×10^{-5}	86	2.00
0.7	$\tilde{x}_3$	2.27×10^{-14}	4.75×10^{-8}	36	1.98
0.6	$\tilde{x}_2$	2.95×10^{-10}	1.16×10^{-5}	23	2.05
0.5	$\tilde{x}_2$	4.09×10^{-15}	2.13×10^{-8}	200	1.98
0.4	$\tilde{x}_2$	7.12×10^{-15}	4.38×10^{-8}	114	2.04
0.3	$\tilde{x}_2$	4.94×10^{-10}	1.79×10^{-5}	35	2.03
0.2	$\tilde{x}_2$	1.16×10^{-14}	6.76×10^{-8}	21	1.98
0.1	$\tilde{x}_2$	2.16×10^{-10}	1.08×10^{-5}	39	2.06

Tabla 4.1: Resultados para $F_1(x, y) = \hat{0}$ con estimación inicial $x^{(0)} = (-2, -1.5)^T$

Método NvCO					
α	$\tilde{x}$	$\|F_1(x^{(k+1)})\|$	$\|x^{(k+1)} - x^{(k)}\|$	iter	$\hat{\rho}$
1	$\tilde{x}_1$	8.31×10^{-11}	7.29×10^{-6}	5	2.00
0.9	$\tilde{x}_1$	5.94×10^{-11}	6.15×10^{-6}	5	2.00
0.8	$\tilde{x}_1$	4.21×10^{-11}	5.17×10^{-6}	5	2.00
0.7	$\tilde{x}_1$	2.97×10^{-11}	4.33×10^{-6}	5	2.00
0.6	$\tilde{x}_1$	2.09×10^{-11}	3.62×10^{-6}	5	2.00
0.5	$\tilde{x}_1$	1.45×10^{-11}	3.01×10^{-6}	5	2.00
0.4	$\tilde{x}_1$	1.01×10^{-11}	2.49×10^{-6}	5	2.00
0.3	$\tilde{x}_1$	6.97×10^{-12}	2.06×10^{-6}	5	2.00
0.2	$\tilde{x}_1$	4.80×10^{-12}	1.69×10^{-6}	5	2.00
0.1	$\tilde{x}_1$	3.30×10^{-12}	1.39×10^{-6}	5	2.00

Tabla 4.2: Resultados para $F_1(x, y) = \hat{0}$ con estimación inicial $x^{(0)} = (-2, 1.5)^T$

En la Tabla 4.1 observamos para $F_1(x, y)$ que el método clásico de Newton-Raphson (cuando $\alpha = 1$) no encuentra solución en 500 iteraciones, mientras que NvCO converge. También podemos observar que el orden de convergencia computacional ($\hat{\rho}$) puede ser incluso ligeramente mayor que 2 cuando $\alpha \neq 1$. Debemos comentar que se ha obtenido una raíz compleja ($\tilde{x}_3$) con estimación inicial real para diferentes valores de α

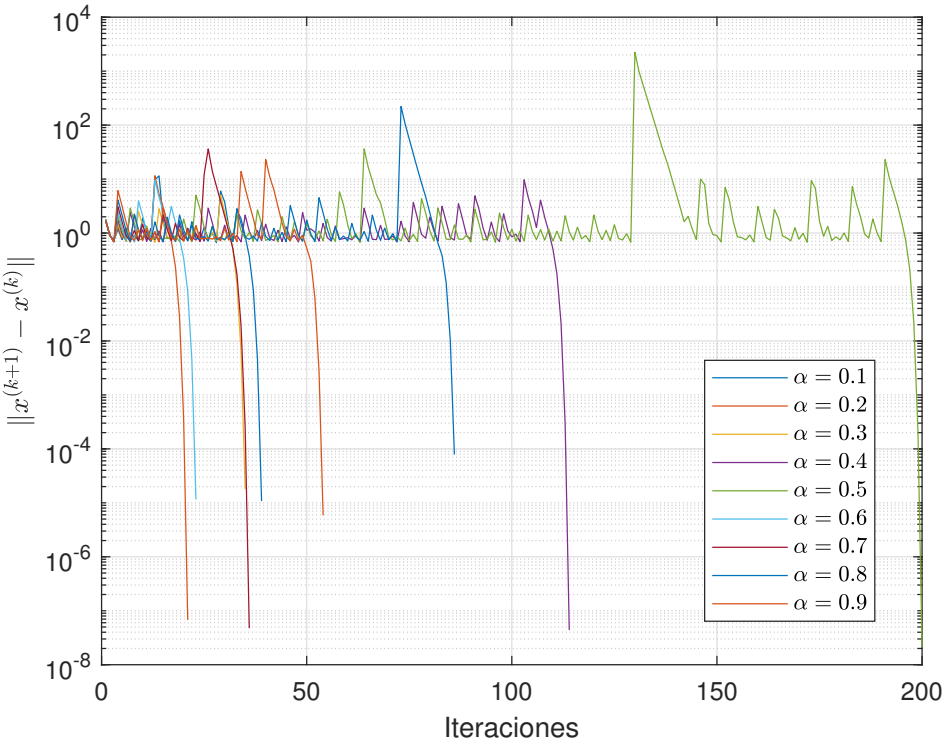


Figura 4.1: Curvas de error de $F_1(x, y)$ para todos los valores de α en la Tabla 4.1

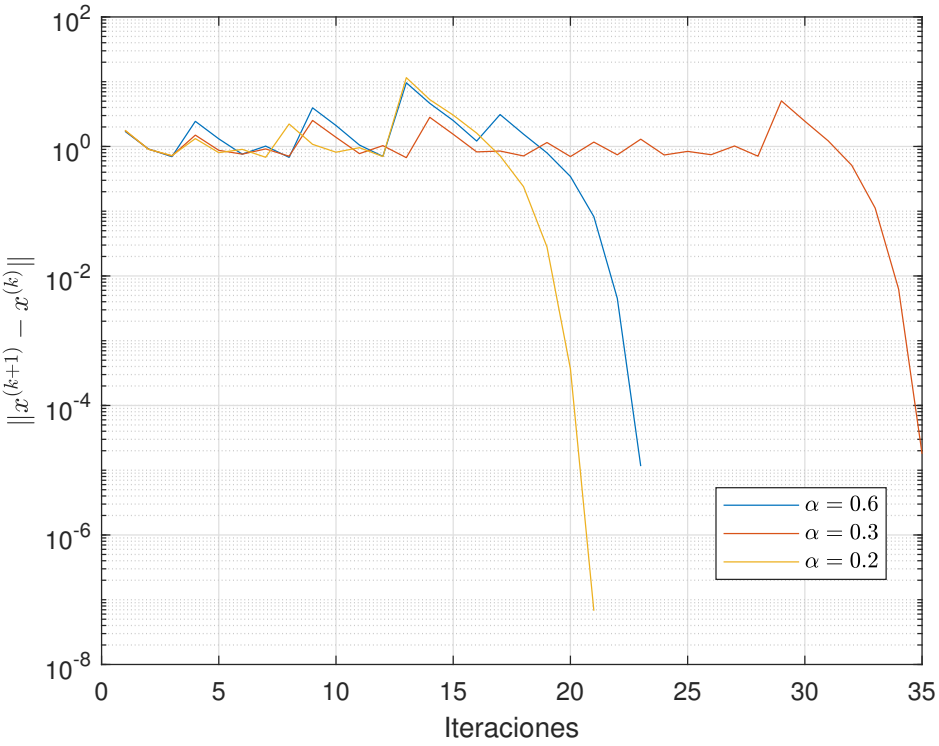


Figura 4.2: Curvas de error de $F_1(x, y)$ para algunos valores de α en la Tabla 4.1

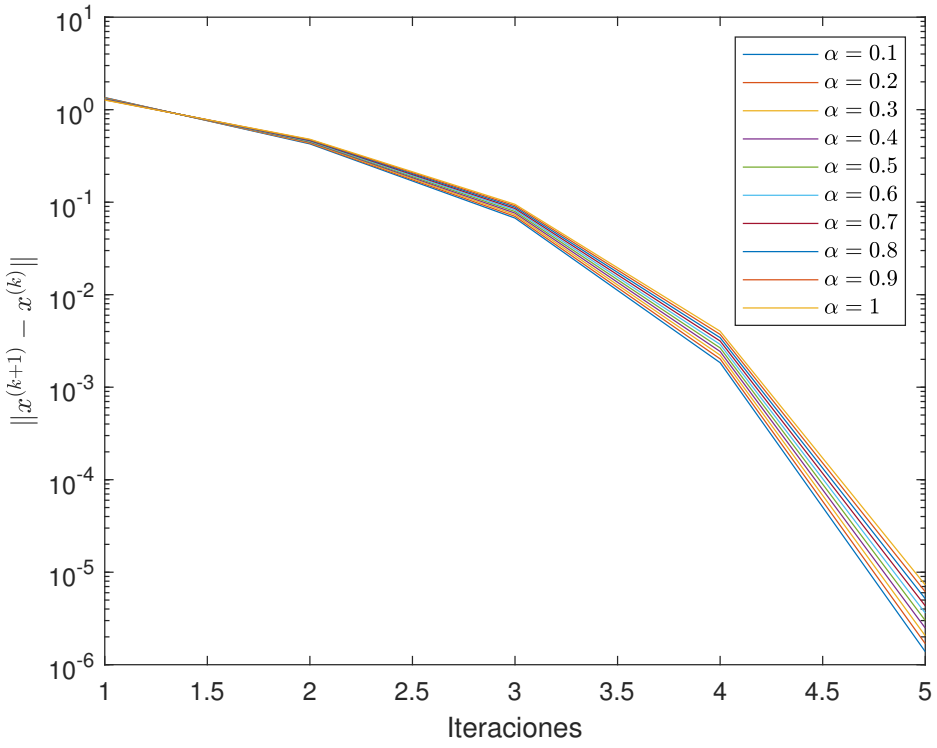


Figura 4.3: Curvas de error de $F_1(x, y)$ para todos los valores de α en la Tabla 4.2

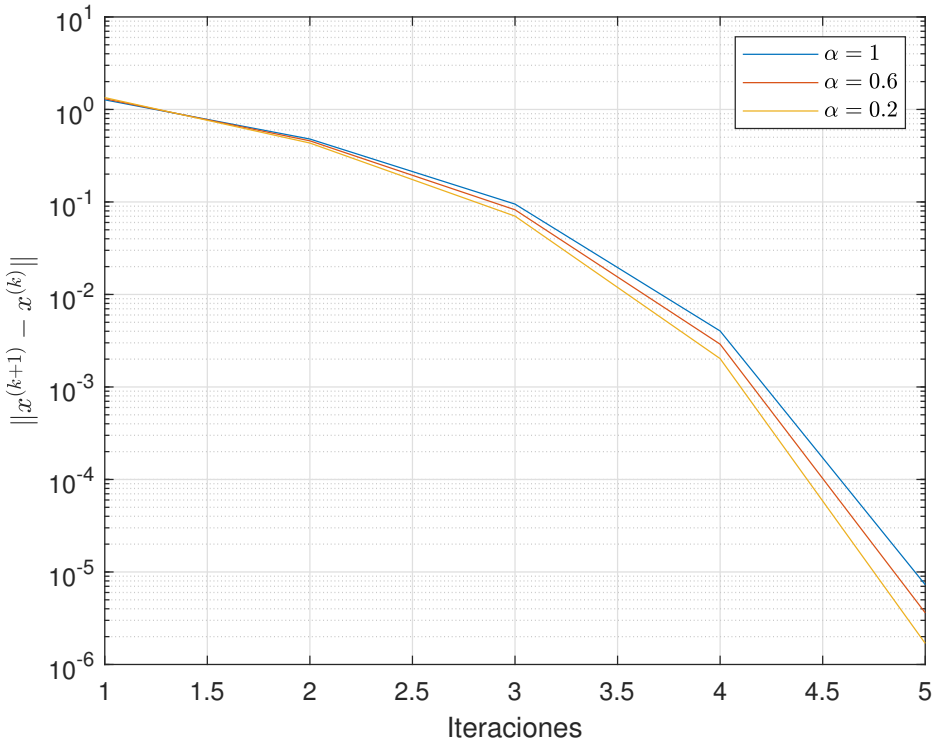


Figura 4.4: Curvas de error de $F_1(x, y)$ para algunos valores de α en la Tabla 4.2

cuando se usa NvCO, y que es posible obtener diferentes soluciones al variar el valor de α. En las Figuras 4.1 y 4.2, la curva de error para el procedimiento clásico de Newton-Raphson no se proporciona porque no se encuentra solución en este caso, mientras que las curvas de error obtenidas detienen su comportamiento errático en las últimas iteraciones.

En la Tabla 4.2 podemos ver para $F_1(x, y)$, con una estimación inicial diferente, que el esquema clásico de Newton-Raphson y NvCO tienen un comportamiento similar en cuanto a la cantidad de iteraciones y al orden de convergencia computacional. De nuevo, se obtiene orden cuadrático al usar NvCO para todo $\alpha \in (0, 1]$. En las Figuras 4.3 y 4.4 no se observa un comportamiento errático, ya que los errores disminuyen con cada iteración.

La segunda función vectorial de prueba es $F_2(x, y) = (x^2 + y^2 - 1, x^2 - y^2 - 1/2)^T$ con raíces reales $\tilde{x}_1 = \left(\sqrt{3}/2, 1/2\right)^T$, $\tilde{x}_2 = \left(-\sqrt{3}/2, 1/2\right)^T$, $\tilde{x}_3 = \left(\sqrt{3}/2, -1/2\right)^T$ y $\tilde{x}_4 = \left(-\sqrt{3}/2, -1/2\right)^T$. La matriz Jacobiana conformable de $F_2(x, y)$ es

$$F_{\hat{a}}^{\alpha(1)}{}_2(x, y) = \begin{pmatrix} (x - a_1)^{1-\alpha}(2x) & (y - a_2)^{1-\alpha}(2y) \\ (x - a_1)^{1-\alpha}(2x) & (y - a_2)^{1-\alpha}(-2y) \end{pmatrix},$$

siendo $a = (a_1, a_2) = (-10, -10)$.

Método NvCO					
α	$\tilde{x}$	$\|F_2(x^{(k+1)})\|$	$\|x^{(k+1)} - x^{(k)}\|$	iter	$\hat{\rho}$
1	$\tilde{x}_3$	7.60×10^{-12}	2.32×10^{-6}	6	2.00
0.9	$\tilde{x}_3$	5.21×10^{-12}	1.92×10^{-6}	6	2.00
0.8	$\tilde{x}_3$	3.54×10^{-12}	1.59×10^{-6}	6	2.00
0.7	$\tilde{x}_3$	2.38×10^{-12}	1.31×10^{-6}	6	2.00
0.6	$\tilde{x}_3$	1.58×10^{-12}	1.07×10^{-6}	6	2.00
0.5	$\tilde{x}_3$	1.04×10^{-12}	8.69×10^{-7}	6	2.00
0.4	$\tilde{x}_3$	6.73×10^{-13}	7.03×10^{-7}	6	2.00
0.3	$\tilde{x}_3$	4.35×10^{-13}	5.65×10^{-7}	6	2.00
0.2	$\tilde{x}_3$	2.71×10^{-13}	4.51×10^{-7}	6	2.00
0.1	$\tilde{x}_3$	1.82×10^{-13}	3.58×10^{-7}	6	2.00

Tabla 4.3: Resultados para $F_2(x, y) = \hat{0}$ con estimación inicial $x^{(0)} = (2, -2.5)^T$

Método NvCO					
α	$\tilde{x}$	$\|F_2(x^{(k+1)})\|$	$\|x^{(k+1)} - x^{(k)}\|$	iter	$\hat{\rho}$
1	$\tilde{x}_1$	7.60×10^{-12}	2.32×10^{-6}	6	2.00
0.9	$\tilde{x}_1$	9.98×10^{-12}	2.65×10^{-6}	6	2.00
0.8	$\tilde{x}_1$	1.30×10^{-11}	3.02×10^{-6}	6	2.00
0.7	$\tilde{x}_1$	1.70×10^{-11}	3.44×10^{-6}	6	2.00
0.6	$\tilde{x}_1$	2.20×10^{-11}	3.91×10^{-6}	6	2.00
0.5	$\tilde{x}_1$	2.84×10^{-11}	4.43×10^{-6}	6	2.00
0.4	$\tilde{x}_1$	3.65×10^{-11}	5.01×10^{-6}	6	2.00
0.3	$\tilde{x}_1$	4.67×10^{-11}	5.66×10^{-6}	6	2.00
0.2	$\tilde{x}_1$	5.96×10^{-11}	6.37×10^{-6}	6	2.00
0.1	$\tilde{x}_1$	7.57×10^{-11}	7.16×10^{-6}	6	2.00

Tabla 4.4: Resultados para $F_2(x, y) = \hat{0}$ con estimación inicial $x^{(0)} = (2, 2.5)^T$

Podemos ver en las Tablas 4.3 y 4.4 para $F_2(x, y)$ que el método clásico de Newton-Raphson y NvCO tienen un comportamiento similar en cuanto a la cantidad de iteraciones y al orden de convergencia computacional. Las Figuras 4.5, 4.6, 4.7 y 4.8 muestran que no se observa comportamiento errático, porque los errores disminuyen con las iteraciones.

Nuestra tercera función vectorial de prueba es $F_3(x, y) = (x^2 - x - y^2 - 1, -\sin x + y)^T$ con raíces reales

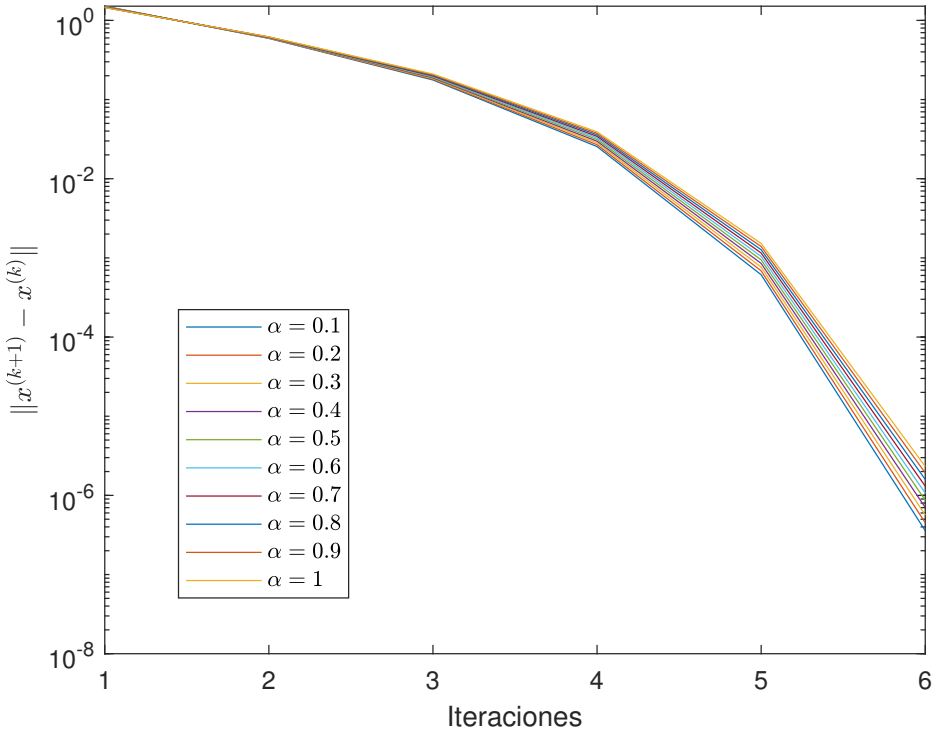


Figura 4.5: Curvas de error de $F_2(x, y)$ para todos los valores de α en la Tabla 4.3

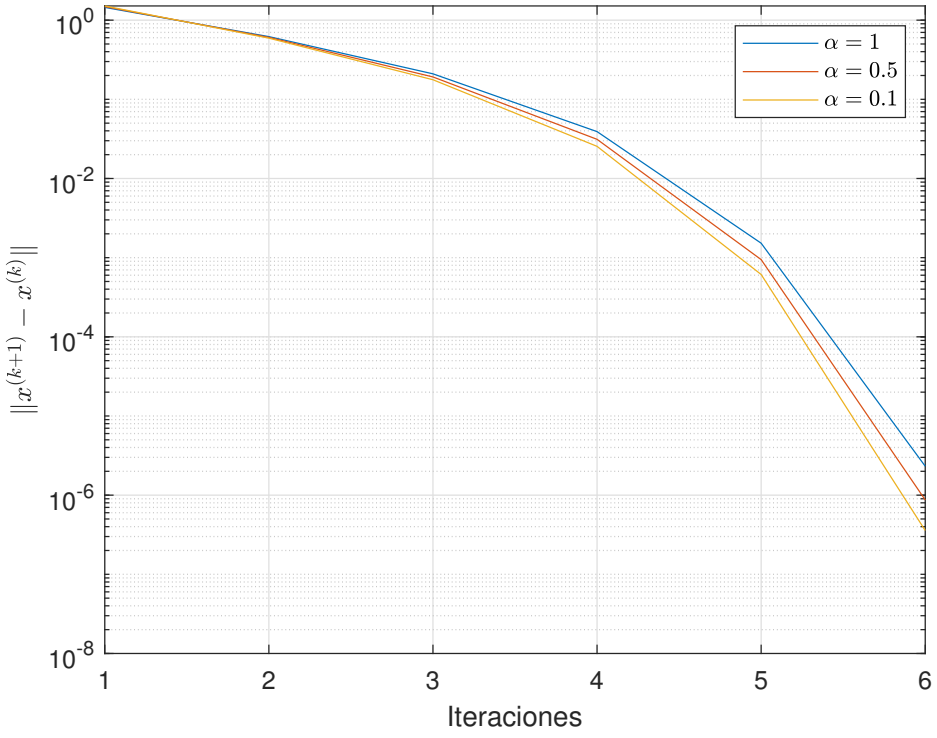


Figura 4.6: Curvas de error de $F_2(x, y)$ para algunos valores de α en la Tabla 4.3

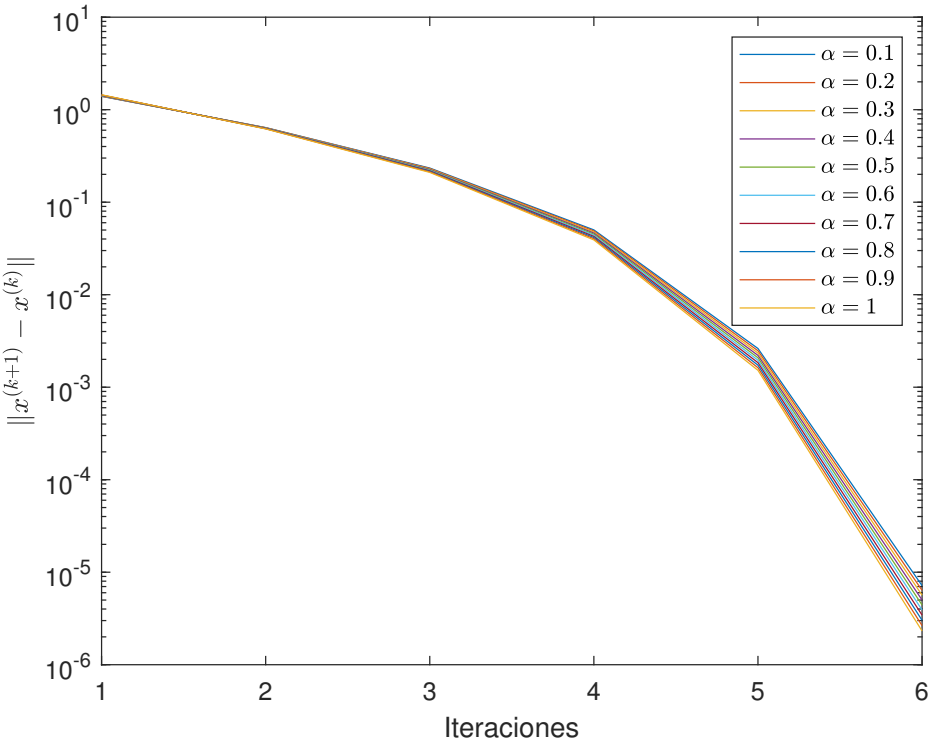


Figura 4.7: Curvas de error de $F_2(x, y)$ para todos los valores de α en la Tabla 4.4

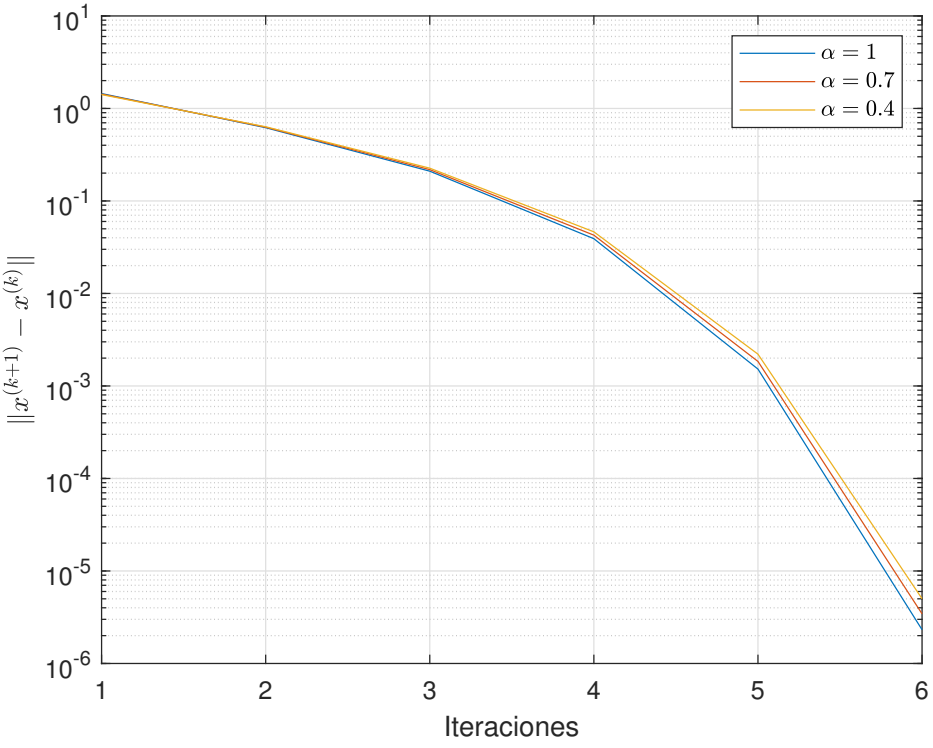


Figura 4.8: Curvas de error de $F_2(x, y)$ para algunos valores de α en la Tabla 4.4

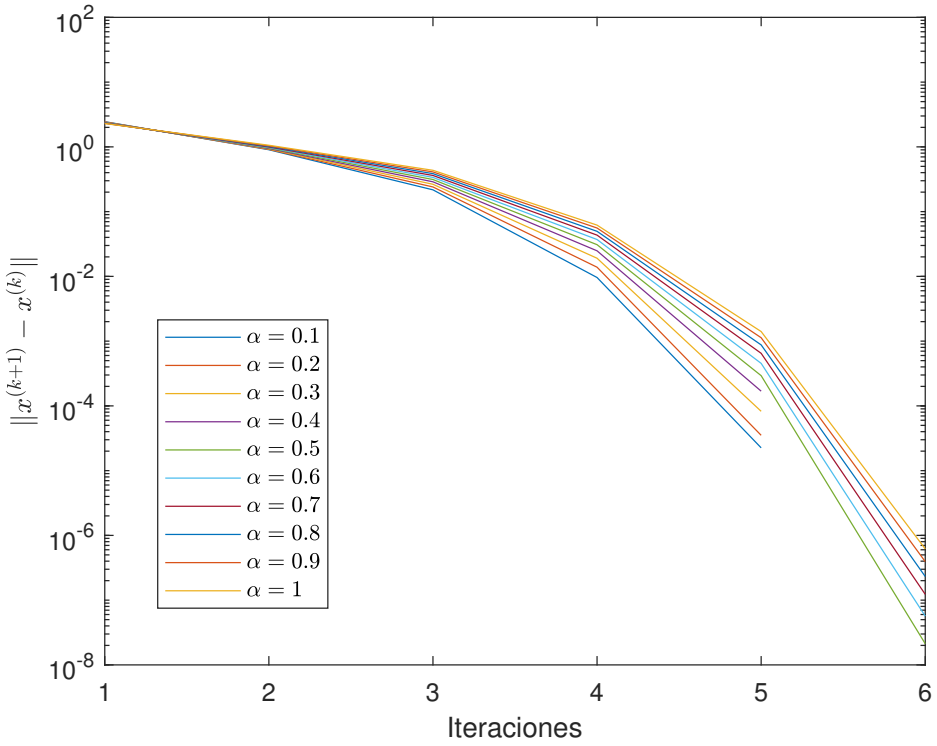


Figura 4.9: Curvas de error de $F_3(x, y)$ para todos los valores de α en la Tabla 4.5

$\tilde{x}_1 \approx (-0.8453, -0.7481)^T$ y $\tilde{x}_2 \approx (1.9529, 0.9279)^T$. La matriz Jacobiana conformable de $F_3(x, y)$ es

$$F_{\hat{a}}^{\alpha(1)}{}_3(x, y) = \begin{pmatrix} (x-a_1)^{1-\alpha}(2x-1) & (y-a_2)^{1-\alpha}(-2y) \\ (x-a_1)^{1-\alpha}(-\cos x) & (y-a_2)^{1-\alpha}(1) \end{pmatrix},$$

siendo $a = (a_1, a_2) = (-10, -10)$.

Método NvCO					
α	$\tilde{x}$	$\|F_3(x^{(k+1)})\|$	$\|x^{(k+1)} - x^{(k)}\|$	iter	$\hat{\rho}$
1	$\tilde{x}_2$	1.03×10^{-13}	6.18×10^{-7}	6	2.04
0.9	$\tilde{x}_2$	4.51×10^{-14}	3.94×10^{-7}	6	2.04
0.8	$\tilde{x}_2$	1.29×10^{-14}	2.32×10^{-7}	6	2.04
0.7	$\tilde{x}_2$	7.62×10^{-15}	1.23×10^{-7}	6	2.04
0.6	$\tilde{x}_2$	3.79×10^{-15}	5.67×10^{-8}	6	2.04
0.5	$\tilde{x}_2$	4.67×10^{-15}	2.12×10^{-8}	6	2.05
0.4	$\tilde{x}_2$	7.53×10^{-9}	1.69×10^{-4}	5	2.03
0.3	$\tilde{x}_2$	3.93×10^{-9}	8.25×10^{-5}	5	2.07
0.2	$\tilde{x}_2$	1.31×10^{-9}	3.51×10^{-5}	5	2.09
0.1	$\tilde{x}_2$	1.75×10^{-10}	2.25×10^{-5}	5	1.94

Tabla 4.5: Resultados para $F_3(x, y) = \hat{0}$ con estimación inicial $x^{(0)} = (2.5, -0.5)^T$

Podemos observar en la Tabla 4.5 para $F_3(x, y)$ que NvCO requiere menos iteraciones que el método clásico de Newton-Raphson para $\alpha < 0.5$. También observamos que el orden de convergencia computacional puede ser ligeramente superior a 2 para pequeños valores de α. En las Figuras 4.9 y 4.10 los errores decrecen en cada iteración.

Cambiando la estimación inicial, en la Tabla 4.6 podemos ver para $F_3(x, y)$ que NvCO y el método clásico de

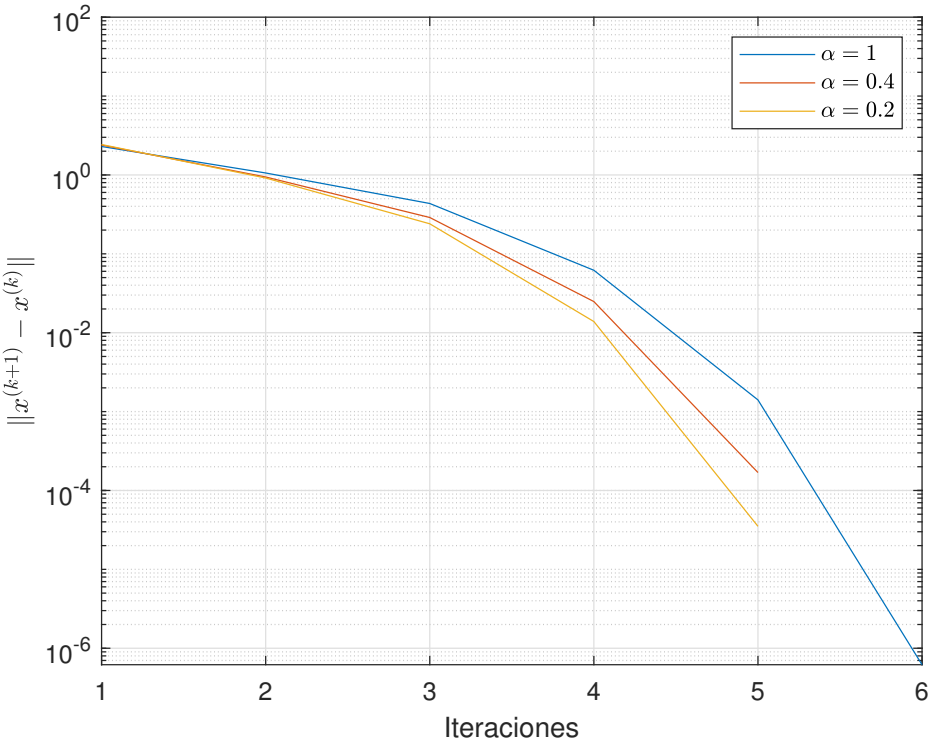


Figura 4.10: Curvas de error de $F_3(x, y)$ para algunos valores de α en la Tabla 4.5

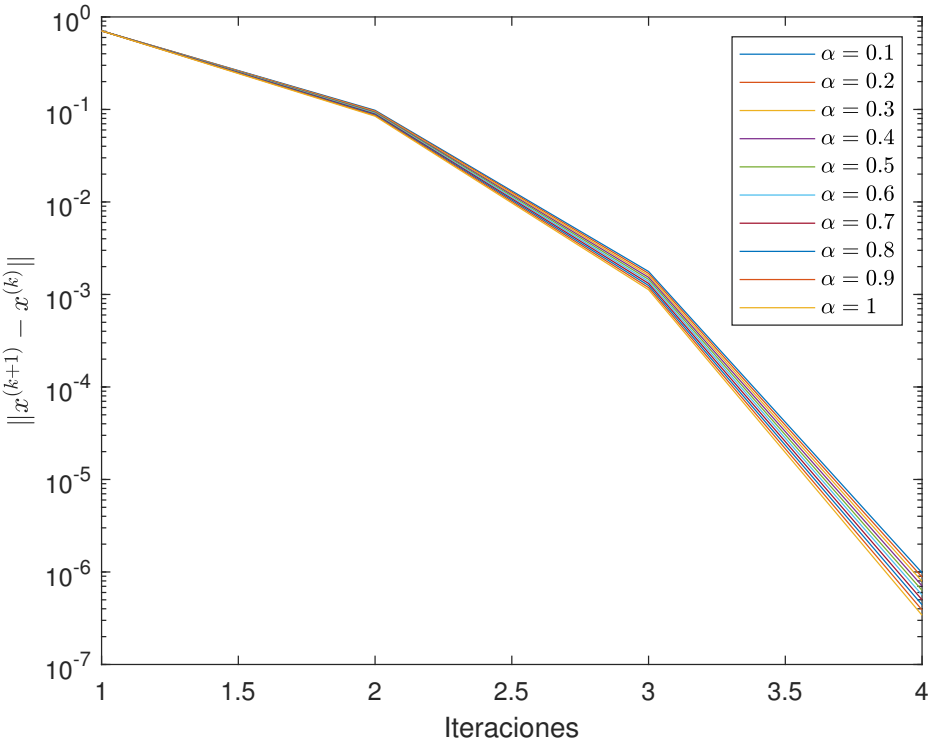


Figura 4.11: Curvas de error de $F_3(x, y)$ para todos los valores de α en la Tabla 4.6

Método NvCO					
α	$\tilde{x}$	$\|F_3(x^{(k+1)})\|$	$\|x^{(k+1)} - x^{(k)}\|$	iter	$\hat{\rho}$
1	$\tilde{x}_2$	8.80×10^{-14}	3.40×10^{-7}	4	1.88
0.9	$\tilde{x}_2$	1.26×10^{-13}	3.89×10^{-7}	4	1.87
0.8	$\tilde{x}_2$	1.58×10^{-13}	4.42×10^{-7}	4	1.87
0.7	$\tilde{x}_2$	1.94×10^{-13}	5.00×10^{-7}	4	1.87
0.6	$\tilde{x}_2$	2.47×10^{-13}	5.63×10^{-7}	4	1.87
0.5	$\tilde{x}_2$	3.07×10^{-13}	6.33×10^{-7}	4	1.87
0.4	$\tilde{x}_2$	3.85×10^{-13}	7.09×10^{-7}	4	1.87
0.3	$\tilde{x}_2$	4.81×10^{-13}	7.92×10^{-7}	4	1.87
0.2	$\tilde{x}_2$	5.81×10^{-13}	8.82×10^{-7}	4	1.87
0.1	$\tilde{x}_2$	7.15×10^{-13}	9.80×10^{-7}	4	1.87

Tabla 4.6: Resultados para $F_3(x, y) = \hat{0}$ con estimación inicial $x^{(0)} = (2.5, 0.5)^T$

Newton-Raphson requieren la misma cantidad de iteraciones, y que el orden de convergencia computacional es alrededor de 2 para todo α. De nuevo, los errores decrecen en cada iteración en las Figuras 4.11 y 4.12.

La cuarta función vectorial de prueba es $F_4(x, y) = (x^2 + y^2 - 4, e^x + y - 1)^T$ con raíces reales $\tilde{x}_1 \approx (-1.8163, 0.8374)^T$ y $\tilde{x}_2 \approx (1.0042, -1.7296)^T$. La matriz Jacobiana conformable de $F_4(x, y)$ es

$$F_{\tilde{a}}^{\alpha(1)}{}_4(x, y) = \begin{pmatrix} (x - a_1)^{1-\alpha}(2x) & (y - a_2)^{1-\alpha}(2y) \\ (x - a_1)^{1-\alpha}(e^x) & (y - a_2)^{1-\alpha}(1) \end{pmatrix},$$

siendo $a = (a_1, a_2) = (-10, -10)$.

Método NvCO					
α	$\tilde{x}$	$\|F_4(x^{(k+1)})\|$	$\|x^{(k+1)} - x^{(k)}\|$	iter	$\hat{\rho}$
1	$\tilde{x}_1$	2.32×10^{-15}	5.88×10^{-9}	7	2.00
0.9	$\tilde{x}_1$	6.04×10^{-9}	7.83×10^{-5}	6	1.99
0.8	$\tilde{x}_1$	1.57×10^{-9}	4.03×10^{-5}	6	1.99
0.7	$\tilde{x}_1$	4.29×10^{-10}	2.12×10^{-5}	6	1.99
0.6	$\tilde{x}_1$	1.29×10^{-10}	1.17×10^{-5}	6	1.99
0.5	$\tilde{x}_1$	4.62×10^{-11}	7.07×10^{-6}	6	1.99
0.4	$\tilde{x}_1$	2.18×10^{-11}	4.90×10^{-6}	6	1.99
0.3	$\tilde{x}_1$	1.52×10^{-11}	4.12×10^{-6}	6	1.99
0.2	$\tilde{x}_1$	1.72×10^{-11}	4.41×10^{-6}	6	1.99
0.1	$\tilde{x}_1$	3.30×10^{-11}	6.15×10^{-6}	6	1.98

Tabla 4.7: Resultados para $F_4(x, y) = \hat{0}$ con estimación inicial $x^{(0)} = (-2.5, -3.5)^T$

Vemos en la Tabla 4.7 para $F_4(x, y)$ que de nuevo, NvCO requiere menos iteraciones que el esquema clásico de Newton-Raphson para todos los valores de α. También podemos ver que el orden de convergencia computacional es alrededor de 2. Vemos que en las Figuras 4.13 y 4.14 los errores disminuyen con las iteraciones.

En la Tabla 4.8 observamos para $F_4(x, y)$ que NvCO y el procedimiento clásico de Newton-Raphson requieren la misma cantidad de iteraciones, y que el orden de convergencia computacional es alrededor de 2. De nuevo, los errores disminuyen con las iteraciones en las Figuras 4.15 y 4.16.

Nuestra quinta función vectorial de prueba es $F_5(x) = (f_1(x), \ldots, f_{15}(x))^T$, siendo $x = (x_1, \ldots, x_{15})^T$ y $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, 2, \ldots, 14, 15$, tal que

$$\begin{aligned} f_i(x) &= x_i x_{i+1} - 1, \quad i = 1, 2, \ldots, 13, 14 \\ f_{15}(x) &= x_{15} x_1 - 1, \end{aligned}$$

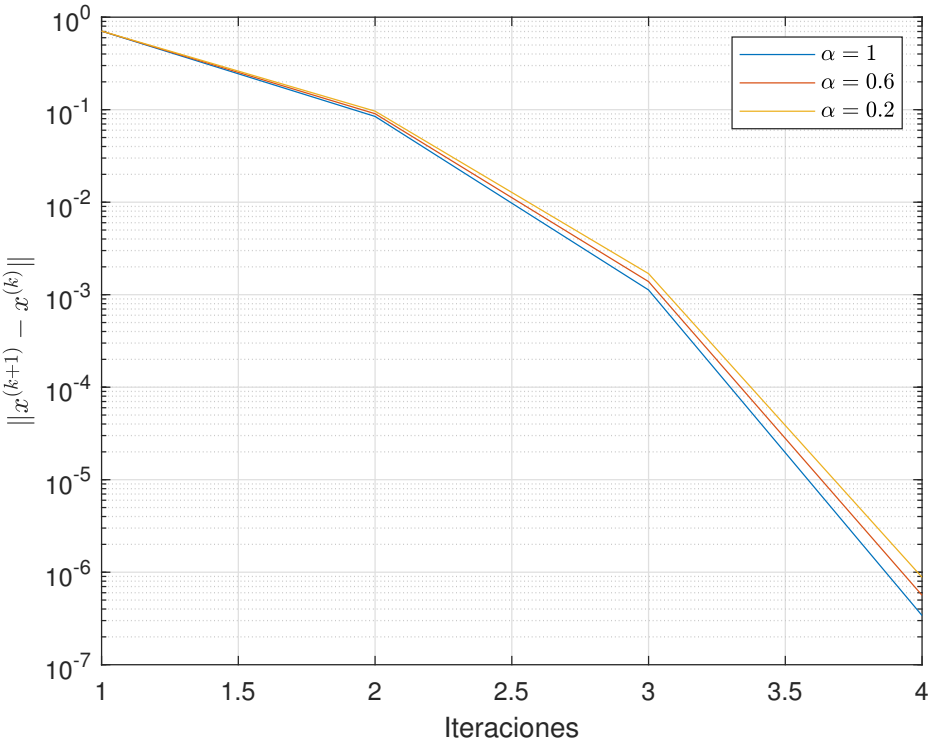


Figura 4.12: Curvas de error de $F_3(x, y)$ para algunos valores de α en la Tabla 4.6

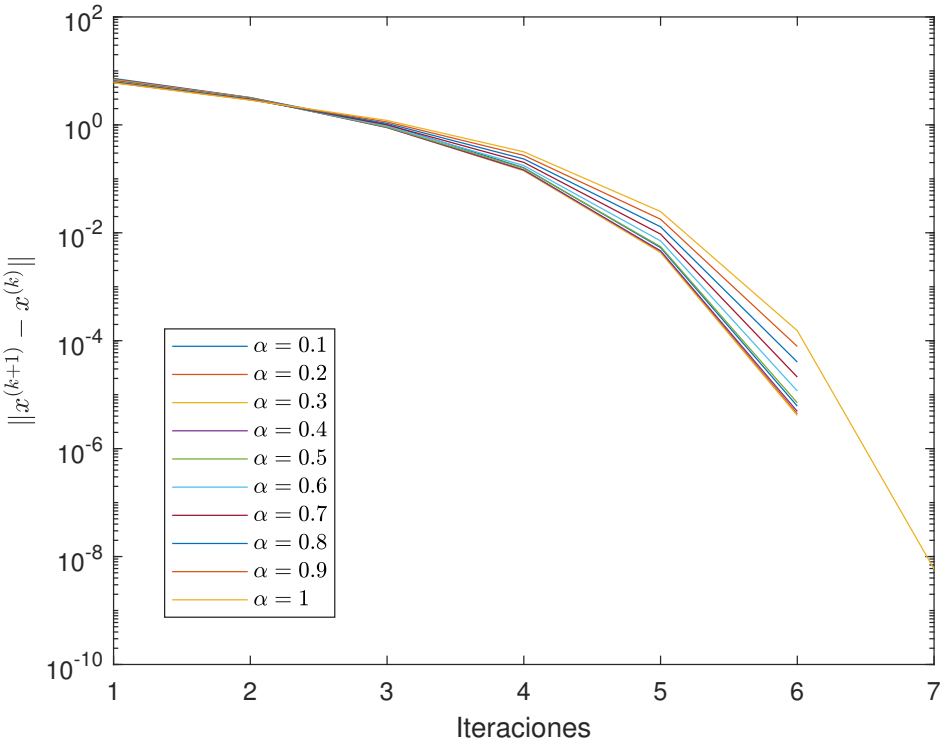


Figura 4.13: Curvas de error de $F_4(x, y)$ para todos los valores de α en la Tabla 4.7

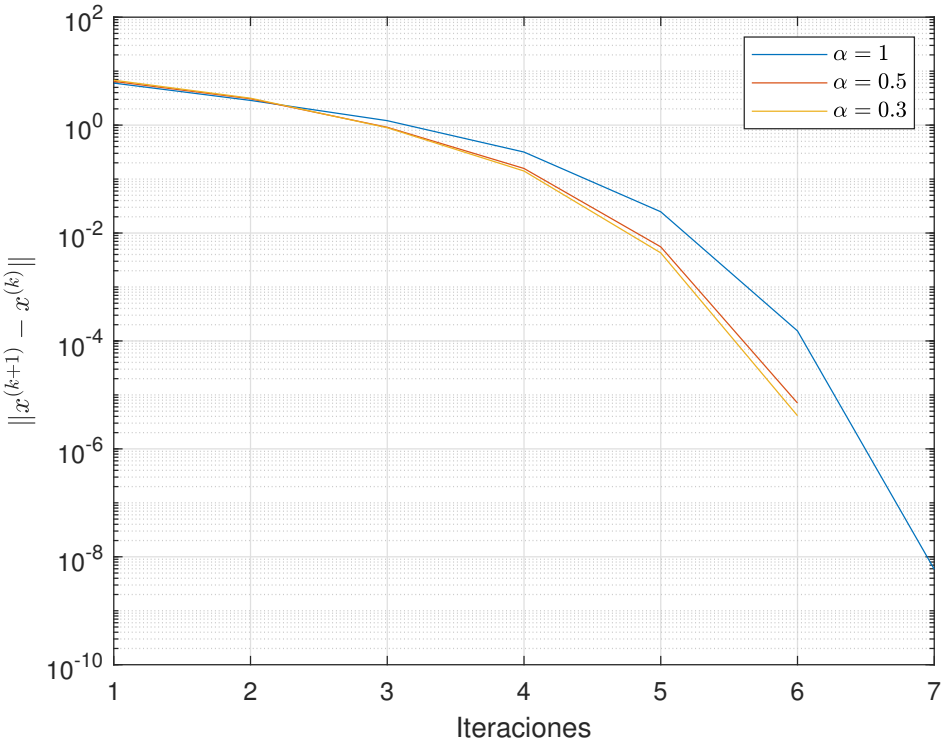


Figura 4.14: Curvas de error de $F_4(x, y)$ para algunos valores de α en la Tabla 4.7

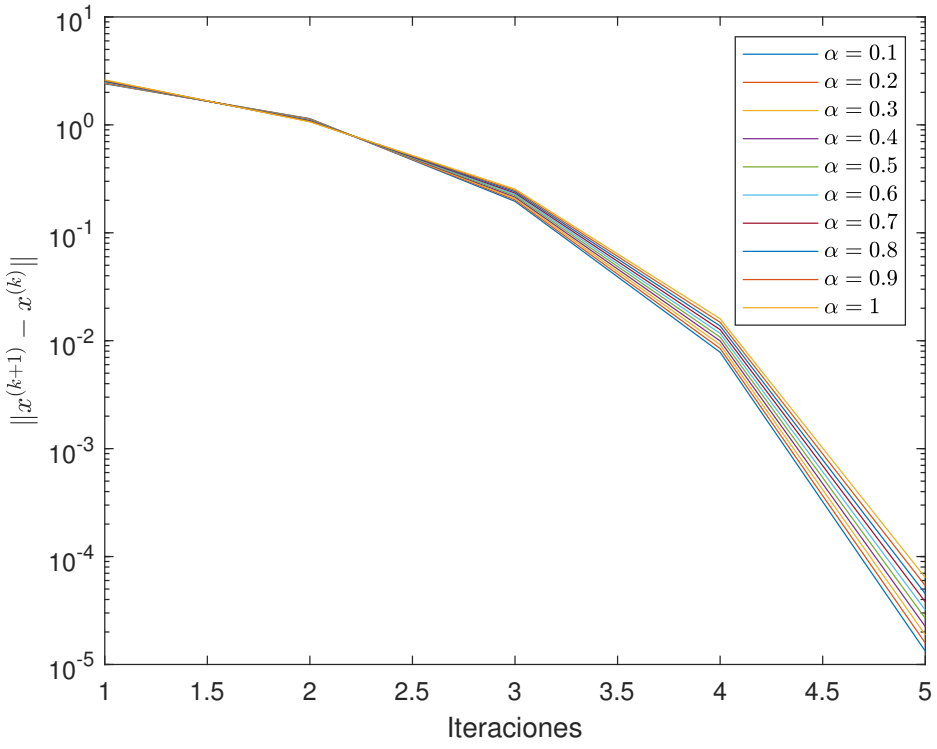


Figura 4.15: Curvas de error de $F_4(x, y)$ para todos los valores de α en la Tabla 4.8

Método NvCO					
α	$\tilde{x}$	$\|F_4(x^{(k+1)})\|$	$\|x^{(k+1)} - x^{(k)}\|$	iter	$\hat{\rho}$
1	$\tilde{x}_1$	4.09×10^{-9}	6.39×10^{-5}	5	2.00
0.9	$\tilde{x}_1$	2.86×10^{-9}	5.38×10^{-5}	5	2.00
0.8	$\tilde{x}_1$	1.99×10^{-9}	4.53×10^{-5}	5	1.99
0.7	$\tilde{x}_1$	1.38×10^{-9}	3.81×10^{-5}	5	1.99
0.6	$\tilde{x}_1$	9.58×10^{-10}	3.20×10^{-5}	5	1.99
0.5	$\tilde{x}_1$	6.64×10^{-10}	2.68×10^{-5}	5	1.99
0.4	$\tilde{x}_1$	4.60×10^{-10}	2.25×10^{-5}	5	1.99
0.3	$\tilde{x}_1$	3.19×10^{-10}	1.89×10^{-5}	5	1.98
0.2	$\tilde{x}_1$	2.21×10^{-10}	1.58×10^{-5}	5	1.98
0.1	$\tilde{x}_1$	1.54×10^{-10}	1.33×10^{-5}	5	1.98

Tabla 4.8: Resultados para $F_4(x, y) = \hat{0}$ con estimación inicial $x^{(0)} = (-2.5, 3.5)^T$

con raíces reales $\tilde{x}_1 = (-1, \ldots, -1)^T$ y $\tilde{x}_2 = (1, \ldots, 1)^T$. La matriz Jacobiana conformable de $F_5(x)$ es

$$F_{\hat{a}}^{\alpha(1)}{}_5(x) = \begin{pmatrix} \chi_{1,1} & \chi_{1,2} & 0 & \ldots & \ldots & 0 & 0 \\ 0 & \chi_{2,2} & \chi_{2,3} & 0 & \ldots & 0 & 0 \\ & & & \vdots & & & \\ 0 & 0 & \ldots & \ldots & 0 & \chi_{14,14} & \chi_{14,15} \\ \chi_{15,1} & 0 & \ldots & \ldots & \ldots & 0 & \chi_{15,15} \end{pmatrix},$$

donde

$$\begin{aligned} \chi_{1,1} &= x_2(x_1 - a_1)^{1-\alpha} \\ \chi_{1,2} &= x_1(x_2 - a_2)^{1-\alpha} \\ \chi_{2,2} &= x_3(x_2 - a_2)^{1-\alpha} \\ \chi_{2,3} &= x_2(x_3 - a_3)^{1-\alpha} \\ \chi_{14,14} &= x_{15}(x_{14} - a_{14})^{1-\alpha} \\ \chi_{14,15} &= x_{14}(x_{15} - a_{15})^{1-\alpha} \\ \chi_{15,1} &= x_{15}(x_1 - a_1)^{1-\alpha} \\ \chi_{15,15} &= x_1(x_{15} - a_{15})^{1-\alpha}, \end{aligned}$$

siendo $a = (a_1, \ldots, a_{15}) = (-10, \ldots, -10)$.

Método NvCO					
α	$\tilde{x}$	$\|F_5(x^{(k+1)})\|$	$\|x^{(k+1)} - x^{(k)}\|$	iter	$\hat{\rho}$
1	$\tilde{x}_1$	1.02×10^{-10}	5.08×10^{-11}	5	2.00
0.9	$\tilde{x}_1$	8.55×10^{-11}	4.27×10^{-11}	5	2.00
0.8	$\tilde{x}_1$	7.18×10^{-11}	3.59×10^{-11}	5	2.00
0.7	$\tilde{x}_1$	6.01×10^{-11}	3.01×10^{-11}	5	2.00
0.6	$\tilde{x}_1$	5.02×10^{-11}	2.51×10^{-11}	5	2.00
0.5	$\tilde{x}_1$	4.17×10^{-11}	2.09×10^{-11}	5	2.00
0.4	$\tilde{x}_1$	3.47×10^{-11}	1.73×10^{-11}	5	2.00
0.3	$\tilde{x}_1$	2.87×10^{-11}	1.43×10^{-11}	5	2.00
0.2	$\tilde{x}_1$	2.36×10^{-11}	1.18×10^{-11}	5	2.00
0.1	$\tilde{x}_1$	1.93×10^{-11}	9.69×10^{-12}	5	2.00

Tabla 4.9: Resultados para $F_5(x) = \hat{0}$ con estimación inicial $x^{(0)} = (-1.5, \ldots, -1.5)^T$

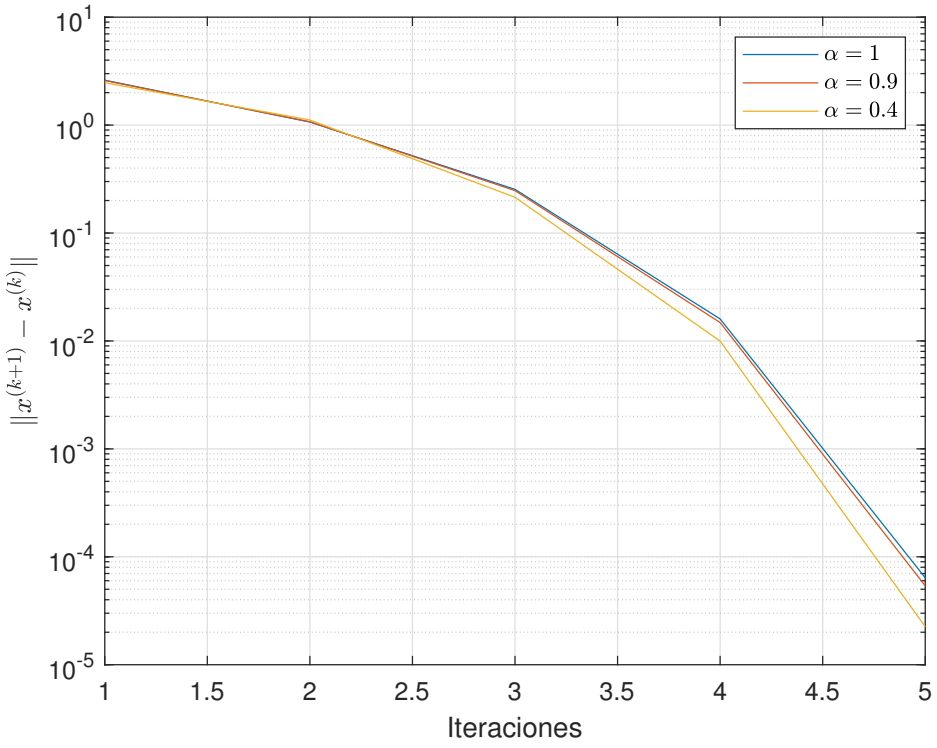


Figura 4.16: Curvas de error de $F_4(x, y)$ para algunos valores de α en la Tabla 4.8

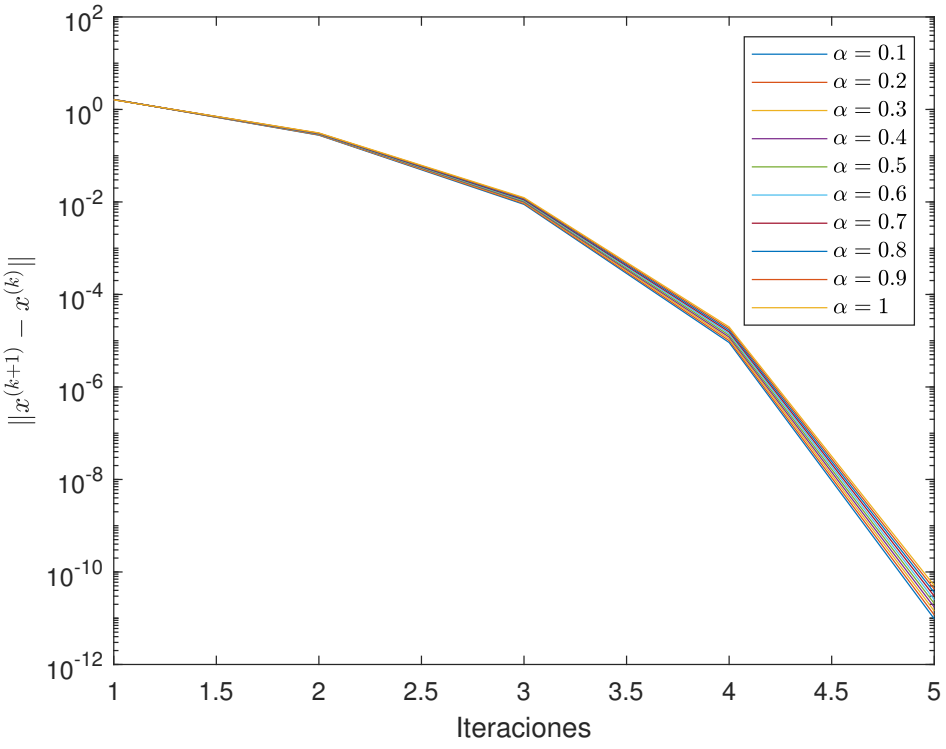


Figura 4.17: Curvas de error de $F_5(x)$ para todos los valores de α en la Tabla 4.9

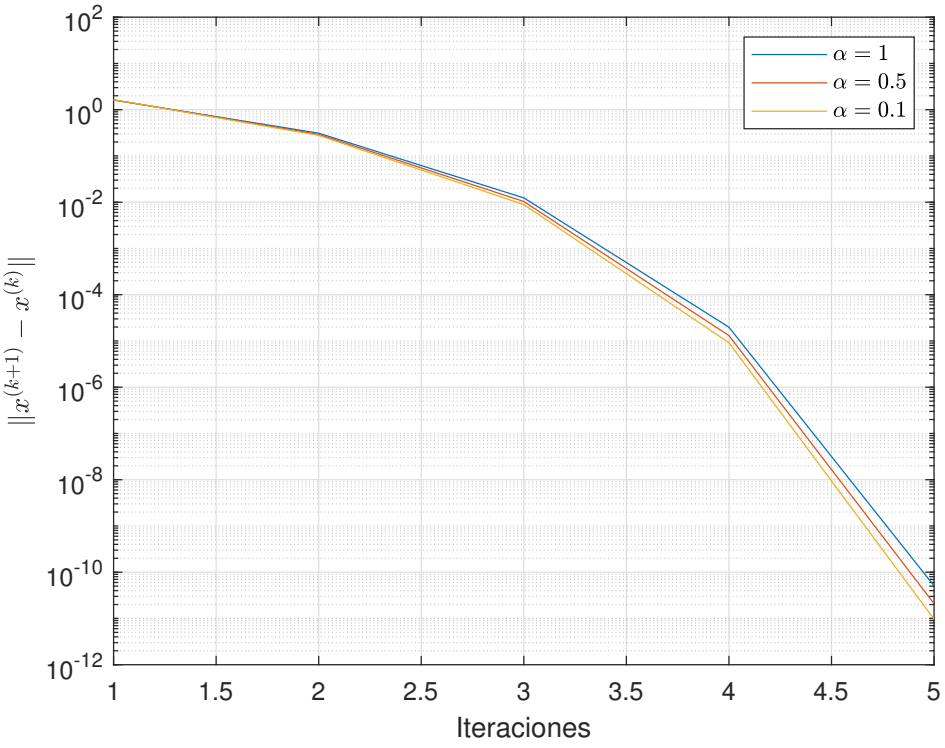


Figura 4.18: Curvas de error de $F_5(x)$ para algunos valores de α en la Tabla 4.9

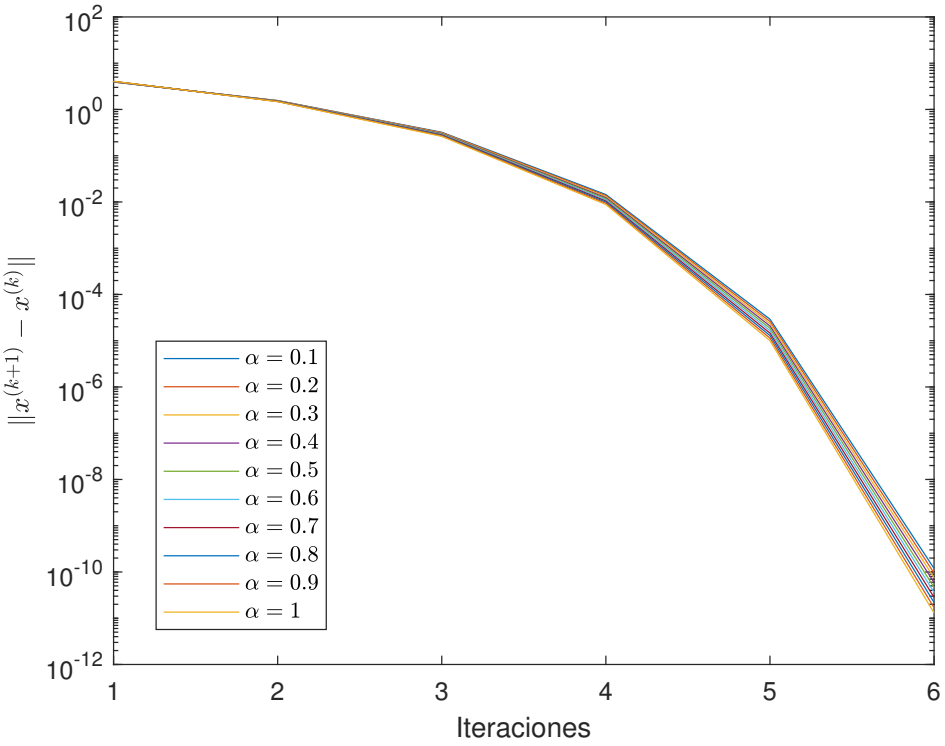


Figura 4.19: Curvas de error de $F_5(x)$ para todos los valores de α en la Tabla 4.10

Método NvCO					
α	$\tilde{x}$	$\|F_5(x^{(k+1)})\|$	$\|x^{(k+1)} - x^{(k)}\|$	iter	$\hat{\rho}$
1	$\tilde{x}_2$	2.60×10^{-11}	1.30×10^{-11}	6	2.00
0.9	$\tilde{x}_2$	3.38×10^{-11}	1.69×10^{-11}	6	2.00
0.8	$\tilde{x}_2$	4.38×10^{-11}	2.19×10^{-11}	6	2.00
0.7	$\tilde{x}_2$	5.65×10^{-11}	2.83×10^{-11}	6	2.00
0.6	$\tilde{x}_2$	7.26×10^{-11}	3.63×10^{-11}	6	2.00
0.5	$\tilde{x}_2$	9.28×10^{-11}	4.64×10^{-11}	6	2.00
0.4	$\tilde{x}_2$	1.18×10^{-10}	5.91×10^{-11}	6	2.00
0.3	$\tilde{x}_2$	1.50×10^{-10}	7.49×10^{-11}	6	2.00
0.2	$\tilde{x}_2$	1.89×10^{-10}	9.47×10^{-11}	6	2.00
0.1	$\tilde{x}_2$	2.38×10^{-10}	1.19×10^{-10}	6	2.00

Tabla 4.10: Resultados para $F_5(x) = \hat{0}$ con estimación inicial $x^{(0)} = (2.5, \ldots, 2.5)^T$

Podemos observar en las Tablas 4.9 y 4.10 para $F_5(x)$ que NvCo y el método clásico de Newton-Raphson tienen un comportamiento similar en cuanto a la cantidad de iteraciones y al orden de convergencia computacional. Podemos ver que las Figuras 4.17, 4.18, 4.19 y 4.20 muestran que los errores decrecen con las iteraciones, por tanto, no se observa comportamiento errático.

Finalmente, la sexta función vectorial de prueba es $F_6(x) = (f_1(x), \ldots, f_{10}(x))^T$, donde $x = (x_1, \ldots, x_{10})^T$ y $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, 2, \ldots, 9, 10$, tal que

$$f_i(x) = x_i - 1.5 \sin(x_1 + x_2 + \cdots + x_9 + x_{10} - x_i), \quad i = 1, 2, \ldots, 9, 10,$$

con raíces reales $\tilde{x}_1 \approx (-0.9691, \ldots, -0.9691)^T$, $\tilde{x}_2 \approx (-0.7569, \ldots, -0.7569)^T$, $\tilde{x}_3 \approx (-0.3248, \ldots, -0.3248)^T$, $\tilde{x}_4 = (0, \ldots, 0)^T$, $\tilde{x}_5 \approx (0.3248, \ldots, 0.3248)^T$, $\tilde{x}_6 \approx (0.7569, \ldots, 0.7569)^T$ y $\tilde{x}_7 \approx (0.9691, \ldots, 0.9691)^T$. La matriz Jacobiana conformable de $F_6(x)$ es

$$F_{\hat{a}}^{\alpha(1)}{}_6(x) = \begin{pmatrix} \chi_{1,1} & \chi_{1,2} & \cdots & \cdots & \cdots & \chi_{1,9} & \chi_{1,10} \\ \chi_{2,1} & \chi_{2,2} & \cdots & \cdots & \cdots & \chi_{2,9} & \chi_{2,10} \\ \vdots & \vdots & & & & \vdots & \vdots \\ \chi_{9,1} & \chi_{9,2} & \cdots & \cdots & \cdots & \chi_{9,9} & \chi_{9,10} \\ \chi_{10,1} & \chi_{10,2} & \cdots & \cdots & \cdots & \chi_{10,9} & \chi_{10,10} \end{pmatrix},$$

donde

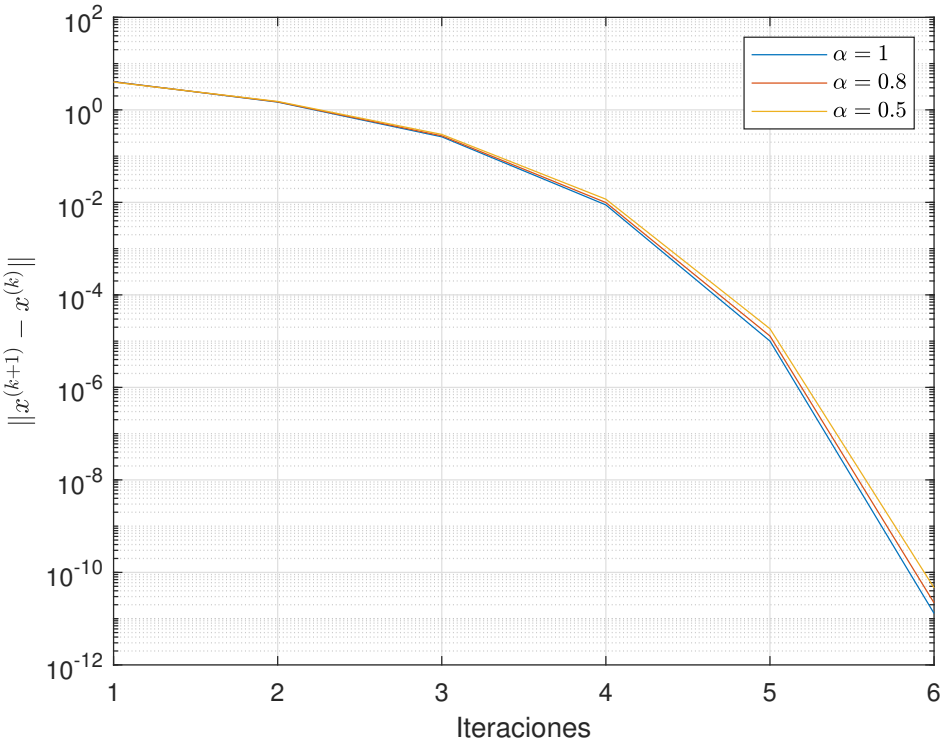


Figura 4.20: Curvas de error de $F_5(x)$ para algunos valores de α en la Tabla 4.10

$$
\begin{aligned}
\chi_{1,1} &= (x_1 - a_1)^{1-\alpha} \\
\chi_{1,2} &= (x_2 - a_2)^{1-\alpha}(-1.5\cos(x_1 + x_2 + \cdots + x_9 + x_{10} - x_2)) \\
\chi_{1,9} &= (x_9 - a_9)^{1-\alpha}(-1.5\cos(x_1 + x_2 + \cdots + x_9 + x_{10} - x_9)) \\
\chi_{1,10} &= (x_{10} - a_{10})^{1-\alpha}(-1.5\cos(x_1 + x_2 + \cdots + x_9 + x_{10} - x_{10})) \\
\chi_{2,1} &= (x_1 - a_1)^{1-\alpha}(-1.5\cos(x_1 + x_2 + \cdots + x_9 + x_{10} - x_1)) \\
\chi_{2,2} &= (x_2 - a_2)^{1-\alpha} \\
\chi_{2,9} &= (x_9 - a_9)^{1-\alpha}(-1.5\cos(x_1 + x_2 + \cdots + x_9 + x_{10} - x_9)) \\
\chi_{2,10} &= (x_{10} - a_{10})^{1-\alpha}(-1.5\cos(x_1 + x_2 + \cdots + x_9 + x_{10} - x_{10})) \\
\chi_{9,1} &= (x_1 - a_1)^{1-\alpha}(-1.5\cos(x_1 + x_2 + \cdots + x_9 + x_{10} - x_1)) \\
\chi_{9,2} &= (x_2 - a_2)^{1-\alpha}(-1.5\cos(x_1 + x_2 + \cdots + x_9 + x_{10} - x_2)) \\
\chi_{9,9} &= (x_9 - a_9)^{1-\alpha} \\
\chi_{9,10} &= (x_{10} - a_{10})^{1-\alpha}(-1.5\cos(x_1 + x_2 + \cdots + x_9 + x_{10} - x_{10})) \\
\chi_{10,1} &= (x_1 - a_1)^{1-\alpha}(-1.5\cos(x_1 + x_2 + \cdots + x_9 + x_{10} - x_1)) \\
\chi_{10,2} &= (x_2 - a_2)^{1-\alpha}(-1.5\cos(x_1 + x_2 + \cdots + x_9 + x_{10} - x_2)) \\
\chi_{10,9} &= (x_9 - a_9)^{1-\alpha}(-1.5\cos(x_1 + x_2 + \cdots + x_9 + x_{10} - x_9)) \\
\chi_{10,10} &= (x_{10} - a_{10})^{1-\alpha},
\end{aligned}
$$

siendo $a = (a_1, \ldots, a_{10}) = (-10, \ldots, -10)$.

Podemos ver en la Tabla 4.11 para $F_6(x)$ que NvCO requiere, en general, muchas menos iteraciones que el esquema clásico de Newton-Raphson. También podemos observar que el orden de convergencia computacional puede ser ligeramente superior a 2, y que es posible obtener distintas soluciones al cambiar el valor de α. En

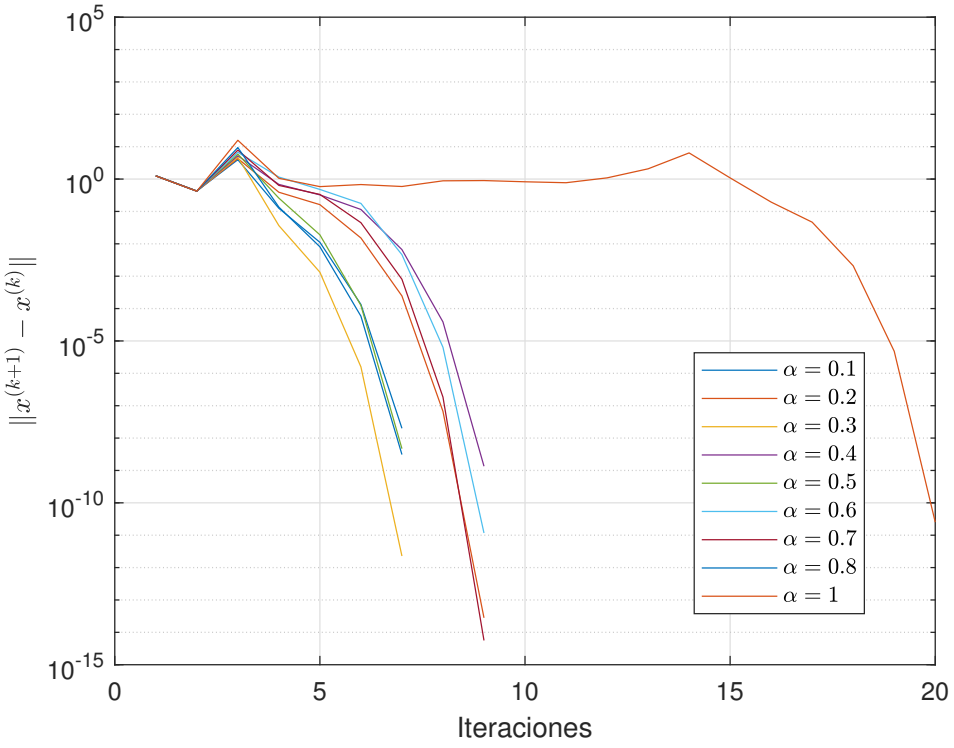


Figura 4.21: Curvas de error de $F_6(x)$ para todos los valores de α en la Tabla 4.11

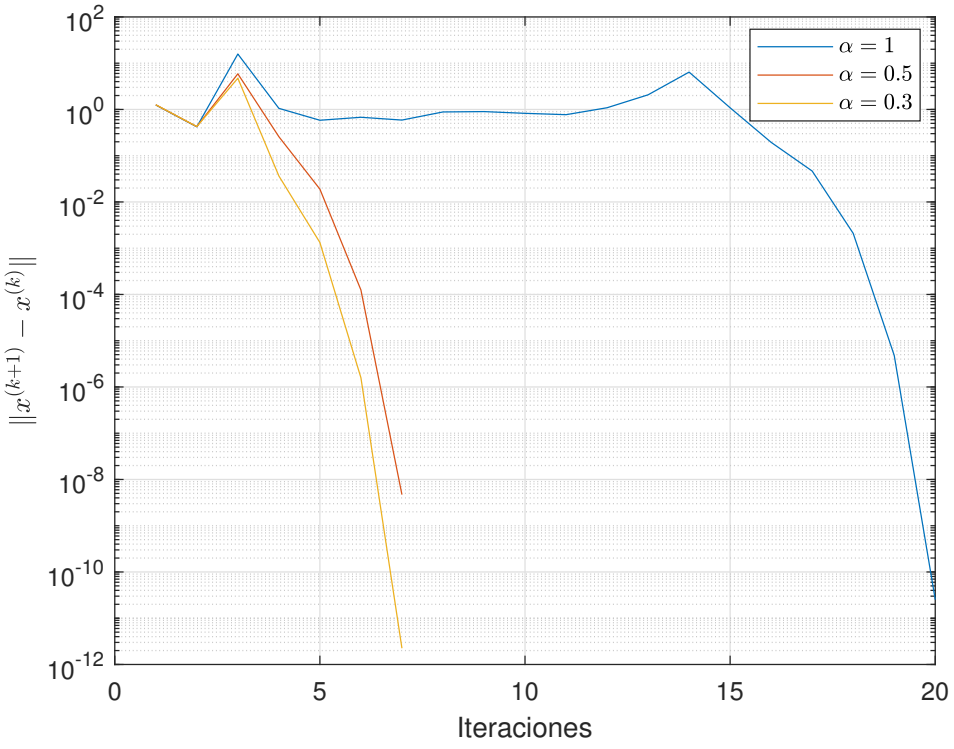


Figura 4.22: Curvas de error de $F_6(x)$ para algunos valores de α en la Tabla 4.11

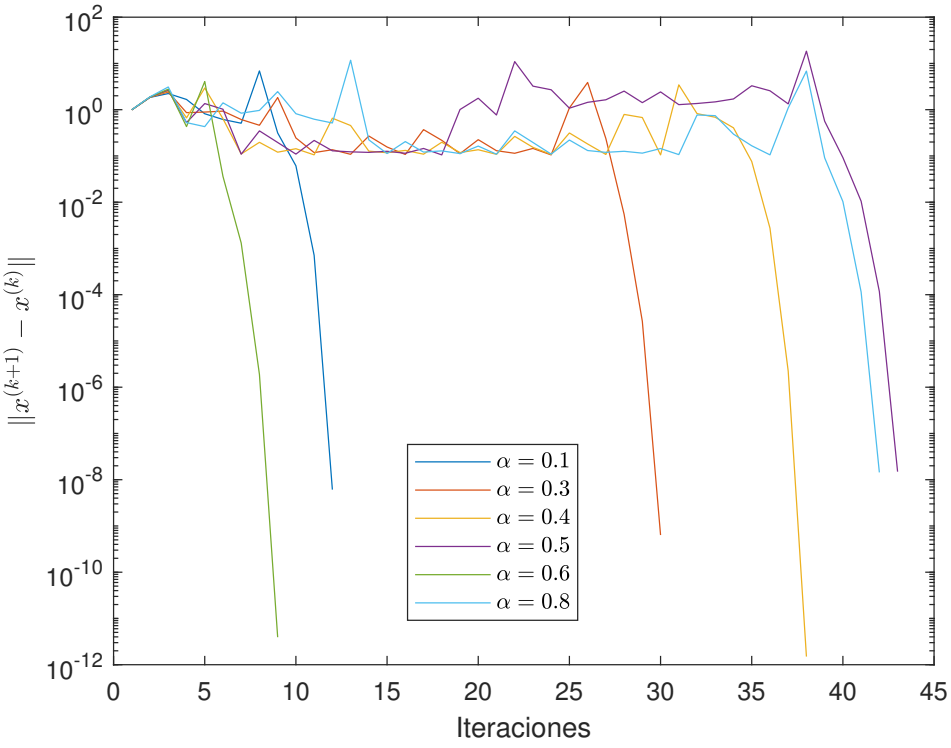


Figura 4.23: Curvas de error de $F_6(x)$ para todos los valores de α en la Tabla 4.12

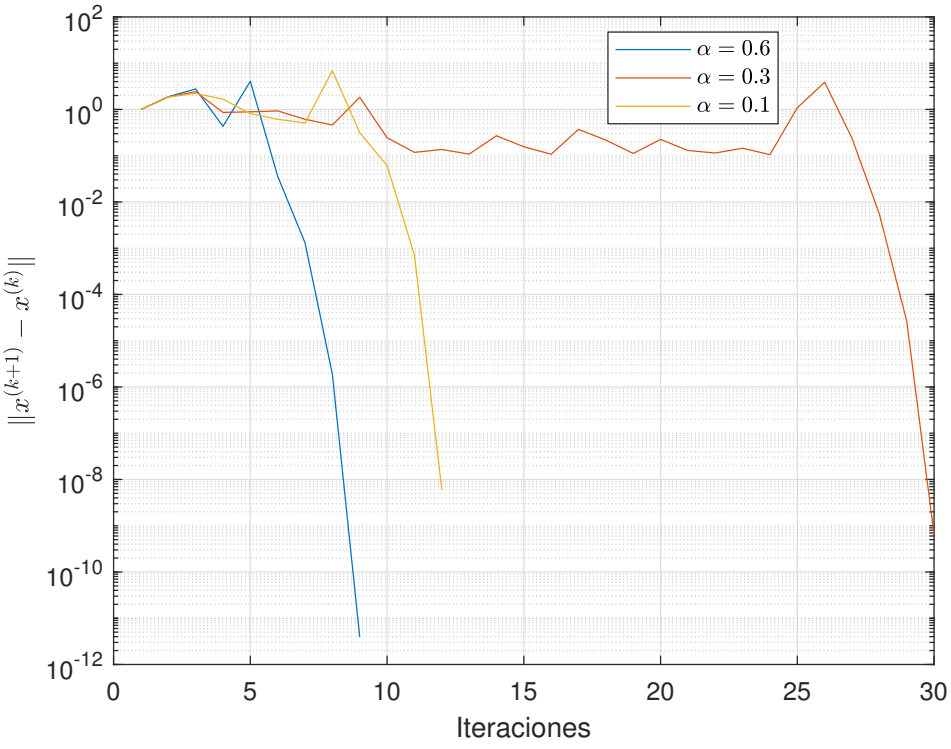


Figura 4.24: Curvas de error de $F_6(x)$ para algunos valores de α en la Tabla 4.12

Método NvCO					
α	$\tilde{x}$	$\|F_6(x^{(k+1)})\|$	$\|x^{(k+1)} - x^{(k)}\|$	iter	$\hat{\rho}$
1	$\tilde{x}_1$	2.86×10^{-10}	2.53×10^{-11}	20	2.00
0.9	-	-	-	> 500	-
0.8	$\tilde{x}_2$	3.30×10^{-8}	3.10×10^{-9}	7	1.99
0.7	$\tilde{x}_3$	1.40×10^{-13}	5.62×10^{-15}	9	2.07
0.6	$\tilde{x}_3$	1.66×10^{-10}	1.17×10^{-11}	9	2.01
0.5	$\tilde{x}_5$	6.67×10^{-8}	4.70×10^{-9}	7	2.03
0.4	$\tilde{x}_6$	1.44×10^{-8}	1.35×10^{-9}	9	2.00
0.3	$\tilde{x}_6$	2.40×10^{-11}	2.25×10^{-12}	7	2.00
0.2	$\tilde{x}_7$	1.87×10^{-13}	2.81×10^{-14}	9	1.79
0.1	$\tilde{x}_7$	1.87×10^{-13}	0	8	1.99

Tabla 4.11: Resultados para $F_6(x) = \hat{0}$ con estimación inicial $x^{(0)} = (2, \ldots, 2)^T$

Método NvCO					
α	$\tilde{x}$	$\|F_6(x^{(k+1)})\|$	$\|x^{(k+1)} - x^{(k)}\|$	iter	$\hat{\rho}$
1	-	-	-	> 500	-
0.9	-	-	-	> 500	-
0.8	$\tilde{x}_7$	6.04×10^{-14}	0	43	1.99
0.7	-	-	-	-	-
0.6	$\tilde{x}_7$	4.44×10^{-11}	3.93×10^{-12}	9	2.00
0.5	$\tilde{x}_1$	2.81×10^{-15}	0	44	1.99
0.4	$\tilde{x}_5$	2.15×10^{-11}	1.50×10^{-12}	38	2.00
0.3	$\tilde{x}_6$	6.77×10^{-9}	6.36×10^{-10}	30	2.00
0.2	-	-	-	> 500	-
0.1	$\tilde{x}_4$	7.63×10^{-8}	6.11×10^{-9}	12	2.62

Tabla 4.12: Resultados para $F_6(x) = \hat{0}$ con estimación inicial $x^{(0)} = (3, \ldots, 3)^T$

las Figuras 4.21 y 4.22, vemos que las curvas de error detienen su comportamiento errático en las últimas iteraciones.

En la Tabla 4.12 observamos para $F_6(x)$ que el procedimiento clásico de Newton-Raphson no encuentra solución en 500 iteraciones, mientras que NvCO converge para la mayoría de los valores de α. Observamos que el orden de convergencia computacional es alrededor de 2, pero mucho mayor que 2 cuando $\alpha = 0.1$. No se muestran resultados cuando $\alpha = 0.7$ porque la matriz Jacobiana conformable se hace singular en este caso. Tampoco se muestran resultados cuando no se encuentra solución en 500 iteraciones (para $\alpha = 0.9$ y $\alpha = 0.2$). Podemos notar que también se pueden obtener diferentes soluciones al escoger un valor de α distinto. De nuevo, en las Figuras 4.23 y 4.24 podemos observar que las curvas de error detienen su comportamiento errático en las últimas iteraciones.

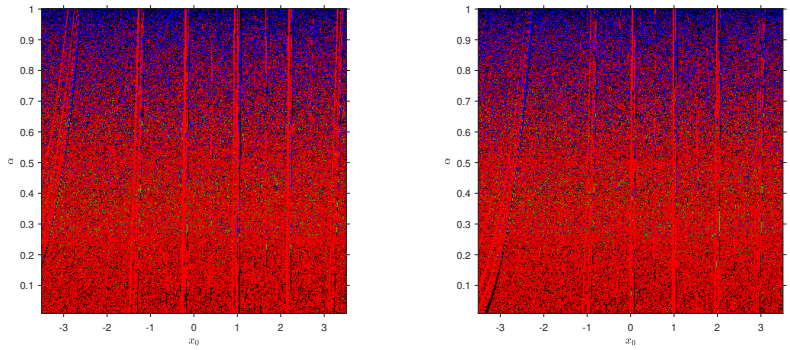
En las Tablas 4.11 y 4.12 algunos errores son cero porque se usa aritmética de precisión doble. Se podría observar un valor muy cercano a cero si se usara aritmética de precisión variable.

4.5.1. Estabilidad numérica

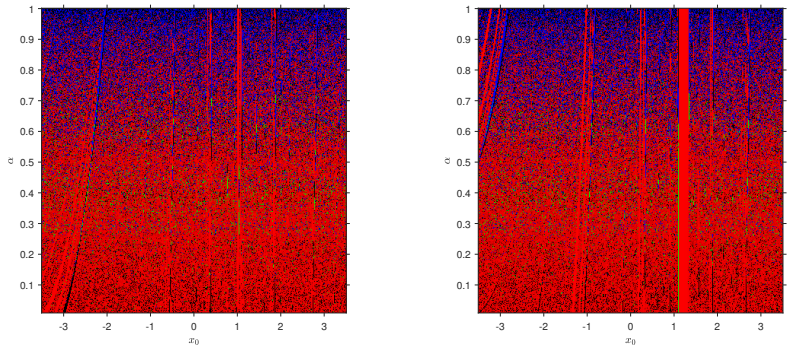
En esta subsección, estudiamos la estabilidad de NvCO; en tal sentido, analizamos la dependencia respecto a las estimaciones iniciales por medio de planos de convergencia definidos en [29]. Sólo podemos observar dos dimensiones en los planos de convergencia, por lo tanto, los vamos a proporcionar sólo para las funciones vectoriales F_1, F_2, F_3 y F_4 (ver fichero MATLAB en Anexo A.23).

Para la construcción de los planos de convergencia, consideramos de la estimación inicial (x_0, y_0), los puntos x_0 en el eje horizontal, y los valores de $\alpha \in (0, 1]$ en el eje vertical. Cada uno de los 8 planos que se proveen en cada figura representa un valor diferente de y_0 de la estimación inicial (x_0, y_0). Cada color representa

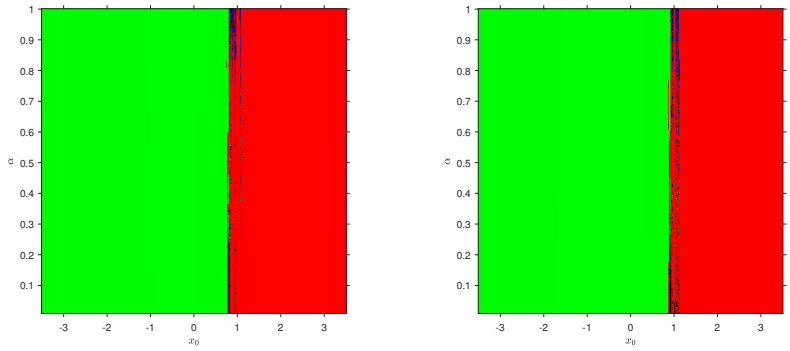
una solución diferente encontrada, y se representa en negro cuando no se encuentra solución después de 500 iteraciones. Cada plano se hace con una malla de 400×400, con un máximo de 500 iteraciones, y tolerancia de 0.001.



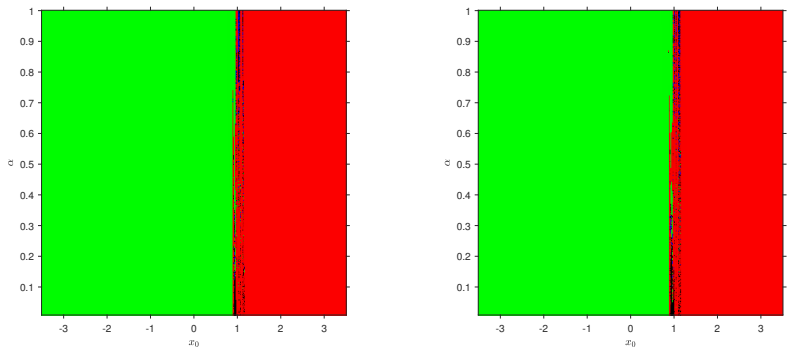
(a) $y_0 = -3.5$, 86.22 % de convergencia **(b)** $y_0 = -2.5$, 86.14 % de convergencia



(c) $y_0 = -1.5$, 85.99 % de convergencia **(d)** $y_0 = -0.5$, 86.61 % de convergencia

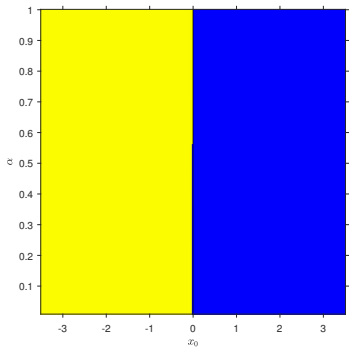


(e) $y_0 = 0.5$, 99.66 % de convergencia **(f)** $y_0 = 1.5$, 99.62 % de convergencia

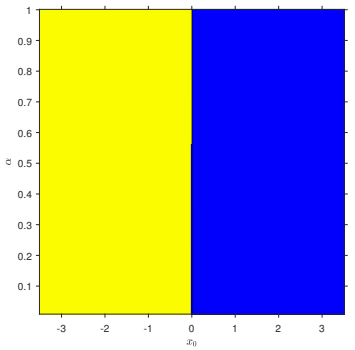


(g) $y_0 = 2.5$, 99.68 % de convergencia **(h)** $y_0 = 3.5$, 99.60 % de convergencia

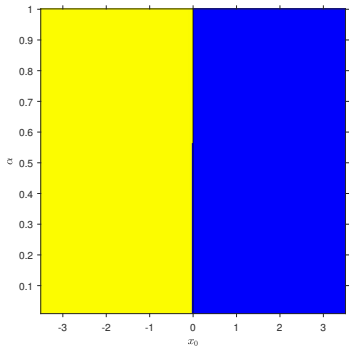
Figura 4.25: Planos de convergencia para $F_1(x, y)$. $\tilde{x}_1$: verde, $\tilde{x}_2$: rojo, $\tilde{x}_3$: azul



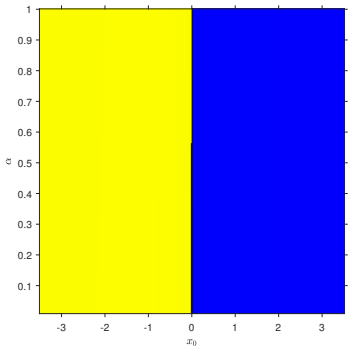
(a) $y_0 = -3.5$, 99.75 % de convergencia



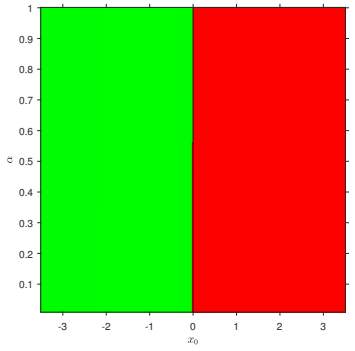
(b) $y_0 = -2.5$, 99.75 % de convergencia



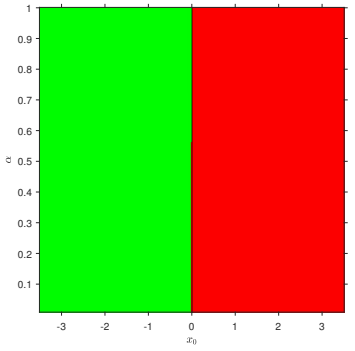
(c) $y_0 = -1.5$, 99.75 % de convergencia



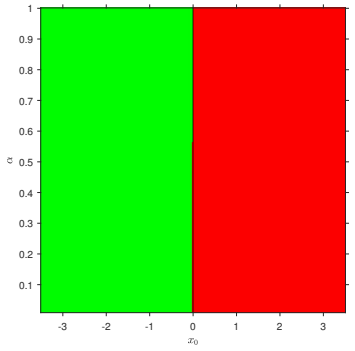
(d) $y_0 = -0.5$, 99.75 % de convergencia



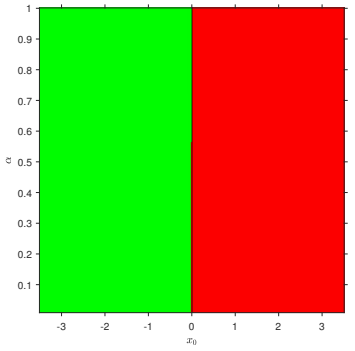
(e) $y_0 = 0.5$, 99.75 % de convergencia



(f) $y_0 = 1.5$, 99.75 % de convergencia



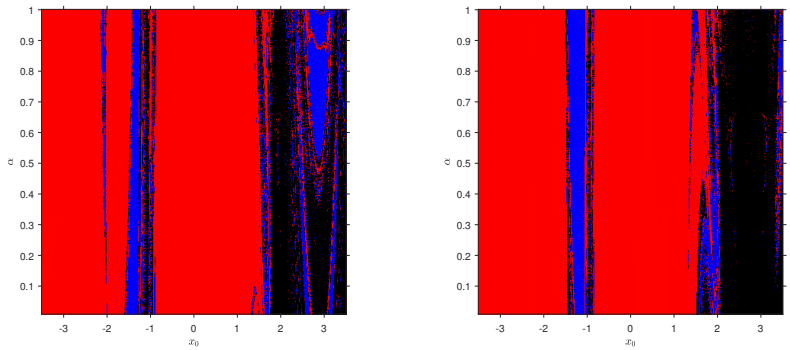
(g) $y_0 = 2.5$, 99.75 % de convergencia



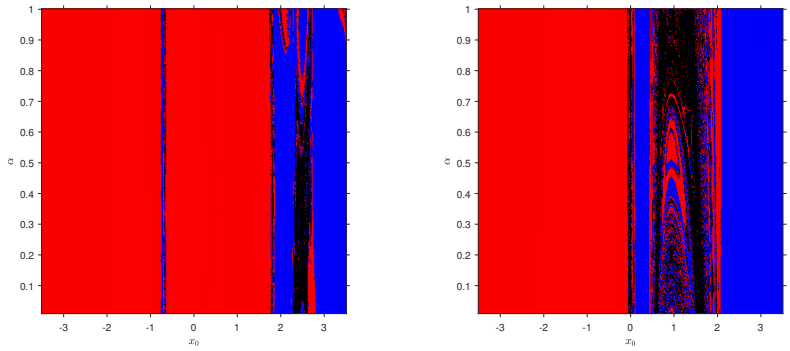
(h) $y_0 = 3.5$, 99.75 % de convergencia

Figura 4.26: Planos de convergencia para $F_2(x, y)$. $\tilde{x}_1$: rojo, $\tilde{x}_2$: verde, $\tilde{x}_3$: azul, $\tilde{x}_4$: amarillo

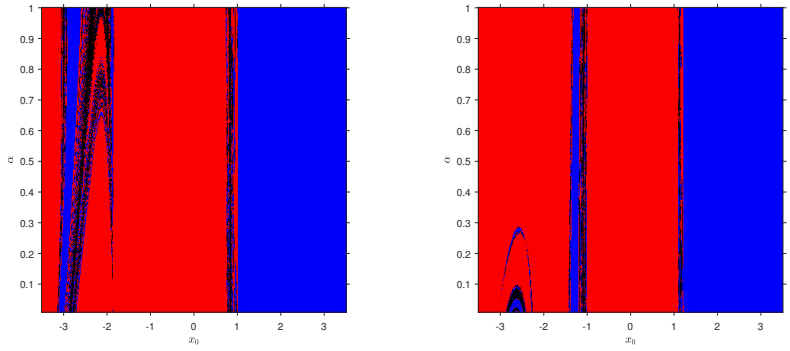
En la Figura 4.25, notemos que para el mismo rango de valores en x_0, se utilizan diferentes valores fijos de y_0 y se representan en las respectivas subfiguras; esto se mantiene para las demás figuras. Podemos ver que



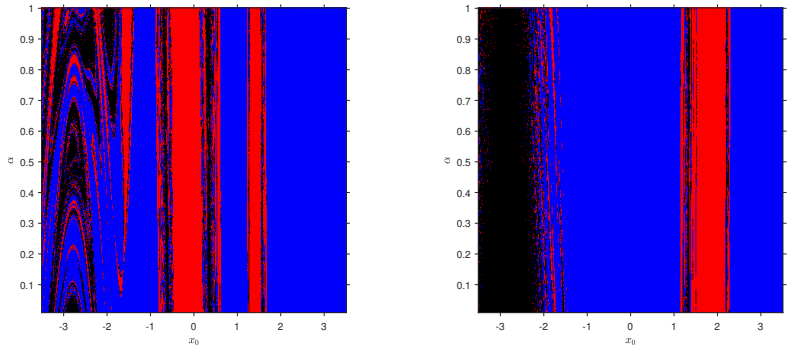
(a) $y_0 = -3.5$, 79.00 % de convergencia **(b)** $y_0 = -2.5$, 77.05 % de convergencia



(c) $y_0 = -1.5$, 95.81 % de convergencia **(d)** $y_0 = -0.5$, 87.42 % de convergencia



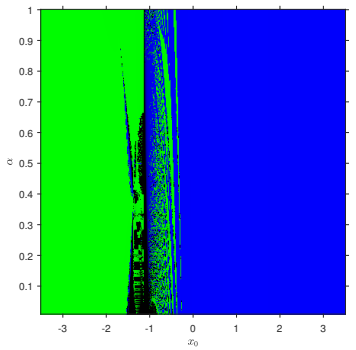
(e) $y_0 = 0.5$, 94.82 % de convergencia **(f)** $y_0 = 1.5$, 97.77 % de convergencia



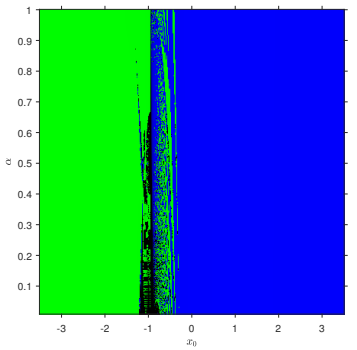
(g) $y_0 = 2.5$, 86.24 % de convergencia **(h)** $y_0 = 3.5$, 78.97 % de convergencia

Figura 4.27: Planos de convergencia para $F_3(x, y)$. $\tilde{x}_1$: rojo, $\tilde{x}_2$: azul

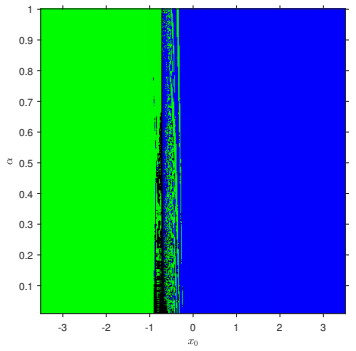
en la Figura 4.25, para $F_1(x, y)$ en (e), (f), (g) y (h) se obtiene casi un 100 % de convergencia, mientras que en (a), (b), (c) y (d) se obtiene alrededor del 86 % de convergencia. En cada caso, este método converge a



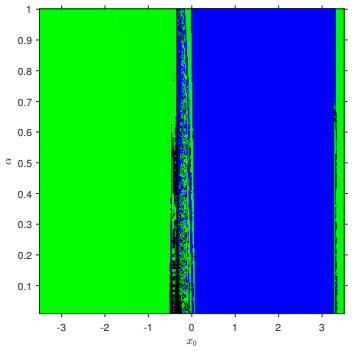
(a) $y_0 = -3.5$, 97.01 % de convergencia



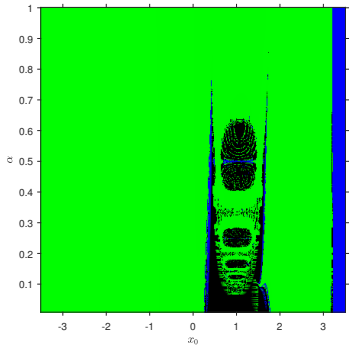
(b) $y_0 = -2.5$, 97.93 % de convergencia



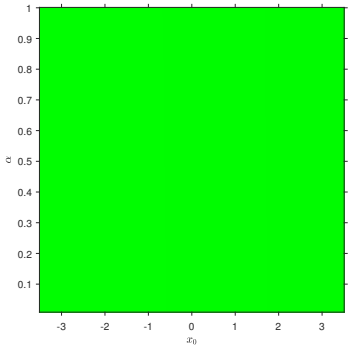
(c) $y_0 = -1.5$, 98.49 % de convergencia



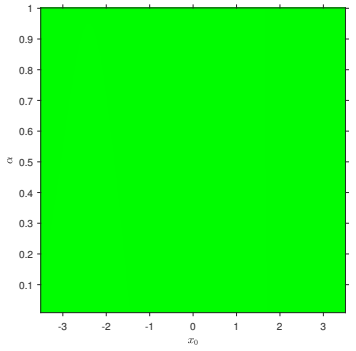
(d) $y_0 = -0.5$, 98.39 % de convergencia



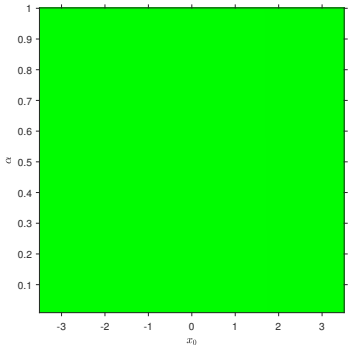
(e) $y_0 = 0.5$, 93.87 % de convergencia



(f) $y_0 = 1.5$, 100.00 % de convergencia



(g) $y_0 = 2.5$, 100.00 % de convergencia



(h) $y_0 = 3.5$, 100.00 % de convergencia

Figura 4.28: Planos de convergencia para $F_4(x, y)$. $\tilde{x}_1$: verde, $\tilde{x}_2$: azul

todas las raíces, incluso a una raíz compleja con estimación inicial real.

En la Figura 4.26, para $F_2(x, y)$ casi se consigue el 100 % de convergencia en cada plano. En (a), (b), (c) y

(d) este esquema converge a 2 de las 4 raíces, y en (e), (f), (g) y (h) converge a las otras 2 raíces.

Podemos observar que en la Figura 4.27, para $F_3(x, y)$ se obtiene entre el 77 % y el 98 % de convergencia. Este procedimiento converge a ambas raíces en cada plano.

En la Figura 4.28, para $F_4(x, y)$ podemos ver que se consigue el 100 % de convergencia en unos casos, y casi en otros. Para (a), (b), (c), (d) y (e) este método converge a ambas raíces, mientras que para (f), (g), y (h) converge a una sola raíz.

En este capítulo, hemos llevado a cabo el diseño, análisis de convergencia y estudio de la estabilidad de la versión vectorial del esquema conformable de tipo Newton-Raphson. En la teoría, hemos conseguido el orden de convergencia cuadrático del procedimiento clásico de Newton-Raphson. En la práctica hemos observado que, en general, es posible converger cuando el método clásico falla, el orden de convergencia computacional puede ser ligeramente superior a dos, se puede converger a raíces complejas partiendo de una estimación inicial real, y es posible encontrar distintas soluciones con la misma estimación inicial al variar el valor de α. En el siguiente capítulo, veremos el diseño, análisis de convergencia y estudio de la estabilidad de la versión fraccionaria de algunos métodos multipunto clásicos.

Capítulo 5

Métodos iterativos fraccionarios escalares multipunto

"La esencia de las matemáticas reside en su libertad"

Georg Cantor (1845 - 1918)

En este capítulo se muestran los trabajos desarrollados por Akgül et al en [3] y por Cordero et al en [15], cuyos esquemas iterativos fraccionarios de un solo paso son el punto de partida de los procedimientos fraccionarios multipunto. También se presenta el diseño, análisis de convergencia y estudio de la estabilidad de variantes fraccionarias de tipo Traub con derivadas de Caputo y de Riemann-Liouville[1] . Por último, se propone una técnica general para obtener la versión conformable de cualquier método clásico; utilizamos esta técnica para diseñar algunos esquemas conformables multipunto (de tipos Traub, Chun-Kim, Ostrowski y Chun), realizamos sus respectivos análisis de convergencia, y estudiamos la estabilidad de dichos procedimientos[2] .

Los métodos diseñados en [3] ya fueron tratados en el Capítulo 3 (véase (3.2) y (3.3)), donde los autores proponen las primeras variantes fraccionarias de tipo Newton-Raphson (con derivadas de Caputo y Riemann-Liouville) cuyo análisis de convergencia se proporciona, obteniendo orden 2α en cada caso. Posteriormente al trabajo publicado en [3], podemos encontrar en [15] los primeros intentos de aumentar el orden 2α, donde los autores diseñan tres esquemas de tipo Chebyshev con derivadas de Caputo basados en el procedimiento clásico de Chebyshev [34, 36, 42], cuyo orden de convergencia es cúbico. El primer método es el siguiente:

$$x_{k+1} = x_k - \Gamma(\alpha+1)\left(1+\frac{1}{2}cL_f^{\alpha}(x_k)\right)\frac{f(x_k)}{cD_{0^+}^{\alpha}f(x_k)}, \quad k=0,1,2,\ldots, \tag{5.1}$$

donde

$$cL_f^{\alpha}(x) = \frac{f(x)cD_{0^+}^{2\alpha}f(x)}{\left(cD_{0^+}^{\alpha}f(x)\right)^2}, \tag{5.2}$$

y con ecuación del error

[1] Resultados parciales de este capítulo forman parte del artículo *Multipoint Fractional Iterative Methods with (2α+1)th-Order of Convergence for Solving Nonlinear Problems* publicado en Mathematics, MDPI [10], del Capítulo 5 del libro *Fractional-Order Modeling of Dynamic Systems with Application in Optimization, Signal Processing, and Control* publicado en Academic Press, Elsevier [38], y presentados en el congreso CEDYA CMA 2020, Gijón, España

[2] Los resultados de este capítulo forman parte del manuscrito *On a new technique to generate optimal multipoint fractional methods for solving nonlinear equations* enviado para su posible publicación a Journal of Mathematical Analysis and Applications, Elsevier [13]

$$e_{k+1}^{\alpha} = \frac{2\Gamma^2(\alpha+1) - \Gamma(2\alpha+1)}{2\Gamma^2(\alpha+1)} C_2^{CA} e_k^{2\alpha} + O\left(e_k^{3\alpha}\right),$$

cuyo orden de convergencia es al menos 2α para $0 < \alpha \leq 1$, siendo $\Gamma^n(\cdot) = (\Gamma(\cdot))^2$, y $C_j^{CA} = \frac{\Gamma(\alpha+1)}{\Gamma(j\alpha+1)} \frac{{}_cD_{0^+}^{j\alpha} f(\bar{x})}{{}_cD_{0^+}^{\alpha} f(\bar{x})}$. El segundo esquema es el siguiente:

$$x_{k+1} = x_k - \Gamma(\alpha+1) \frac{f(x_k)}{{}_cD_{0^+}^{\alpha} f(x_k)} \left(1 + \frac{1}{2} \frac{f(x_k) {}_cD_{0^+}^{\alpha+1} f(x_k)}{{}_cD_{0^+}^{\alpha} f(x_k) {}_cD_{0^+}^{\alpha} f(x_k)}\right), \quad k = 0, 1, 2, \ldots \tag{5.3}$$

En (5.3) los autores cambiaron la derivada de orden 2α en (5.2) por una de orden $\alpha + 1$. La ecuación del error es

$$e_{k+1}^{\alpha} = \left(\frac{\Gamma^2(\alpha+1) - \Gamma(2\alpha+1)}{\Gamma^3(\alpha+1)} C_2^{CA} + \frac{1}{2} \frac{1}{\Gamma^2(\alpha+1)} \frac{{}_cD_{0^+}^{\alpha+1} f(\bar{x})}{{}_cD_{0^+}^{\alpha} f(\bar{x})}\right) e_k^{2\alpha} + O\left(e_k^{3\alpha+1}\right),$$

y con orden de convergencia de al menos 2α para $0 < \alpha \leq \frac{2}{3}$, y por otro lado, la ecuación del error es

$$e_{k+1}^{\alpha} = \left(\frac{\Gamma^2(\alpha+1) - \Gamma(2\alpha+1)}{\Gamma^3(\alpha+1)} C_2^{CA} + \frac{1}{2} \frac{1}{\Gamma^2(\alpha+1)} \frac{{}_cD_{0^+}^{\alpha+1} f(\bar{x})}{{}_cD_{0^+}^{\alpha} f(\bar{x})}\right) e_k^{2\alpha} + O\left(e_k^{3\alpha}\right),$$

con orden de convergencia de al menos 2α para $\frac{2}{3} \leq \alpha < 1$. Por último, el tercer procedimiento es:

$$x_{k+1} = x_k - \Gamma(\alpha+1) \frac{f(x_k)}{{}_cD_{0^+}^{\alpha} f(x_k)} \left(A + Bc L_f^{\alpha}(x_k)\right), \quad k = 0, 1, 2, \ldots, \tag{5.4}$$

con ecuación del error

$$\begin{aligned} e_{k+1}^{\alpha} = \Bigg[& -\Gamma(2\alpha+1) \left(1 - \frac{\Gamma(2\alpha+1)}{\Gamma^4(\alpha+1)}\right) C_2^{CA} + \frac{B\Gamma(2\alpha+1)}{\Gamma^3(\alpha+1)} \left(2 - 3\frac{\Gamma(2\alpha+1)}{\Gamma^2(\alpha+1)}\right) C_2^{CA^2} \\ & + \frac{1}{\Gamma(\alpha+1)} \left(\frac{B\Gamma(3\alpha+1)}{\Gamma^3(\alpha+1)} - \frac{\Gamma(3\alpha+1)}{\Gamma(2\alpha+1)\Gamma(\alpha+1)} + 1\right) C_3^{CA} \Bigg] e_k^{3\alpha} + O\left(e_k^{4\alpha}\right). \end{aligned}$$

Este método es de orden 3α sólo si $A = 1$ y $B = \frac{\Gamma(2\alpha+1) - \Gamma^2(\alpha+1)}{\Gamma(2\alpha+1)}$ para $0 < \alpha \leq 1$.

Como hemos visto, en [15] fue posible el desarrollo de esquemas de tipo Chebyshev que superan el orden de los procedimientos (3.2) y (3.3) del Capítulo 3. Al igual que los métodos de un solo paso de tipo Newton-Raphson de mayor orden vistos en el capítulo 3 (NeL3, NeA3 y NeLA4), los esquemas de tipo Chebyshev tienen la desventaja de que usan derivadas de alto orden, lo cual involucra una alta complejidad computacional, además de que no son procedimientos óptimos; estas son las razones por la cual los métodos multipunto son ampliamente utilizados en la literatura. Para superar el orden 3α de (5.4), diseñamos las variantes de tipo Newton-Raphson NeCA (3.4) y NeRL (3.5) vistas en el Capítulo 3, cuyo orden es $\alpha + 1$ en cada caso, superando el orden 2α de los métodos (3.2) y (3.3), para usarlos luego como predictores de esquemas multipunto de tipo Traub.

En la siguiente sección, diseñamos y analizamos la convergencia de dos procedimientos fraccionarios de tipo Traub.

5.1. Métodos de tipo Traub con derivadas de Caputo y Riemann-Liouville

A continuación se muestra el diseño de nuevos métodos fraccionarios de tipo Traub con derivadas de Caputo y de Riemann-Liouville que publicamos en la revista *Mathematics* (ver [10]).

5.1.1. Deducción de los métodos

Para el diseño de estos métodos, partimos de un doble Newton-Raphson, y congelamos la derivada en el segundo paso. Teniendo esto en cuenta, usaremos los métodos de Newton-Raphson con derivada de Caputo (3.4) y con derivada de Riemann-Liouville (3.5) como predictores de los esquemas de tipo Traub con el uso de estas derivadas, respectivamente. De esta manera, el procedimiento de tipo Traub con derivada de Caputo resulta

$$x_{k+1} = y_k - \left(\Gamma(\alpha+1)\frac{f(y_k)}{cD^{\alpha}_{0^+}f(x_k)}\right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots,$$

donde

$$y_k = x_k - \left(\Gamma(\alpha+1)\frac{f(x_k)}{cD^{\alpha}_{0^+}f(x_k)}\right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots$$

que denotaremos por TeCA. Por otro lado, el método de tipo Traub con derivada de Riemann-Liouville resulta

$$x_{k+1} = y_k - \left(\Gamma(\alpha+1)\frac{f(y_k)}{D^{\alpha}_{0^+}f(x_k)}\right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots,$$

donde

$$y_k = x_k - \left(\Gamma(\alpha+1)\frac{f(x_k)}{D^{\alpha}_{0^+}f(x_k)}\right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots$$

que denotaremos por TeRL.

5.1.2. Análisis de convergencia

En el siguiente resultado se establecen las condiciones que aseguran la convergencia de TeCA usando el desarrollo de Taylor (2.45).

Teorema 5.1.1 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función con derivada fraccionaria de Caputo de orden αk, para todo entero positivo k y para cualquier $\alpha \in (0, 1]$ en el intervalo I, conteniendo el cero $\bar{x}$ de $f(x)$. Supongamos que $cD^{\alpha}_{0^+}f(x)$ es continua y no nula en $\bar{x}$. Si una aproximación inicial x_0 es lo suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del esquema de Traub fraccionario con derivada de Caputo (TeCA)*

$$x_{k+1} = y_k - \left(\Gamma(\alpha+1)\frac{f(y_k)}{cD^{\alpha}_{0^+}f(x_k)}\right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots, \tag{5.5}$$

donde

$$y_k = x_k - \left(\Gamma(\alpha+1)\frac{f(x_k)}{{}_cD^{\alpha}_{0^+}f(x_k)}\right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots$$

es al menos $2\alpha+1$, $0 < \alpha \leq 1$, *y la ecuación del error es*

$$e_{k+1} = -\frac{\Gamma(2\alpha+1)}{\alpha^2\Gamma^2(\alpha+1)}A_1 C_2^{CA\,2} e_k^{2\alpha+1} + O\left(e_k^{\alpha^2+2\alpha+1}\right),$$

siendo $\alpha^2+2\alpha+1 < 3\alpha+1$ *para* $\alpha \in (0,1]$,

$$A_1 = \frac{\Gamma^2(\alpha+1) - \Gamma(2\alpha+1)}{\Gamma^2(\alpha+1)},$$

y

$C_j^{CA} = \dfrac{\Gamma(\alpha+1)}{\Gamma(j\alpha+1)}\dfrac{{}_cD^{j\alpha}_{0^+}f(\bar{x})}{{}_cD^{\alpha}_{0^+}f(\bar{x})}$, $j \geq 2$, *tal que* $x_k > 0$, $\forall k$.

Demostración. El primer paso y_k de (5.5) es (3.4), cuyo orden de convergencia fue probado en el Teorema 3.2.1. El desarrollo de Taylor del error cometido en el paso y_k en torno a $\bar{x}$ hasta orden $2\alpha+1$ se puede expresar como

$$y_k - \bar{x} = -\frac{1}{\alpha}A_1 C_2^{CA} e_k^{\alpha+1} + \frac{1}{\alpha}\left[\left(\frac{\Gamma(2\alpha+1)}{\Gamma^2(\alpha+1)} + \frac{\alpha-1}{2\alpha}A_1\right)A_1 C_2^{CA\,2} - A_2 C_3^{CA}\right] e_k^{2\alpha+1} + O\left(e_k^{3\alpha+1}\right),$$

donde

$$A_1 = \frac{\Gamma^2(\alpha+1) - \Gamma(2\alpha+1)}{\Gamma^2(\alpha+1)},$$

y

$$A_2 = \frac{\Gamma(\alpha+1)\Gamma(2\alpha+1) - \Gamma(3\alpha+1)}{\Gamma(\alpha+1)\Gamma(2\alpha+1)}.$$

El desarrollo de $f(y_k)$ es

$$f(y_k) = \frac{{}_cD^{\alpha}_{0^+}f(\bar{x})}{\Gamma(\alpha+1)}\left[(y_k-\bar{x})^{\alpha} + C_2^{CA}(y_k-\bar{x})^{2\alpha}\right] + O\left(e_k^{3\alpha+1}\right).$$

Usando el Teorema generalizado del binomio (2.52) y el coeficiente binomial fraccionario (2.53),

$$\begin{aligned}(y_k-\bar{x})^{\alpha} &= \left(-\frac{1}{\alpha}\right)^{\alpha} A_1^{\alpha} C_2^{CA\,\alpha} e_k^{\alpha^2+\alpha} + \left(-\frac{1}{\alpha}\right)^{\alpha-1} A_1^{\alpha-1} C_2^{CA\,\alpha-1}\left[\left(\frac{\Gamma(2\alpha+1)}{\Gamma^2(\alpha+1)} + \frac{\alpha-1}{2\alpha}A_1\right)A_1 C_2^{CA\,2}\right.\\ &- \left. A_2 C_3^{CA}\right] e_k^{\alpha^2+2\alpha} + \frac{\alpha^2-\alpha}{2}\left(-\frac{1}{\alpha}\right)^{\alpha} A_1^{\alpha-2} C_2^{CA\,\alpha-2}\left[\left(\frac{\Gamma(2\alpha+1)}{\Gamma^2(\alpha+1)} + \frac{\alpha-1}{2\alpha}A_1\right)A_1 C_2^{CA\,2}\right.\\ &- \left. A_2 C_3^{CA}\right]^2 e_k^{\alpha^2+3\alpha} + O\left(e_k^{3\alpha+1}\right)\end{aligned}$$

y

$$(y_k-\bar{x})^{2\alpha} = \left(-\frac{1}{\alpha}\right)^{2\alpha} A_1^{2\alpha} C_2^{CA^{2\alpha}} e_k^{2\alpha^2+2\alpha} + O\left(e_k^{3\alpha+1}\right).$$

Notemos que, para todo $\alpha \in (0,1)$, $\alpha^2+3\alpha < 3\alpha+1$ y $2\alpha^2+2\alpha < 3\alpha+1$, pero $\alpha^2+4\alpha > 3\alpha+1$ y $2\alpha^2+3\alpha > 3\alpha+1$.

Por lo tanto,

$$\begin{aligned} f(y_k) &= \frac{{}_{c}D_{0^+}^{\alpha} f(\bar{x})}{\Gamma(\alpha+1)} \left[\left(-\frac{1}{\alpha}\right)^{\alpha} A_1^{\alpha} C_2^{CA^{\alpha}} e_k^{\alpha^2+\alpha} + \left(-\frac{1}{\alpha}\right)^{\alpha-1} A_3 e_k^{\alpha^2+2\alpha} \right. \\ &\quad \left. + \left(-\frac{1}{\alpha}\right)^{2\alpha} A_1^{2\alpha} C_2^{CA^{2\alpha+1}} e_k^{2\alpha^2+2\alpha} + \frac{\alpha-1}{2}\left(-\frac{1}{\alpha}\right)^{\alpha-2} A_4 e_k^{\alpha^2+3\alpha} \right] + O\left(e_k^{3\alpha+1}\right), \end{aligned}$$

siendo

$$A_3 = \left(\frac{\Gamma(2\alpha+1)}{\Gamma^2(\alpha+1)} + \frac{\alpha-1}{2\alpha} A_1\right) A_1^{\alpha} C_2^{CA^{\alpha+1}} - A_1^{\alpha-1} A_2 C_3^{CA} C_2^{CA^{\alpha-1}}$$

y

$$\begin{aligned} A_4 &= \frac{1}{\alpha}\left(\frac{\Gamma(2\alpha+1)}{\Gamma^2(\alpha+1)} + \frac{\alpha-1}{2\alpha} A_1\right)^2 A_1^{\alpha} C_2^{CA^{\alpha+2}} + \frac{1}{\alpha} A_1^{\alpha-2} A_2^2 C_2^{CA^{\alpha-2}} C_3^{CA^2} \\ &\quad - \frac{2}{\alpha}\left(\frac{\Gamma(2\alpha+1)}{\Gamma^2(\alpha+1)} + \frac{\alpha-1}{2\alpha} A_1\right) A_1^{\alpha-1} A_2 C_2^{CA^{\alpha}} C_3^{CA}. \end{aligned}$$

El desarrollo del cociente $\dfrac{f(y_k)}{{}_{c}D_{0^+}^{\alpha} f(x_k)}$ resulta

$$\begin{aligned} \frac{f(y_k)}{{}_{c}D_{0^+}^{\alpha} f(x_k)} &= \frac{1}{\Gamma(\alpha+1)} \left[\left(-\frac{1}{\alpha}\right)^{\alpha} A_1^{\alpha} C_2^{CA^{\alpha}} e_k^{\alpha^2+\alpha} + \left(-\frac{1}{\alpha}\right)^{\alpha-1} \left(A_3 + \frac{\Gamma(2\alpha+1)}{\Gamma(\alpha+1)} \frac{1}{\alpha} A_1^{\alpha} C_2^{CA^{\alpha+1}}\right) e_k^{\alpha^2+2\alpha} \right. \\ &+ \left(-\frac{1}{\alpha}\right)^{2\alpha} A_1^{2\alpha} C_2^{CA^{2\alpha+1}} e_k^{2\alpha^2+2\alpha} + \left(-\frac{1}{\alpha}\right)^{\alpha-2} \left(\frac{\alpha-1}{2} A_4 \right. \\ &- \frac{\Gamma(3\alpha+1)}{\Gamma(\alpha+1)\Gamma(2\alpha+1)} \frac{1}{\alpha^2} A_1^{\alpha} C_2^{CA^{\alpha}} C_3^{CA} \frac{1}{\alpha} \frac{\Gamma(2\alpha+1)}{\Gamma^2(\alpha+1)} A_3 C_2^{CA} \\ &\left. \left. + \frac{\Gamma^2(2\alpha+1)}{\Gamma^4(\alpha+1)} \frac{1}{\alpha^2} A_1^{\alpha} C_2^{CA^{\alpha+2}} \right) e_k^{\alpha^2+3\alpha} \right] + O\left(e_k^{3\alpha+1}\right). \end{aligned}$$

Usando de nuevo el Teorema generalizado del binomio,

$$\begin{aligned} \left(\Gamma(\alpha+1)\frac{f(y_k)}{{}_{c}D_{0^+}^{\alpha} f(x_k)}\right)^{1/\alpha} &= \left(-\frac{1}{\alpha}\right) \left[A_1 C_2^{CA} e_k^{\alpha+1} + \left(-\frac{\Gamma(2\alpha+1)}{\Gamma(\alpha+1)} \left(1+\frac{1}{\alpha}\right) A_1 C_2^{CA^2} - \frac{\alpha-1}{2\alpha} A_1^2 C_2^{CA^2} \right. \right. \\ &+ \left. A_2 C_3^{CA} \right) e_k^{2\alpha+1} - \left(-\frac{1}{\alpha}\right)^{\alpha+1} A_1^{\alpha+1} C_2^{CA^{\alpha+2}} e_k^{\alpha^2+2\alpha+1} \Bigg] + O\left(e_k^{3\alpha+1}\right). \end{aligned}$$

Sea $x_{k+1} = e_{k+1} + \bar{x}$,

$$e_{k+1} = -\frac{\Gamma(2\alpha+1)}{\alpha^2\Gamma^2(\alpha+1)} A_1 C_2^{CA^2} e_k^{2\alpha+1} + O\left(e_k^{\alpha^2+2\alpha+1}\right).$$

Esto completa la prueba. □

En el siguiente resultado se plantean las condiciones que aseguran la convergencia de TeRL de manera similar, pero usando el desarrollo de Taylor (2.43). Aquí se omite la prueba, ya que es totalmente análoga a la demostración del Teorema 5.1.1.

Teorema 5.1.2 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función con derivada fraccionaria de Riemann-Liouville de orden αk, para todo entero positivo k y para cualquier $\alpha \in (0,1]$ en el intervalo I, conteniendo el cero $\bar{x}$ de $f(x)$. Supongamos que $D^{\alpha}_{0^+} f(x)$ es continua y no nula en $\bar{x}$. Si una aproximación inicial x_0 es lo suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del procedimiento de Traub fraccionario con derivada de Riemann-Liouville (TeRL)*

$$x_{k+1} = y_k - \left(\Gamma(\alpha+1)\frac{f(y_k)}{D^{\alpha}_{0^+} f(x_k)}\right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots, \tag{5.6}$$

donde

$$y_k = x_k - \left(\Gamma(\alpha+1)\frac{f(x_k)}{D^{\alpha}_{0^+} f(x_k)}\right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots$$

es al menos $2\alpha+1$, $0 < \alpha \leq 1$, y la ecuación del error es

$$e_{k+1} = -\frac{\Gamma(2\alpha+1)}{\alpha^2\Gamma^2(\alpha+1)} A_1 C_2^{RL^2} e_k^{2\alpha+1} + O\left(e_k^{\alpha^2+2\alpha+1}\right),$$

siendo $\alpha^2 + 2\alpha + 1 < 3\alpha + 1$ para $\alpha \in (0,1]$,

$$A_1 = \frac{\Gamma^2(\alpha+1) - \Gamma(2\alpha+1)}{\Gamma^2(\alpha+1)},$$

y

$C_j^{RL} = \dfrac{\Gamma(\alpha+1)}{\Gamma(j\alpha+1)} \dfrac{D^{j\alpha}_{0^+} f(\bar{x})}{D^{\alpha}_{0^+} f(\bar{x})}$, *$j \geq 2$, tal que $x_k > 0$, $\forall k$.*

En la siguiente sección, proponemos una técnica que nos permite obtener la versión conformable de cualquier método clásico, y la utilizamos para el diseño y análisis de convergencia de algunos esquemas conformables multipunto.

5.2. Técnica general para el diseño de métodos fraccionarios conformables basados en esquemas clásicos

Para construir una técnica general con el fin de obtener una versión conformable de un procedimiento clásico conocido, partimos del supuesto de que cualquier método se puede representar como (véase [42])

$$\phi(x) = x - f(x)g(x), \tag{5.7}$$

donde $g(\bar{x})$ es finito y no nulo, siendo $\bar{x}$ un punto fijo de ϕ. Notemos que si $g(x)$ en (5.7) es $1/f'(x)$, se obtiene el esquema clásico de Newton-Raphson.

Es bien sabido que considerando el desarrollo de Taylor de una función f en el entorno de la raíz $\bar{x}$ hasta orden uno como

$$f(x) \approx f(\bar{x}) + f'(\bar{x})(x - \bar{x}), \tag{5.8}$$

podemos conseguir, sabiendo que $f(\bar{x}) = 0$, el procedimiento clásico de Newton-Raphson despejando $\bar{x}$ de $(x - \bar{x})$ del lado derecho de (5.8), y considerando los iterados x_k y x_{k+1} como aproximaciones de la raíz $\bar{x}$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots \tag{5.9}$$

Procediendo en sentido contrario, de (5.9) podemos conseguir (5.8), de nuevo, sabiendo que $f(\bar{x}) = 0$. Por lo tanto, de (5.7) podemos obtener una representación general de la serie de Taylor clásica en el entorno de $\bar{x}$ hasta orden uno de f como

$$f(x) \approx \frac{1}{g(\bar{x})}(x - \phi(\bar{x})). \tag{5.10}$$

Considerando la serie de Taylor conformable (2.49), podemos conseguir la versión conformable de (5.10) como

$$f(x) \approx \frac{1}{\alpha g_\alpha(\bar{x})} \left[(x-a)^\alpha - (\phi(\bar{x}) - a)^\alpha\right]. \tag{5.11}$$

Si la expresión analítica de $g(x)$ incluye derivadas enteras de $f(x)$, entonces la expresión analítica de $g_\alpha(x)$ incluye derivadas conformables de $f(x)$. Ahora, despejando $\phi(\bar{x})$ de (5.11) tenemos

$$\phi(\bar{x}) \approx a + \left((x-a)^\alpha - \alpha f(x) g_\alpha(\bar{x})\right)^{1/\alpha}. \tag{5.12}$$

Considerando x como aproximación de $\bar{x}$, finalmente obtenemos la versión conformable de (5.7) como

$$\phi(x) = a + \left((x-a)^\alpha - \alpha f(x) g_\alpha(x)\right)^{1/\alpha}. \tag{5.13}$$

Cuando $g_\alpha(x)$ en (5.13) es $1/(T^a_\alpha f)(x)$, se obtiene el método NeCO (3.9). Por lo tanto, dado un esquema clásico, debemos identificar su expresión analítica de $g(x)$ para obtener su versión conformable. Hasta ahora, hemos podido ver que el orden de convergencia teórico de la versión conformable de un procedimiento iterativo se preserva, y además puede presentar en ocasiones algunas ventajas numéricas frente a la versión original; al menos con el uso de las versiones conformables de tipo Newton-Raphson (NeCO y NvCO) vistas en los Capítulos 3 y 4, hemos observado que, en general: Se puede encontrar solución cuando los métodos clásicos fallan (con $\alpha \neq 1$), el orden de convergencia computacional puede ser ligeramente superior y en algunos casos es posible converger en menos iteraciones que en el caso clásico, se puede conseguir una raíz distinta al variar el valor del orden fraccionario α, y es posible converger a raíces complejas partiendo de estimaciones iniciales reales.

5.2.1. Deducción de algunos métodos multipunto

Usemos (5.13) para diseñar algunos procedimientos conformables multipunto de tipos Traub, Chun-Kim, Ostrowski y Chun (véase [14, 36, 42]). Consideremos primero el método de Traub de tercer orden [36, 42]

$$\psi_1(x) = \phi_1(x) - \frac{f[\phi_1(x)]}{f'(x)}, \tag{5.14}$$

donde

$$\phi_1(x) = x - \frac{f(x)}{f'(x)}. \tag{5.15}$$

En este caso,

$$g(x) = \frac{1}{f'(x)}$$

en (5.14), por lo que

$$g_\alpha(x) = \frac{1}{(T_\alpha^a f)(x)}.$$

En consecuencia, la versión conformable del procedimiento de Traub denotado por TeCO es

$$\psi_2(x) = a + \left((\phi_2(x) - a)^\alpha - \alpha \frac{f[\phi_2(x)]}{(T_\alpha^a f)(x)}\right)^{1/\alpha}, \tag{5.16}$$

donde

$$\phi_2(x) = a + \left((x - a)^\alpha - \alpha \frac{f(x)}{(T_\alpha^a f)(x)}\right)^{1/\alpha}. \tag{5.17}$$

Consideremos ahora el método de Chun-Kim de tercer orden (véase [14, 36])

$$\psi_3(x) = x - \frac{1}{2}\left[3 - \frac{f'[\phi_3(x)]}{f'(x)}\right]\frac{f(x)}{f'(x)}, \tag{5.18}$$

donde

$$\phi_3(x) = x - \frac{f(x)}{f'(x)},$$

Aquí,

$$g(x) = \frac{1}{2}\left[3 - \frac{f'[\phi(x)]}{f'(x)}\right]\frac{1}{f'(x)}$$

en (5.18), y por tanto

$$g_\alpha(x) = \frac{1}{2}\left[3 - \frac{(T_\alpha^a f)[\phi(x)]}{(T_\alpha^a f)(x)}\right]\frac{1}{(T_\alpha^a f)(x)}.$$

Por consiguiente, la versión conformable del esquema de Chun-Kim denotado por CKeCO es

$$\psi_4(x) = a + \left((x-a)^\alpha - \frac{\alpha}{2}\left[3 - \frac{(T_\alpha^a f)[\phi_4(x)]}{(T_\alpha^a f)(x)}\right]\frac{f(x)}{(T_\alpha^a f)(x)}\right)^{1/\alpha}, \tag{5.19}$$

donde

$$\phi_4(x) = a + \left((x-a)^\alpha - \alpha\frac{f(x)}{(T_\alpha^a f)(x)}\right)^{1/\alpha}.$$

Ahora, consideremos un procedimiento óptimo de cuarto orden, el método de Ostrowski [36, 42]

$$\psi_5(x) = \phi_5(x) - \left[\frac{f(x)}{f(x) - 2f[\phi_5(x)]}\right]\frac{f[\phi_5(x)]}{f'(x)}, \tag{5.20}$$

donde

$$\phi_5(x) = x - \frac{f(x)}{f'(x)},$$

En este caso,

$$g(x) = \left[\frac{f(x)}{f(x) - 2f[\phi(x)]}\right]\frac{1}{f'(x)}$$

en (5.20), por lo que

$$g_\alpha(x) = \left[\frac{f(x)}{f(x) - 2f[\phi(x)]}\right]\frac{1}{(T_\alpha^a f)(x)}.$$

En consecuencia, la versión conformable del esquema de Ostrowski denotado por OeCO es

$$\psi_6(x) = a + \left((\phi_6(x)-a)^\alpha - \alpha\left[\frac{f(x)}{f(x) - 2f[\phi_6(x)]}\right]\frac{f[\phi_6(x)]}{(T_\alpha^a f)(x)}\right)^{1/\alpha}, \tag{5.21}$$

donde

$$\phi_6(x) = a + \left((x-a)^\alpha - \alpha\frac{f(x)}{(T_\alpha^a f)(x)}\right)^{1/\alpha}.$$

Finalmente, consideremos otro procedimiento óptimo de cuarto orden, el método de Chun [36]

$$\psi_7(x) = \phi_7(x) - \left[\frac{f(x) + 2f[\phi_7(x)]}{f(x)}\right]\frac{f[\phi_7(x)]}{f'(x)}, \tag{5.22}$$

donde

$$\phi_7(x) = x - \frac{f(x)}{f'(x)},$$

Aquí,

$$g(x) = \left[\frac{f(x) + 2f[\phi(x)]}{f(x)}\right] \frac{1}{f'(x)}$$

en (5.22), y por tanto

$$g_\alpha(x) = \left[\frac{f(x) + 2f[\phi(x)]}{f(x)}\right] \frac{1}{(T_\alpha^a f)(x)}.$$

Por consiguiente, la versión conformable del esquema de Chun denotado por CeCO es

$$\psi_8(x) = a + \left((\phi_8(x) - a)^\alpha - \alpha \left[\frac{f(x) + 2f[\phi_8(x)]}{f(x)}\right] \frac{f[\phi_8(x)]}{(T_\alpha^a f)(x)}\right)^{1/\alpha}, \tag{5.23}$$

donde

$$\phi_8(x) = a + \left((x - a)^\alpha - \alpha \frac{f(x)}{(T_\alpha^a f)(x)}\right)^{1/\alpha}.$$

En adelante, usaremos la notación $x = x_k$, $\phi(x) = y_k$ y $\psi(x) = x_{k+1}$.

5.2.2. Análisis de convergencia

En el siguiente resultado se establecen las condiciones que aseguran la convergencia de TeCO. Recordemos que utilizaremos derarrollos de Taylor clásicos, debido a su equivalencia con los desarrollos de Taylor conformables vista en la observación 3.3.3.

Teorema 5.2.1 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo el cero $\bar{x}$ de $f(x)$. Sea $(T_\alpha^a f)(x)$ la derivada fraccionaria conformable de $f(x)$ centrada en a, de orden α, para cualquier $\alpha \in (0, 1]$. Supongamos que $(T_\alpha^a f)(x)$ es continua y no nula en $\bar{x}$. Si una aproximación inicial x_0 es lo suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del procedimiento fraccionario conformable de tipo Traub (TeCO)*

$$x_{k+1} = a + \left((y_k - a)^\alpha - \alpha \frac{f(y_k)}{(T_\alpha^a f)(x_k)}\right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots,$$

donde

$$y_k = a + \left((x_k - a)^\alpha - \alpha \frac{f(x_k)}{(T_\alpha^a f)(x_k)}\right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots,$$

es al menos 3, y la ecuación del error es

$$e_{k+1} = \left(2C_2^2 + 2\frac{(1-\alpha)C_2}{\bar{x} - a} + \frac{1}{2}\frac{(1-\alpha)^2}{(\bar{x} - a)^2}\right) e_k^3 + O\left(e_k^4\right),$$

siendo $C_j = \dfrac{f^{(j)}(\bar{x})}{j!f'(\bar{x})}$, *para* $j \geq 2$, *tal que* $a < x_k$, $\forall k$.

Demostración. El desarrollo de Taylor de $f(x_k)$ en el entorno de $\bar{x}$, y considerando $x_k = \bar{x} + e_k$, es

$$f(x_k) = f'(\bar{x})\left[e_k + C_2 e_k^2 + C_3 e_k^3 + C_4 e_k^4\right] + O\left(e_k^5\right),$$

donde $C_j = \dfrac{f^{(j)}(\bar{x})}{j!f'(\bar{x})}$, para $j = 2, 3, 4, \ldots$

Usando el Teorema generalizado del binomio (2.52) y el coeficiente binomial fraccionario (2.53),

$$\begin{aligned}(T_\alpha^a f)(x_k) &= (x_k - a)^{1-\alpha} f'(x_k) = f'(\bar{x})\left[(\bar{x}-a)^{1-\alpha} + (\bar{x}-a)^{-\alpha}(1-\alpha+2(\bar{x}-a)C_2)e_k\right.\\ &+ \frac{1}{2}(\bar{x}-a)^{-1-\alpha}((\alpha-1)\alpha + 4(1-\alpha)(\bar{x}-a)C_2 + 6(\bar{x}-a)^2 C_3)e_k^2\\ &+ \frac{1}{6}(\bar{x}-a)^{-2-\alpha}(\alpha^3 - \alpha + 6(1-\alpha)\alpha(\bar{x}-a)C_2 - 18(1-\alpha)(\bar{x}-a)^2 C_3\\ &+ \left.24(a^3 - 3a^2\bar{x} + 3a\bar{x}^2 - \bar{x}^3)C_4)e_k^3\right] + O\left(e_k^4\right).\end{aligned}$$

El primer paso y_k es NeCO (3.9), cuyo orden de convergencia fue probado en el Corolario 3.3.1. El desarrollo de y_k hasta orden 3 se puede expresar como

$$y_k - \bar{x} = \left(C_2 + \frac{1-\alpha}{2(\bar{x}-a)}\right)e_k^2 + \left(2C_3 - 2C_2^2 + \frac{(\alpha-1)C_2}{\bar{x}-a} + \frac{(1-\alpha)(\alpha-2)}{3(\bar{x}-a)^2}\right)e_k^3 + O\left(e_k^4\right).$$

Así, el desarrollo de $f(y_k)$ es

$$f(y_k) = f'(\bar{x})\left[\left(C_2 + \frac{1-\alpha}{2(\bar{x}-a)}\right)e_k^2 + \left(2C_3 - 2C_2^2 + \frac{(\alpha-1)C_2}{\bar{x}-a} + \frac{(1-\alpha)(\alpha-2)}{3(\bar{x}-a)^2}\right)e_k^3\right] + O\left(e_k^4\right).$$

Entonces,

$$\begin{aligned}\alpha\frac{f(y_k)}{(T_\alpha^a f)(x_k)} &= \frac{\alpha}{2}(\bar{x}-a)^{\alpha-2}(1-\alpha-2(\bar{x}-a)C_2)e_k^2\\ &+ \frac{\alpha}{6}(\bar{x}-a)^{\alpha-3}((\alpha-1)(5\alpha-7) + 18(1-\alpha)(\bar{x}-a)C_2 + 24(\bar{x}-a)^2C_2^2 - 12(\bar{x}-a)^2C_3)e_k^3\\ &+ O\left(e_k^4\right).\end{aligned}$$

Usando de nuevo el Teorema generalizado del binomio, el desarrollo de $(y_k - a)^\alpha$ resulta

$$\begin{aligned}(y_k-a)^\alpha &= (\bar{x}-a)^\alpha + \frac{\alpha}{2}(\bar{x}-a)^{\alpha-2}(1-\alpha-2(\bar{x}-a)C_2)e_k^2\\ &+ \frac{\alpha}{3}(\bar{x}-a)^{\alpha-3}((\alpha-1)(\alpha-2) + 3(1-\alpha)(\bar{x}-a)C_2 + 6(\bar{x}-a)^2C_2^2 - 6(\bar{x}-a)^2C_3)e_k^3 + O\left(e_k^4\right),\end{aligned}$$

por lo tanto, usando el Teorema generalizado del binomio una vez más,

$$a + \left((y_k-a)^\alpha - \alpha\frac{f(y_k)}{(T_\alpha^a f)(x_k)}\right)^{1/\alpha} = \bar{x} + \left(2C_2^2 + 2\frac{(1-\alpha)C_2}{\bar{x}-a} + \frac{1}{2}\frac{(1-\alpha)^2}{(\bar{x}-a)^2}\right)e_k^3 + O\left(e_k^4\right).$$

Finalmente, sabiendo que $x_{k+1} = \bar{x} + e_{k+1}$,

$$e_{k+1} = \left(2C_2^2 + 2\frac{(1-\alpha)C_2}{\bar{x}-a} + \frac{1}{2}\frac{(1-\alpha)^2}{(\bar{x}-a)^2}\right) e_k^3 + O\left(e_k^4\right).$$

Esto completa la prueba. □

Observación 5.2.1 *Dado que la ecuación del error del método clásico de Traub es (ver (2.21))*

$$e_{k+1} = 2C_2^2 e_k^3 + O\left(e_k^4\right),$$

se comprueba la relación entre las constantes asintóticas del error vista en el Teorema 2.1.3.

En el siguiente resultado se plantean las condiciones que garantizan la convergencia de CKeCO.

Teorema 5.2.2 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo el cero $\bar{x}$ de $f(x)$. Sea $(T_\alpha^a f)(x)$ la derivada fraccionaria conformable de $f(x)$ centrada en a, de orden α, para cualquier $\alpha \in (0,1]$. Supongamos que $(T_\alpha^a f)(x)$ es continua y no nula en $\bar{x}$. Si una aproximación inicial x_0 es lo suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del esquema fraccionario conformable de tipo Chun-Kim (CKeCO)*

$$x_{k+1} = a + \left((x_k - a)^\alpha - \frac{\alpha}{2}\left[3 - \frac{(T_\alpha^a f)(y_k)}{(T_\alpha^a f)(x_k)}\right]\frac{f(x_k)}{(T_\alpha^a f)(x_k)}\right)^{1/\alpha},$$

donde

$$y_k = a + \left((x_k - a)^\alpha - \alpha\frac{f(x_k)}{(T_\alpha^a f)(x_k)}\right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots,$$

es al menos 3, y la ecuación del error es

$$e_{k+1} = \left(2C_2^2 + \frac{1}{2}C_3 + \frac{5}{2}\frac{(1-\alpha)C_2}{\bar{x}-a} + \frac{1}{12}\frac{(1-\alpha)(7-8\alpha)}{(\bar{x}-a)^2}\right) e_k^3 + O\left(e_k^4\right),$$

siendo $C_j = \dfrac{f^{(j)}(\bar{x})}{j!f'(\bar{x})}$, para $j \geq 2$, tal que $a < x_k$, $\forall k$.

Demostración. El desarrollo de Taylor de $f(x_k)$ en el entorno de $\bar{x}$, y considerando $x_k = \bar{x} + e_k$, es

$$f(x_k) = f'(\bar{x})\left[e_k + C_2 e_k^2 + C_3 e_k^3 + C_4 e_k^4\right] + O\left(e_k^5\right),$$

donde $C_j = \dfrac{f^{(j)}(\bar{x})}{j!f'(\bar{x})}$, para $j = 2, 3, 4, \ldots$

Usando el Teorema generalizado del binomio (2.52) y el coeficiente binomial fraccionario (2.53),

$$\begin{aligned}(T_\alpha^a f)(x_k) &= (x_k-a)^{1-\alpha}f'(x_k) = f'(\bar{x})\left[(\bar{x}-a)^{1-\alpha}+(\bar{x}-a)^{-\alpha}(1-\alpha+2(\bar{x}-a)C_2)e_k\right.\\ &+ \frac{1}{2}(\bar{x}-a)^{-1-\alpha}((\alpha-1)\alpha+4(1-\alpha)(\bar{x}-a)C_2+6(\bar{x}-a)^2C_3)e_k^2\\ &+ \frac{1}{6}(\bar{x}-a)^{-2-\alpha}(\alpha^3-\alpha+6(1-\alpha)\alpha(\bar{x}-a)C_2-18(1-\alpha)(\bar{x}-a)^2C_3\\ &+ \left.24(a^3-3a^2\bar{x}+3a\bar{x}^2-\bar{x}^3)C_4)e_k^3\right]+O\left(e_k^4\right).\end{aligned}$$

El primer paso y_k es NeCO (3.9), cuyo orden de convergencia fue probado en el Corolario 3.3.1. El desarrollo de y_k hasta orden 3 se puede expresar como

$$y_k-\bar{x}=\left(C_2+\frac{1-\alpha}{2(\bar{x}-a)}\right)e_k^2+\left(2C_3-2C_2^2+\frac{(\alpha-1)C_2}{\bar{x}-a}+\frac{(1-\alpha)(\alpha-2)}{3(\bar{x}-a)^2}\right)e_k^3+O\left(e_k^4\right).$$

Así, el desarrollo de $(T_\alpha^a f)(y_k)$ es

$$\begin{aligned}(T_\alpha^a f)(y_k) &= (y_k-a)^{1-\alpha}f'(y_k) = f'(\bar{x})\left[(\bar{x}-a)^{1-\alpha}+\frac{1}{2}(\bar{x}-a)^{-1-\alpha}(\alpha-1-2(\bar{x}-a)C_2)^2e_k^2\right.\\ &+ \frac{1}{3}(\bar{x}-a)^{-2-\alpha}(\alpha-1-2(\bar{x}-a)C_2)((\alpha-1)(\alpha-2)-3(\bar{x}-a)(C_2(\alpha-1-2(\bar{x}-a)C_2)\\ &+ \left.2(\bar{x}-a)C_3))e_k^3\right]+O\left(e_k^4\right).\end{aligned}$$

Entonces,

$$\begin{aligned}\frac{f(x_k)}{(T_\alpha^a f)(x_k)} &= (\bar{x}-a)^{\alpha-1}e_k-(\bar{x}-a)^{\alpha-2}(1-\alpha+(\bar{x}-a)C_2)e_k^2\\ &+ \frac{1}{2}(\bar{x}-a)^{\alpha-3}((\alpha-1)(\alpha-2)-2(\bar{x}-a)(C_2(\alpha-1-2(\bar{x}-a)C_2)+2(\bar{x}-a)C_3))e_k^3\\ &+ O\left(e_k^4\right),\end{aligned}$$

y

$$\begin{aligned}\frac{\alpha}{2}\left[3-\frac{(T_\alpha^a f)(y_k)}{(T_\alpha^a f)(x_k)}\right] &= \alpha\left[1+\left(C_2+\frac{1}{2}\frac{1-\alpha}{\bar{x}-a}\right)e_k\right.\\ &+ \left(\frac{\alpha(5-2\alpha)-3}{4(\bar{x}-a)^2}+\frac{2(\alpha-1)C_2}{\bar{x}-a}-3C_2^2+\frac{3}{2}C_3\right)e_k^2\\ &+ \frac{1}{12}\left(\frac{84(1-\alpha)C_2^2}{\bar{x}-a}+96C_2^3+2C_2\left(\frac{(\alpha-1)(17\alpha-22)}{(\bar{x}-a)^2}-48C_3\right)\right.\\ &+ \left.\left.\frac{(1-\alpha)(\alpha(6\alpha-17)+13-30(\bar{x}-a)^2C_3)}{(\bar{x}-a)^3}+24C_4\right)e_k^3\right]\\ &+ O\left(e_k^4\right).\end{aligned}$$

Por lo tanto,

$$\begin{aligned}\frac{\alpha}{2}\left[3-\frac{(T_\alpha^a f)(y_k)}{(T_\alpha^a f)(x_k)}\right]\frac{f(x_k)}{(T_\alpha^a f)(x_k)} &= \alpha\left[(\bar{x}-a)^{\alpha-1}e_k+\frac{\alpha(\alpha-1)}{2}(\bar{x}-a)^{\alpha-2}e_k^2\right.\\ &- \frac{1}{4}((\bar{x}-a)^{\alpha-3}((\alpha-1)(2\alpha-1)\\ &- 2(\bar{x}-a)(5(\alpha-1)C_2-4(\bar{x}-a)C_2^2-(\bar{x}-a)C_3)))e_k^3]\\ &+ O\left(e_k^4\right).\end{aligned}$$

Usando de nuevo el Teorema generalizado del binomio, el desarrollo de $(x_k-a)^\alpha$ resulta

$$(x_k-a)^\alpha=(\bar{x}-a)^\alpha+\alpha(\bar{x}-a)^{\alpha-1}e_k+\frac{1}{2}\alpha(\alpha-1)(\bar{x}-a)^{\alpha-2}e_k^2+\frac{1}{6}\alpha(\alpha-1)(\alpha-2)(\bar{x}-a)^{\alpha-3}e_k^3+O\left(e_k^4\right),$$

de modo que, usando el Teorema generalizado del binomio una vez más,

$$\begin{aligned}a+\left((x_k-a)^\alpha-\frac{\alpha}{2}\left[3-\frac{(T_\alpha^a f)(y_k)}{(T_\alpha^a f)(x_k)}\right]\frac{f(x_k)}{(T_\alpha^a f)(x_k)}\right)^{1/\alpha} &= \bar{x}+\left(2C_2^2+\frac{1}{2}C_3+\frac{5}{2}\frac{(1-\alpha)C_2}{\bar{x}-a}\right.\\ &+ \left.\frac{1}{12}\frac{(1-\alpha)(7-8\alpha)}{(\bar{x}-a)^2}\right)e_k^3+O\left(e_k^4\right).\end{aligned}$$

Finalmente, sabiendo que $x_{k+1}=\bar{x}+e_{k+1}$,

$$e_{k+1}=\left(2C_2^2+\frac{1}{2}C_3+\frac{5}{2}\frac{(1-\alpha)C_2}{\bar{x}-a}+\frac{1}{12}\frac{(1-\alpha)(7-8\alpha)}{(\bar{x}-a)^2}\right)e_k^3+O\left(e_k^4\right).$$

Esto completa la prueba. □

Observación 5.2.2 *Dado que la ecuación del error del procedimiento clásico de Chun-Kim es (ver (2.23))*

$$e_{k+1}=\left(2C_2^2+\frac{1}{2}C_3\right)e_k^3+O\left(e_k^4\right),$$

se comprueba la relación entre las constantes asintóticas del error vista en el Teorema 2.1.3.

En el siguiente resultado se establecen las condiciones que aseguran la convergencia de OeCO.

Teorema 5.2.3 *Sea $f:I\subseteq\mathbb{R}\to\mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo el cero $\bar{x}$ de $f(x)$. Sea $(T_\alpha^a f)(x)$ la derivada fraccionaria conformable de $f(x)$ centrada en a, de orden α, para cualquier $\alpha\in(0,1]$. Supongamos que $(T_\alpha^a f)(x)$ es continua y no nula en $\bar{x}$. Si una aproximación inicial x_0 es lo suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del método fraccionario conformable de tipo Ostrowski (OeCO)*

$$x_{k+1}=a+\left((y_k-a)^\alpha-\alpha\left[\frac{f(x_k)}{f(x_k)-2f(y_k)}\right]\frac{f(y_k)}{(T_\alpha^a f)(x_k)}\right)^{1/\alpha},\quad k=0,1,2,\ldots,$$

donde

$$y_k = a + \left((x_k - a)^\alpha - \alpha \frac{f(x_k)}{(T_\alpha^a f)(x_k)} \right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots,$$

es al menos 4, y la ecuación del error es

$$e_{k+1} = \left(C_2^3 - C_2 C_3 + \frac{1}{2}\frac{(1-\alpha)(C_2^2 - C_3)}{\bar{x} - a} + \frac{1}{12}\frac{(1-\alpha^2)^2 C_2}{(\bar{x} - a)^2} + \frac{1}{24}\frac{(1-\alpha)(1-\alpha^2)}{(\bar{x} - a)^3} \right) e_k^4 + O\left(e_k^5\right),$$

siendo $C_j = \dfrac{f^{(j)}(\bar{x})}{j! f'(\bar{x})}$, *para* $j \geq 2$, *tal que* $a < x_k$, $\forall k$.

Demostración. El desarrollo de Taylor de $f(x_k)$ en el entorno de $\bar{x}$, y considerando $x_k = \bar{x} + e_k$, es

$$f(x_k) = f'(\bar{x}) \left[e_k + C_2 e_k^2 + C_3 e_k^3 + C_4 e_k^4 + C_5 e_k^5 \right] + O\left(e_k^6\right),$$

donde $C_j = \dfrac{f^{(j)}(\bar{x})}{j! f'(\bar{x})}$, para $j = 2, 3, 4, \ldots$

Usando el Teorema generalizado del binomio (2.52) y el coeficiente binomial fraccionario (2.53),

$$\begin{aligned}
(T_\alpha^a f)(x_k) &= (x_k - a)^{1-\alpha} f'(x_k) = f'(\bar{x}) \left[(\bar{x} - a)^{1-\alpha} + (\bar{x} - a)^{-\alpha}(1 - \alpha + 2(\bar{x} - a)C_2) e_k \right. \\
&+ \frac{1}{2}(\bar{x} - a)^{-1-\alpha}((\alpha - 1)\alpha + 4(1-\alpha)(\bar{x} - a)C_2 + 6(\bar{x} - a)^2 C_3) e_k^2 \\
&+ \frac{1}{6}(\bar{x} - a)^{-2-\alpha}(\alpha^3 - \alpha + 6(1-\alpha)\alpha(\bar{x} - a)C_2 \\
&- 18(1-\alpha)(\bar{x} - a)^2 C_3 + 24(a^3 - 3a^2\bar{x} + 3a\bar{x}^2 - \bar{x}^3) C_4) e_k^3 \\
&+ \frac{1}{24}(\bar{x} - a)^{-\alpha} \left(\frac{\alpha(\alpha-1)(\alpha+1)(\alpha+2)}{(\bar{x} - a)^3} + \frac{8\alpha(1-\alpha)(1+\alpha)C_2}{(\bar{x} - a)^2} \right. \\
&\left. \left. + \frac{36\alpha(\alpha-1)C_3}{\bar{x} - a} + 96(1-\alpha)C_4 + 120(\bar{x} - a)C_5 \right) e_k^4 \right] + O\left(e_k^5\right).
\end{aligned}$$

El primer paso y_k es NeCO (3.9), cuyo orden de convergencia fue probado en el Corolario 3.3.1. El desarrollo de y_k hasta orden 4 se puede expresar como

$$\begin{aligned}
y_k - \bar{x} &= \left(C_2 + \frac{1-\alpha}{2(\bar{x} - a)} \right) e_k^2 + \left(2C_3 - 2C_2^2 + \frac{(\alpha-1)C_2}{\bar{x} - a} + \frac{(1-\alpha)(\alpha-2)}{3(\bar{x} - a)^2} \right) e_k^3 \\
&+ \left(3C_3 - 7C_2 C_3 + 4C_2^3 + \frac{(1-\alpha)(5C_2^2 - C_3)}{2(\bar{x} - a)} + \frac{(2\alpha^2 - 5\alpha + 3)C_2}{2(\bar{x} - a)^2} + \frac{(1-\alpha)(2\alpha^2 - 7\alpha + 7)}{8(\bar{x} - a)^3} \right) e_k^4 \\
&+ O\left(e_k^5\right).
\end{aligned}$$

Así, el desarrollo de $f(y_k)$ es

$$\begin{aligned}
f(y_k) &= f'(\bar{x}) \left[\left(C_2 + \frac{1-\alpha}{2(\bar{x} - a)} \right) e_k^2 + \left(2C_3 - 2C_2^2 + \frac{(\alpha-1)C_2}{\bar{x} - a} + \frac{(1-\alpha)(\alpha-2)}{3(\bar{x} - a)^2} \right) e_k^3 \right. \\
&+ \left(5C_2^3 - 7C_2 C_3 - \frac{(\alpha-1)(7C_2^2 + 4C_3)}{2(\bar{x} - a)} - \frac{(\alpha-1)(5\alpha-7)C_2}{4(\bar{x} - a)^2} - \frac{(\alpha-1)(2\alpha^2 - 7\alpha + 7)}{8(\bar{x} - a)^3} \right. \\
&\left. \left. - \frac{3(a^3 - 3a^2\bar{x} + 3a\bar{x}^2 - \bar{x}^3)C_4}{(\bar{x} - a)^3} \right) e_k^4 \right] + O\left(e_k^5\right).
\end{aligned}$$

Entonces,

$$\begin{aligned}\frac{f(y_k)}{(T_\alpha^a f)(x_k)} &= \frac{1}{2}(\bar{x}-a)^{\alpha-2}(1-\alpha-2(\bar{x}-a)C_2)e_k^2\\ &+ \frac{1}{6}(\bar{x}-a)^{\alpha-3}((\alpha-1)(5\alpha-7)+18(1-\alpha)(\bar{x}-a)C_2+24(\bar{x}-a)^2C_2^2-12(\bar{x}-a)^2C_3)e_k^3\\ &+ \frac{1}{24}(\bar{x}-a)^{\alpha-4}\left((\alpha-1)(5\alpha-7)(4\alpha-7)+2(53\alpha^2-126\alpha+73)(\bar{x}-a)C_2\right.\\ &+ (1-\alpha)(\bar{x}-a)^2(276C_2^2-132C_3)+(\bar{x}-a)^3(312C_2^3-336C_2C_3)\\ &- \left.72(a^3-3a^2\bar{x}+3a\bar{x}^2-\bar{x}^3)C_4\right)e_k^4+O\left(e_k^5\right),\end{aligned}$$

y

$$\begin{aligned}\left[\frac{f(x_k)}{f(x_k)-2f(y_k)}\right] &= 1+\left(2C_2+\frac{1-\alpha}{\bar{x}-a}\right)e_k+\left(4C_3-2C_2^2+\frac{(1-\alpha)C_2}{\bar{x}-a}+\frac{\alpha^2-1}{3(\bar{x}-a)^2}\right)e_k^2\\ &+ \left(-4C_2C_3-\frac{(\alpha-1)(3C_3-2C_2^2)}{\bar{x}-a}-\frac{(1-\alpha^2)C_2}{2(\bar{x}-a)^2}-\frac{(\alpha-1)(\alpha+1)(2\alpha+1)}{12(\bar{x}-a)^3}\right.\\ &- \left.\frac{6(a^3-3a^2\bar{x}+3a\bar{x}^2-\bar{x}^3)C_4}{(\bar{x}-a)^3}\right)e_k^3+O\left(e_k^4\right).\end{aligned}$$

Por lo tanto,

$$\begin{aligned}\alpha\left[\frac{f(x_k)}{f(x_k)-2f(y_k)}\right]\frac{f(y_k)}{(T_\alpha^a f)(x_k)} &= \alpha\left[\frac{1}{2}(\bar{x}-a)^{\alpha-2}(1-\alpha-2(\bar{x}-a)C_2)e_k^2-\frac{1}{3}(\bar{x}-a)^{\alpha-3}((\alpha-1)(\alpha-2)\right.\\ &+ 3(1-\alpha)(\bar{x}-a)C_2+6(\bar{x}-a)^2(C_2^2-C_3))e_k^3\\ &+ \frac{1}{24}(\bar{x}-a)^{\alpha-4}\left((\alpha-1)(4\alpha^2-15\alpha+17)+36(\alpha-1)(\bar{x}-a)^2(C_2^2-C_3)\right.\\ &- 72(\bar{x}-a)^3C_2^3+2(\alpha-1)(7\alpha-11)(\bar{x}-a)C_2+144(\bar{x}-a)^3C_2C_3\\ &+ \left.\left.72(a^3-3a^2\bar{x}+3a\bar{x}^2-\bar{x}^3)C_4\right)e_k^4\right]+O\left(e_k^5\right).\end{aligned}$$

Usando de nuevo el Teorema generalizado del binomio, el desarrollo de $(y_k-a)^\alpha$ resulta

$$\begin{aligned}(y_k-a)^\alpha &= (\bar{x}-a)^\alpha+\frac{\alpha}{2}(\bar{x}-a)^{\alpha-2}(1-\alpha-2(\bar{x}-a)C_2)e_k^2\\ &+ \frac{\alpha}{3}(\bar{x}-a)^{\alpha-3}((\alpha-1)(\alpha-2)+3(1-\alpha)(\bar{x}-a)C_2+6(\bar{x}-a)^2C_2^2-6(\bar{x}-a)^2C_3)e_k^3\\ &- \frac{\alpha}{8}(\bar{x}-a)^{\alpha-4}((\alpha-1)(\alpha-2)(\alpha-3)+16(\alpha-1)(\bar{x}-a)^2C_2^2-32(\bar{x}-a)^3C_2^3\\ &+ 16(1-\alpha)(\bar{x}-a)^2C_3-4(\alpha^2-3\alpha+2)(\bar{x}-a)C_2+56(\bar{x}-a)^3C_2C_3\\ &+ 24(a^3-3a^2\bar{x}+3a\bar{x}^2-\bar{x}^3)C_4)e_k^4+O\left(e_k^5\right),\end{aligned}$$

de modo que, usando el Teorema generalizado del binomio una vez más,

$$\begin{aligned}x_{k+1} &= a+\left((y_k-a)^\alpha-\alpha\left[\frac{f(x_k)}{f(x_k)-2f(y_k)}\right]\frac{f(y_k)}{(T_\alpha^a f)(x_k)}\right)^{1/\alpha}\\ &= \bar{x}+\left(C_2^3-C_2C_3+\frac{1}{2}\frac{(1-\alpha)(C_2^2-C_3)}{\bar{x}-a}+\frac{1}{12}\frac{(1-\alpha^2)^2C_2}{(\bar{x}-a)^2}+\frac{1}{24}\frac{(1-\alpha)(1-\alpha^2)}{(\bar{x}-a)^3}\right)e_k^4+O\left(e_k^5\right).\end{aligned}$$

Finalmente, sabiendo que $x_{k+1} = \bar{x} + e_{k+1}$,

$$e_{k+1} = \left(C_2^3 - C_2 C_3 + \frac{1}{2}\frac{(1-\alpha)(C_2^2 - C_3)}{\bar{x}-a} + \frac{1}{12}\frac{(1-\alpha^2)^2 C_2}{(\bar{x}-a)^2} + \frac{1}{24}\frac{(1-\alpha)(1-\alpha^2)}{(\bar{x}-a)^3} \right) e_k^4 + O\left(e_k^5\right).$$

Esto completa la prueba. □

Observación 5.2.3 *Dado que la ecuación del error del esquema clásico de Ostrowski es (ver (2.26))*

$$e_{k+1} = \left(C_2^3 - C_2 C_3\right) e_k^4 + O\left(e_k^5\right),$$

se comprueba la relación entre las constantes asintóticas del error vista en el Teorema 2.1.3.

Finalmente, en el siguiente resultado se plantean las condiciones que garantizan la convergencia de CeCO.

Teorema 5.2.4 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo el cero $\bar{x}$ de $f(x)$. Sea $(T_\alpha^a f)(x)$ la derivada fraccionaria conformable de $f(x)$ centrada en a, de orden α, para cualquier $\alpha \in (0,1]$. Supongamos que $(T_\alpha^a f)(x)$ es continua y no nula en $\bar{x}$. Si una aproximación inicial x_0 es lo suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del método fraccionario conformable de tipo Chun (CeCO)*

$$x_{k+1} = a + \left((y_k - a)^\alpha - \alpha \left[\frac{f(x_k) + 2f(y_k)}{f(x_k)} \right] \frac{f(y_k)}{(T_\alpha^a f)(x_k)} \right)^{1/\alpha},$$

donde

$$y_k = a + \left((x_k - a)^\alpha - \alpha \frac{f(x_k)}{(T_\alpha^a f)(x_k)} \right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots,$$

es al menos 4, y la ecuación del error es

$$\begin{aligned} e_{k+1} &= \left(5C_2^3 - C_2 C_3 + \frac{(1-\alpha)(13C_2^2 - C_3)}{2(\bar{x}-a)} + \frac{(24(1-\alpha)^2 + (1-\alpha)(13-11\alpha))C_2}{12(\bar{x}-a)^2} \right. \\ &+ \left. \frac{(1-\alpha)^2(13-11\alpha)}{24(\bar{x}-a)^3} \right) e_k^4 + O\left(e_k^5\right), \end{aligned}$$

siendo $C_j = \dfrac{f^{(j)}(\bar{x})}{j! f'(\bar{x})}$, para $j \geq 2$, tal que $a < x_k$, $\forall k$.

Demostración. El desarrollo de Taylor de $f(x_k)$ en el entorno de $\bar{x}$, y considerando $x_k = \bar{x} + e_k$, es

$$f(x_k) = f'(\bar{x}) \left[e_k + C_2 e_k^2 + C_3 e_k^3 + C_4 e_k^4 + C_5 e_k^5 \right] + O\left(e_k^6\right),$$

donde $C_j = \dfrac{f^{(j)}(\bar{x})}{j! f'(\bar{x})}$, para $j = 2, 3, 4, \ldots$

Usando el Teorema generalizado del binomio (2.52) y el coeficiente binomial fraccionario (2.53),

$$
\begin{aligned}
(T_\alpha^a f)(x_k) &= (x_k-a)^{1-\alpha}f'(x_k) = f'(\bar{x})\left[(\bar{x}-a)^{1-\alpha}+(\bar{x}-a)^{-\alpha}(1-\alpha+2(\bar{x}-a)C_2)e_k\right.\\
&+ \frac{1}{2}(\bar{x}-a)^{-1-\alpha}((\alpha-1)\alpha+4(1-\alpha)(\bar{x}-a)C_2+6(\bar{x}-a)^2C_3)e_k^2\\
&+ \frac{1}{6}(\bar{x}-a)^{-2-\alpha}(\alpha^3-\alpha+6(1-\alpha)\alpha(\bar{x}-a)C_2\\
&- 18(1-\alpha)(\bar{x}-a)^2C_3+24(a^3-3a^2\bar{x}+3a\bar{x}^2-\bar{x}^3)C_4)e_k^3\\
&+ \frac{1}{24}(\bar{x}-a)^{-\alpha}\left(\frac{\alpha(\alpha-1)(\alpha+1)(\alpha+2)}{(\bar{x}-a)^3}+\frac{8\alpha(1-\alpha)(1+\alpha)C_2}{(\bar{x}-a)^2}\right.\\
&+ \left.\left.\frac{36\alpha(\alpha-1)C_3}{\bar{x}-a}+96(1-\alpha)C_4+120(\bar{x}-a)C_5\right)e_k^4\right]+O\left(e_k^5\right).
\end{aligned}
$$

El primer paso y_k es NeCO (3.9), cuyo orden de convergencia fue probado en el Corolario 3.3.1. El desarrollo de y_k hasta orden 4 se puede expresar como

$$
\begin{aligned}
y_k-\bar{x} &= \left(C_2+\frac{1-\alpha}{2(\bar{x}-a)}\right)e_k^2+\left(2C_3-2C_2^2+\frac{(\alpha-1)C_2}{\bar{x}-a}+\frac{(1-\alpha)(\alpha-2)}{3(\bar{x}-a)^2}\right)e_k^3\\
&+ \left(3C_3-7C_2C_3+4C_2^3+\frac{(1-\alpha)(5C_2^2-C_3)}{2(\bar{x}-a)}+\frac{(2\alpha^2-5\alpha+3)C_2}{2(\bar{x}-a)^2}+\frac{(1-\alpha)(2\alpha^2-7\alpha+7)}{8(\bar{x}-a)^3}\right)e_k^4\\
&+ O\left(e_k^5\right).
\end{aligned}
$$

Así, el desarrollo de $f(y_k)$ es

$$
\begin{aligned}
f(y_k) &= f'(\bar{x})\left[\left(C_2+\frac{1-\alpha}{2(\bar{x}-a)}\right)e_k^2+\left(2C_3-2C_2^2+\frac{(\alpha-1)C_2}{\bar{x}-a}+\frac{(1-\alpha)(\alpha-2)}{3(\bar{x}-a)^2}\right)e_k^3\right.\\
&+ \left(5C_2^3-7C_2C_3-\frac{(\alpha-1)(7C_2^2+4C_3)}{2(\bar{x}-a)}-\frac{(\alpha-1)(5\alpha-7)C_2}{4(\bar{x}-a)^2}-\frac{(\alpha-1)(2\alpha^2-7\alpha+7)}{8(\bar{x}-a)^3}\right.\\
&- \left.\left.\frac{3(a^3-3a^2\bar{x}+3a\bar{x}^2-\bar{x}^3)C_4}{(\bar{x}-a)^3}\right)e_k^4\right]+O\left(e_k^5\right).
\end{aligned}
$$

Entonces,

$$
\begin{aligned}
\frac{f(y_k)}{(T_\alpha^a f)(x_k)} &= \frac{1}{2}(\bar{x}-a)^{\alpha-2}(1-\alpha-2(\bar{x}-a)C_2)e_k^2\\
&+ \frac{1}{6}(\bar{x}-a)^{\alpha-3}((\alpha-1)(5\alpha-7)+18(1-\alpha)(\bar{x}-a)C_2+24(\bar{x}-a)^2C_2^2-12(\bar{x}-a)^2C_3)e_k^3\\
&+ \frac{1}{24}(\bar{x}-a)^{\alpha-4}\left((\alpha-1)(5\alpha-7)(4\alpha-7)+2(53\alpha^2-126\alpha+73)(\bar{x}-a)C_2\right.\\
&+ (1-\alpha)(\bar{x}-a)^2(276C_2^2-132C_3)+(\bar{x}-a)^3(312C_2^3-336C_2C_3)\\
&- \left.72(a^3-3a^2\bar{x}+3a\bar{x}^2-\bar{x}^3)C_4\right)e_k^4+O\left(e_k^5\right),
\end{aligned}
$$

y

$$\begin{aligned}\left[\frac{f(x_k)+2f(y_k)}{f(x_k)}\right] &= 1+\left(2C_2+\frac{1-\alpha}{\bar{x}-a}\right)e_k+\left(4C_3-6C_2^2-\frac{3(1-\alpha)C_2}{\bar{x}-a}-\frac{2(\alpha-1)(\alpha-2)}{3(\bar{x}-a)^2}\right)e_k^2\\ &+ \left(16C_2^3-20C_2C_3+6C_4+\frac{(\alpha-1)(5C_3-10C_2^2)}{\bar{x}-a}-\frac{(\alpha-1)C_2}{6(\bar{x}-a)^2}\right.\\ &- \left.\frac{(\alpha-1)(7+\alpha(2\alpha-7))}{4(\bar{x}-a)^3}\right)e_k^3+O\left(e_k^4\right).\end{aligned}$$

Por lo tanto,

$$\begin{aligned}\alpha\left[\frac{f(x_k)+2f(y_k)}{f(x_k)}\right]\frac{f(y_k)}{(T_\alpha^a f)(x_k)} &= \alpha\left[\frac{1}{2}(\bar{x}-a)^{\alpha-2}(1-\alpha-2(\bar{x}-a)C_2)e_k^2-\frac{1}{3}(\bar{x}-a)^{\alpha-3}((\alpha-1)(\alpha-2)\right.\\ &+ 3(1-\alpha)(\bar{x}-a)C_2+6(\bar{x}-a)^2(C_2^2-C_3))e_k^3\\ &+ \frac{1}{24}(\bar{x}-a)^{\alpha-4}\left(8\alpha^3-17\alpha^2+4\alpha+5+(\alpha-1)(\bar{x}-a)^2(108C_2^2-36C_3)\right.\\ &- 24(\bar{x}-a)^3C_2^3-2(29\alpha^2-54\alpha+25)(\bar{x}-a)C_2-144(\bar{x}-a)^3C_2C_3\\ &- \left.\left.72(a^3-3a^2\bar{x}+3a\bar{x}^2-\bar{x}^3)C_4\right)e_k^4\right]+O\left(e_k^5\right).\end{aligned}$$

Usando de nuevo el Teorema generalizado del binomio, el desarrollo de $(y_k-a)^\alpha$ resulta

$$\begin{aligned}(y_k-a)^\alpha &= (\bar{x}-a)^\alpha+\frac{\alpha}{2}(\bar{x}-a)^{\alpha-2}(1-\alpha-2(\bar{x}-a)C_2)e_k^2\\ &+ \frac{\alpha}{3}(\bar{x}-a)^{\alpha-3}((\alpha-1)(\alpha-2)+3(1-\alpha)(\bar{x}-a)C_2+6(\bar{x}-a)^2C_2^2-6(\bar{x}-a)^2C_3)e_k^3\\ &- \frac{\alpha}{8}(\bar{x}-a)^{\alpha-4}((\alpha-1)(\alpha-2)(\alpha-3)+16(\alpha-1)(\bar{x}-a)^2C_2^2-32(\bar{x}-a)^3C_2^3\\ &+ 16(1-\alpha)(\bar{x}-a)^2C_3-4(\alpha^2-3\alpha+2)(\bar{x}-a)C_2+56(\bar{x}-a)^3C_2C_3\\ &+ 24(a^3-3a^2\bar{x}+3a\bar{x}^2-\bar{x}^3)C_4)e_k^4+O\left(e_k^5\right),\end{aligned}$$

de modo que, usando el Teorema generalizado del binomio una vez más,

$$\begin{aligned}x_{k+1} &= a+\left((y_k-a)^\alpha-\alpha\left[\frac{f(x_k)+2f(y_k)}{f(x_k)}\right]\frac{f(y_k)}{(T_\alpha^a f)(x_k)}\right)^{1/\alpha}=\bar{x}+\left(5C_2^3-C_2C_3\right.\\ &+ \left.\frac{(1-\alpha)(13C_2^2-C_3)}{2(\bar{x}-a)}+\frac{(24(1-\alpha)^2+(1-\alpha)(13-11\alpha))C_2}{12(\bar{x}-a)^2}+\frac{(1-\alpha)^2(13-11\alpha)}{24(\bar{x}-a)^3}\right)e_k^4+O\left(e_k^5\right).\end{aligned}$$

Finalmente, sabiendo que $x_{k+1}=\bar{x}+e_{k+1}$,

$$\begin{aligned}e_{k+1} &= \left(5C_2^3-C_2C_3+\frac{(1-\alpha)(13C_2^2-C_3)}{2(\bar{x}-a)}+\frac{(24(1-\alpha)^2+(1-\alpha)(13-11\alpha))C_2}{12(\bar{x}-a)^2}\right.\\ &+ \left.\frac{(1-\alpha)^2(13-11\alpha)}{24(\bar{x}-a)^3}\right)e_k^4+O\left(e_k^5\right).\end{aligned}$$

Esto completa la prueba. □

Observación 5.2.4 *Dado que la ecuación del error del esquema clásico de Chun es (ver (2.28))*

$$e_{k+1} = \left(5C_2^3 - C_2C_3\right) e_k^4 + O\left(e_k^5\right),$$

se comprueba la relación entre las constantes asintóticas del error vista en el Teorema 2.1.3.

Observación 5.2.5 *Los procedimientos OeCO y CeCO son los primeros métodos fraccionarios multipunto óptimos, según la conjetura de Kung y Traub [26].*

Como hemos podido ver hasta ahora, el orden de convergencia teórico de los esquemas clásicos se conserva en la versión conformable de tales procedimientos. En la siguiente sección, realizamos diversas pruebas numéricas con algunas funciones no lineales, y estudiamos la estabilidad de estos esquemas.

5.3. Pruebas numéricas

Para obtener los resultados en esta sección, hemos utilizado Matlab R2020a con aritmética de precisión doble, $|f(x_{k+1})| < 10^{-8}$ o $|x_{k+1} - x_k| < 10^{-8}$ como criterios de parada, y un máximo de 500 iteraciones. Para el cálculo de la función Gamma hemos utilizado el programa en [27] (ver fichero MATLAB en Anexo A.1). Para la función de Mittag-Leffler utilizamos el programa proporcionado por Igor Podlubny en MathWorks (ver fichero MATLAB en Anexo A.2). La función *Gamma.m* del Anexo A.1 se calcula con 15 dígitos de precisión a lo largo del eje real y 13 en otras partes de $\mathbb{C}$, y la función *mlf.m* del Anexo A.2 para calcular la función de Mittag-Lefler tiene 9 cifras significativas de precisión. La función Gamma y la de Mittag-Leffler sólo son calculadas para los métodos TeCA y TeRL, los cuales usan derivadas de Caputo y de Riemann-Liouville, respectivamente; la derivada conformable no necesita que se evalúen estas funciones. También usamos el Orden de Convergencia Computacional Aproximado (ACOC)

$$ACOC = \rho = \frac{\ln(|x_{k+1} - x_k|/|x_k - x_{k-1}|)}{\ln(|x_k - x_{k-1}|/|x_{k-1} - x_{k-2}|)}, \quad k = 2, 3, 4, \ldots,$$

definido en [17], para comprobar que el orden de convergencia teórico también se conserva en la práctica.

Vamos a comprobar el funcionamiento de los procedimientos TeCA, TeRL, TeCO, CKeCO, OeCO y CeCO sobre algunas funciones no lineales (ver ficheros MATLAB en Anexos A.24 a A.29); comparamos cada método con su versión clásica (cuando $\alpha = 1$). Para los esquemas TeCO, CKeCO, OeCO y CeCO, con derivadas conformables, fijamos $a = -10$ para asegurar que $a < x_k$, $\forall k$, mientras que para los procedimientos TeCA y TeRL (con derivadas de Caputo y Riemann-Liouville, respectivamente) $a = 0$, ya que sólo así se puede demostrar la convergencia en la teoría.

En cada tabla mostramos los resultados obtenidos para cada función de prueba con una misma estimación inicial x_0 y $\alpha \in (0, 1]$, con el uso de los seis métodos diseñados en este capítulo.

Nuestra primera función es $f_1(x) = -12.84x^6 - 25.6x^5 + 16.55x^4 - 2.21x^3 + 26.71x^2 - 4.29x - 15.21$, con raíces reales y complejas $\bar{x}_1 = 0.82366 + 0.24769i$, $\bar{x}_2 = 0.82366 - 0.24769i$, $\bar{x}_3 = -2.62297$, $\bar{x}_4 = -0.584$, $\bar{x}_5 = -0.21705 + 0.99911i$ y $\bar{x}_6 = -0.21705 - 0.99911i$. Las derivadas necesarias para esta función son:

$$\begin{aligned}
{}_cD_{0^+}^{\alpha} f_1(x) &= -12.84\frac{\Gamma(7)}{\Gamma(7-\alpha)}x^{6-\alpha} - 25.6\frac{\Gamma(6)}{\Gamma(6-\alpha)}x^{5-\alpha} + 16.55\frac{\Gamma(5)}{\Gamma(5-\alpha)}x^{4-\alpha} \\
&\quad -2.21\frac{\Gamma(4)}{\Gamma(4-\alpha)}x^{3-\alpha} + 26.71\frac{\Gamma(3)}{\Gamma(3-\alpha)}x^{2-\alpha} - 4.29\frac{\Gamma(2)}{\Gamma(2-\alpha)}x^{1-\alpha}, \\
D_{0^+}^{\alpha} f_1(x) &= {}_cD_{0^+}^{\alpha} f_1(x) - 15.21\frac{1}{\Gamma(1-\alpha)}x^{-\alpha}, \\
(T_{\alpha}^{a} f_1)(x) &= (x-a)^{1-\alpha} f_1'(x) = (x-a)^{1-\alpha}(-77.04x^5 - 128x^4 + 66.2x^3 - 6.63x^2 + 53.42x - 4.29).
\end{aligned}$$

	Método TeCA					Método TeRL				
α	$\bar{x}$	$\lvert f_1(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_1(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_3$	4.60×10^{-13}	4.49×10^{-7}	115	2.89	$\bar{x}_3$	4.60×10^{-13}	4.49×10^{-7}	115	2.89
0.9	$\bar{x}_4$	1.15×10^{-6}	7.77×10^{-9}	49	0.96	$\bar{x}_4$	2.81×10^{-6}	8.63×10^{-9}	64	0.95
0.8	$\bar{x}_4$	1.06×10^{-5}	9.45×10^{-9}	91	0.99	$\bar{x}_4$	9.70×10^{-6}	9.83×10^{-9}	125	0.98
0.7	$\bar{x}_6$	1.38×10^{-4}	9.93×10^{-9}	225	1.00	-	-	-	>500	-
0.6	-	-	-	>500	-	-	-	-	>500	-
0.5	-	-	-	>500	-	-	-	-	>500	-
0.4	-	-	-	>500	-	-	-	-	>500	-
0.3	-	-	-	>500	-	-	-	-	>500	-
0.2	-	-	-	>500	-	-	-	-	>500	-
0.1	-	-	-	>500	-	-	-	-	>500	-

	Método TeCO					Método CKeCO				
α	$\bar{x}$	$\lvert f_1(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_1(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_3$	6.18×10^{-13}	4.49×10^{-7}	115	2.89	-	-	-	>500	-
0.9	$\bar{x}_4$	4.04×10^{-9}	2.43×10^{-4}	69	2.80	-	-	-	>500	-
0.8	$\bar{x}_4$	5.30×10^{-11}	5.73×10^{-5}	61	2.83	$\bar{x}_4$	9.95×10^{-14}	9.91×10^{-10}	190	3.00
0.7	$\bar{x}_2$	1.16×10^{-13}	3.57×10^{-6}	329	0.00	-	-	-	>500	-
0.6	$\bar{x}_4$	1.07×10^{-14}	1.12×10^{-6}	119	2.90	-	-	-	>500	-
0.5	$\bar{x}_4$	3.24×10^{-10}	1.05×10^{-4}	213	2.82	-	-	-	>500	-
0.4	$\bar{x}_4$	1.17×10^{-13}	7.31×10^{-9}	104	2.95	$\bar{x}_5$	1.41×10^{-11}	1.88×10^{-5}	484	2.80
0.3	-	-	-	>500	-	$\bar{x}_4$	2.11×10^{-13}	1.11×10^{-5}	490	0.00
0.2	-	-	-	>500	-	-	-	-	>500	-
0.1	-	-	-	>500	-	-	-	-	-	-

	Método OeCO					Método CeCO				
α	$\bar{x}$	$\lvert f_1(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_1(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_4$	1.07×10^{-14}	2.77×10^{-5}	3	2.41	$\bar{x}_3$	6.18×10^{-13}	4.76×10^{-9}	47	3.73
0.9	$\bar{x}_4$	9.95×10^{-14}	1.97×10^{-5}	3	2.43	$\bar{x}_3$	3.11×10^{-10}	2.76×10^{-4}	75	3.21
0.8	$\bar{x}_4$	9.95×10^{-14}	1.36×10^{-5}	3	2.44	$\bar{x}_3$	6.18×10^{-13}	9.77×10^{-5}	66	3.28
0.7	$\bar{x}_4$	1.17×10^{-13}	9.14×10^{-6}	3	2.46	$\bar{x}_3$	2.26×10^{-12}	1.89×10^{-10}	87	3.81
0.6	$\bar{x}_4$	9.95×10^{-14}	5.93×10^{-6}	3	2.48	$\bar{x}_3$	1.81×10^{-10}	2.42×10^{-4}	108	3.22
0.5	$\bar{x}_4$	1.17×10^{-13}	3.68×10^{-6}	3	2.49	$\bar{x}_3$	4.92×10^{-11}	1.76×10^{-4}	87	3.24
0.4	$\bar{x}_4$	1.17×10^{-13}	2.16×10^{-6}	3	2.51	$\bar{x}_4$	1.17×10^{-13}	1.08×10^{-7}	212	3.68
0.3	$\bar{x}_4$	1.07×10^{-14}	1.18×10^{-6}	3	2.53	$\bar{x}_4$	1.07×10^{-14}	1.75×10^{-7}	53	3.67
0.2	$\bar{x}_4$	2.24×10^{-13}	5.90×10^{-7}	3	2.55	$\bar{x}_4$	2.24×10^{-13}	7.13×10^{-7}	45	3.61
0.1	$\bar{x}_4$	4.21×10^{-13}	2.58×10^{-7}	3	2.57	-	-	-	-	-

Tabla 5.1: Resultados para $f_1(x)$, con estimación inicial $x_0 = 1$

En la Tabla 5.1 podemos ver que los esquemas TeCA, TeRL y TeCO pueden requerir menos iteraciones que con el procedimiento clásico de Traub (cuando $\alpha = 1$) en cada caso, ρ es alrededor de 3 en la mayoría de los casos cuando se usa TeCO, mientras que es lineal con TeCA y TeRL; no se muestran resultados cuando no se encuentra solución en 500 iteraciones. Observamos que el método clásico de Chun-Kim no encuentra ninguna solución, mientras que CKeCO converge para algunos valores de α, y el orden de convergencia computacional es alrededor de 3 en la mayoría de los casos. No se muestran resultados cuando $\alpha = 0.1$ porque CKeCO converge a un punto que no es solución de $f_1(x)$; esto sucede porque a pesar de que $|x_{k+1}-x_k| < 10^{-8}$, es decir, cumple con el criterio de parada, $|f(x_{k+1})| \gg 10^{-8}$. Podemos notar que OeCO presenta un comportamiento similar al del esquema clásico de Ostrowski, mientras que ρ no es el esperado para $\alpha \in (0, 1]$. Vemos que CeCO puede requerir menos iteraciones que el procedimiento clásico de Chun, y el orden de convergencia computacional tiende a ser 4. De nuevo, no se muestran resultados cuando $\alpha = 0.1$ porque CeCO converge a un punto que no es solución de $f_1(x)$; esto también sucede porque aunque $|x_{k+1} - x_k| < 10^{-8}$, es decir, cumple con el criterio de parada, $|f(x_{k+1})| \gg 10^{-8}$. En general, podemos observar que se pueden obtener diferentes soluciones al variar el valor de α para una misma estimación inicial x_0, y que es posible conseguir raíces complejas partiendo de estimaciones iniciales reales.

	Método TeCA					Método TeRL				
α	$\bar{x}$	$\lvert f_2(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_2(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_1$	7.16×10^{-15}	6.43×10^{-7}	7	3.18	$\bar{x}_1$	7.16×10^{-15}	6.43×10^{-7}	7	3.18
0.9	$\bar{x}_1$	1.90×10^{-7}	9.52×10^{-9}	31	0.97	$\bar{x}_2$	1.70×10^{-7}	9.04×10^{-9}	27	0.96
0.8	$\bar{x}_1$	1.86×10^{-6}	9.98×10^{-9}	125	0.99	$\bar{x}_2$	1.58×10^{-6}	9.67×10^{-9}	105	0.99
0.7	-	-	-	>500	-	$\bar{x}_2$	1.16×10^{-5}	9.99×10^{-9}	435	1.00
0.6	-	-	-	>500	-	-	-	-	>500	-
0.5	-	-	-	>500	-	-	-	-	>500	-
0.4	-	-	-	>500	-	-	-	-	>500	-
0.3	-	-	-	>500	-	-	-	-	>500	-
0.2	-	-	-	>500	-	-	-	-	>500	-
0.1	-	-	-	>500	-	-	-	-	>500	-

	Método TeCo					Método CKeCO				
α	$\bar{x}$	$\lvert f_2(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_2(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_1$	7.16×10^{-15}	6.43×10^{-7}	7	3.18	$\bar{x}_1$	2.22×10^{-15}	2.34×10^{-8}	7	3.09
0.9	$\bar{x}_1$	2.42×10^{-14}	6.19×10^{-5}	11	3.46	$\bar{x}_1$	1.14×10^{-11}	4.94×10^{-4}	8	2.76
0.8	$\bar{x}_2$	9.57×10^{-15}	2.49×10^{-7}	10	2.90	$\bar{x}_2$	8.40×10^{-9}	0.005	7	3.87
0.7	$\bar{x}_2$	4.72×10^{-15}	4.02×10^{-6}	7	2.75	$\bar{x}_2$	3.97×10^{-15}	2.37×10^{-5}	8	2.76
0.6	$\bar{x}_2$	1.45×10^{-13}	1.37×10^{-4}	8	2.95	$\bar{x}_2$	3.97×10^{-15}	5.55×10^{-6}	9	3.22
0.5	$\bar{x}_2$	1.78×10^{-15}	6.41×10^{-8}	7	2.86	$\bar{x}_2$	3.66×10^{-15}	2.49×10^{-5}	6	2.94
0.4	$\bar{x}_2$	1.26×10^{-14}	3.59×10^{-5}	6	2.97	$\bar{x}_2$	9.27×10^{-15}	4.86×10^{-6}	5	2.97
0.3	$\bar{x}_2$	5.33×10^{-15}	3.10×10^{-7}	5	3.07	$\bar{x}_2$	1.65×10^{-14}	1.51×10^{-7}	7	2.96
0.2	$\bar{x}_2$	1.65×10^{-14}	2.47×10^{-6}	5	3.03	$\bar{x}_2$	1.65×10^{-14}	5.19×10^{-9}	19	3.00
0.1	$\bar{x}_1$	2.14×10^{-14}	4.53×10^{-9}	30	2.98	$\bar{x}_2$	2.31×10^{-11}	6.61×10^{-4}	9	2.97

	Método OeCO					Método CeCO				
α	$\bar{x}$	$\lvert f_2(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_2(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_1$	2.04×10^{-9}	0.026	4	5.51	$\bar{x}_1$	2.89×10^{-13}	0.002	7	2.74
0.9	$\bar{x}_1$	4.29×10^{-14}	8.34×10^{-9}	5	3.72	$\bar{x}_1$	1.61×10^{-14}	2.34×10^{-4}	8	3.77
0.8	$\bar{x}_1$	1.39×10^{-14}	2.05×10^{-7}	5	3.65	$\bar{x}_1$	1.90×10^{-14}	2.93×10^{-7}	13	4.31
0.7	$\bar{x}_1$	3.31×10^{-14}	4.74×10^{-6}	5	3.57	$\bar{x}_2$	3.31×10^{-9}	0.0227	8	3.06
0.6	$\bar{x}_1$	3.30×10^{-14}	9.49×10^{-5}	5	3.52	$\bar{x}_2$	6.03×10^{-11}	0.0087	9	2.85
0.5	$\bar{x}_1$	2.25×10^{-14}	0.0015	5	3.59	$\bar{x}_2$	1.99×10^{-15}	4.90×10^{-8}	10	3.80
0.4	$\bar{x}_1$	1.00×10^{-9}	0.0206	5	4.07	$\bar{x}_1$	6.52×10^{-10}	0.0118	12	2.67
0.3	$\bar{x}_1$	2.22×10^{-15}	5.59×10^{-6}	6	3.43	$\bar{x}_1$	1.11×10^{-14}	5.65×10^{-4}	12	4.19
0.2	$\bar{x}_1$	1.38×10^{-9}	0.0219	6	3.47	$\bar{x}_2$	4.80×10^{-14}	5.66×10^{-6}	95	3.89
0.1	$\bar{x}_1$	1.90×10^{-14}	1.94×10^{-8}	8	4.27	$\bar{x}_2$	8.42×10^{-14}	5.83×10^{-7}	383	4.04

Tabla 5.2: Resultados para $f_2(x)$, con estimación inicial $x_0 = 0.5$

La segunda función es $f_2(x) = ix^{1.8} - x^{0.9} - 16$, con raíces complejas $\bar{x}_1 = 2.90807 - 4.24908i$ y $\bar{x}_2 = -3.85126 + 1.74602i$. Las derivadas necesarias para esta función son:

$$\begin{aligned} {}_cD^{\alpha}_{0+} f_2(x) &= i\frac{\Gamma(2.8)}{\Gamma(2.8-\alpha)}x^{1.8-\alpha} - \frac{\Gamma(1.9)}{\Gamma(1.9-\alpha)}x^{0.9-\alpha}, \\ D^{\alpha}_{0+} f_2(x) &= {}_cD^{\alpha}_{0+} f_2(x) - 16\frac{1}{\Gamma(1-\alpha)}x^{-\alpha}, \\ (T^a_{\alpha} f_2)(x) &= (x-a)^{1-\alpha} f_2'(x) = (x-a)^{1-\alpha}(1.8ix^{0.8} - 0.9x^{-0.1}). \end{aligned}$$

En la Tabla 5.2 podemos observar que TeCA y TeRL requieren más iteraciones cuando el valor de α disminuye, y que ρ es lineal. Notamos que TeCO y CKeCO pueden requerir menos o más iteraciones que los métodos clásicos de Traub y Chun-Kim, respectivamente, y el orden de convergencia computacional es alrededor de 3 para cualquier α en ambos esquemas. Podemos ver que OeCO no mejora el procedimiento clásico de Ostrowski en cuanto a número de iteraciones y ρ. Observamos que CeCO presenta un comportamiento similar

	Método TeCA					Método TeRL				
α	$\bar{x}$	$\lvert f_3(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_3(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_1$	5.33×10^{-15}	2.20×10^{-5}	4	2.92	$\bar{x}_1$	5.33×10^{-15}	2.20×10^{-5}	4	2.92
0.9	$\bar{x}_1$	2.32×10^{-9}	7.89×10^{-8}	7	1.00	-	-	-	>500	-
0.8	$\bar{x}_1$	5.21×10^{-9}	5.77×10^{-8}	9	1.00	-	-	-	>500	-
0.7	$\bar{x}_1$	9.80×10^{-9}	5.62×10^{-8}	11	1.00	-	-	-	>500	-
0.6	$\bar{x}_1$	4.73×10^{-9}	1.69×10^{-8}	14	1.00	-	-	-	>500	-
0.5	$\bar{x}_1$	5.09×10^{-9}	1.23×10^{-8}	17	1.00	-	-	-	>500	-
0.4	$\bar{x}_1$	3.75×10^{-9}	6.44×10^{-9}	21	1.00	-	-	-	>500	-
0.3	$\bar{x}_1$	6.67×10^{-9}	8.31×10^{-9}	25	1.00	-	-	-	>500	-
0.2	$\bar{x}_1$	5.93×10^{-9}	5.45×10^{-9}	31	1.00	-	-	-	>500	-
0.1	$\bar{x}_1$	8.45×10^{-9}	5.93×10^{-9}	37	1.00	-	-	-	>500	-
	Método TeCO					Método CKeCO				
α	$\bar{x}$	$\lvert f_3(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_3(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_1$	5.33×10^{-15}	2.20×10^{-5}	4	2.92	$\bar{x}_1$	8.17×10^{-14}	5.18×10^{-5}	4	2.90
0.9	$\bar{x}_1$	8.88×10^{-15}	2.46×10^{-5}	4	2.92	$\bar{x}_1$	1.19×10^{-13}	5.80×10^{-5}	4	2.90
0.8	$\bar{x}_1$	1.24×10^{-14}	2.74×10^{-5}	4	2.91	$\bar{x}_1$	1.67×10^{-13}	6.48×10^{-5}	4	2.90
0.7	$\bar{x}_1$	1.60×10^{-14}	3.06×10^{-5}	4	2.91	$\bar{x}_1$	2.31×10^{-13}	7.22×10^{-5}	4	2.90
0.6	$\bar{x}_1$	2.13×10^{-14}	3.41×10^{-5}	4	2.91	$\bar{x}_1$	3.29×10^{-13}	8.04×10^{-5}	4	2.90
0.5	$\bar{x}_1$	2.84×10^{-14}	3.78×10^{-5}	4	2.91	$\bar{x}_1$	4.58×10^{-13}	8.93×10^{-5}	4	2.90
0.4	$\bar{x}_1$	4.44×10^{-14}	4.20×10^{-5}	4	2.91	$\bar{x}_1$	6.41×10^{-13}	9.90×10^{-5}	4	2.89
0.3	$\bar{x}_1$	5.86×10^{-14}	4.65×10^{-5}	4	2.91	$\bar{x}_1$	8.86×10^{-13}	1.10×10^{-4}	4	2.89
0.2	$\bar{x}_1$	8.17×10^{-14}	5.14×10^{-5}	4	2.90	$\bar{x}_1$	1.22×10^{-12}	1.21×10^{-4}	4	2.89
0.1	$\bar{x}_1$	1.12×10^{-13}	5.68×10^{-5}	4	2.90	$\bar{x}_1$	1.66×10^{-12}	1.34×10^{-4}	4	2.89
	Método OeCO					Método CeCO				
α	$\bar{x}$	$\lvert f_3(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_3(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_1$	0	7.53×10^{-6}	3	3.91	$\bar{x}_1$	1.54×10^{-10}	0.0041	3	3.58
0.9	$\bar{x}_1$	1.18×10^{-15}	7.76×10^{-6}	3	3.91	$\bar{x}_1$	1.97×10^{-10}	0.0043	3	3.58
0.8	$\bar{x}_1$	1.78×10^{-15}	7.99×10^{-6}	3	3.91	$\bar{x}_1$	2.52×10^{-10}	0.0046	3	3.58
0.7	$\bar{x}_1$	1.78×10^{-15}	8.22×10^{-6}	3	3.90	$\bar{x}_1$	3.20×10^{-10}	0.0048	3	3.57
0.6	$\bar{x}_1$	0	8.44×10^{-6}	3	3.90	$\bar{x}_1$	4.05×10^{-10}	0.0051	3	3.57
0.5	$\bar{x}_1$	1.78×10^{-15}	8.67×10^{-6}	3	3.90	$\bar{x}_1$	5.11×10^{-10}	0.0054	3	3.57
0.4	$\bar{x}_1$	0	8.89×10^{-6}	3	3.90	$\bar{x}_1$	6.42×10^{-10}	0.0056	3	3.57
0.3	$\bar{x}_1$	0	9.10×10^{-6}	3	3.90	$\bar{x}_1$	8.04×10^{-10}	0.0059	3	3.56
0.2	$\bar{x}_1$	3.55×10^{-15}	9.31×10^{-6}	3	3.89	$\bar{x}_1$	1.00×10^{-9}	0.0062	3	3.56
0.1	$\bar{x}_1$	5.33×10^{-15}	9.52×10^{-6}	3	3.89	$\bar{x}_1$	1.25×10^{-9}	0.0065	3	3.56

Tabla 5.3: Resultados para $f_3(x)$, con estimación inicial $x_0 = 1.5$

al del método clásico de Chun en en algunos casos, y en otros casos el orden de convergencia computacional puede ser ligeramente mayor, pero requiriendo más iteraciones. Podemos notar, en general, que es posible conseguir distintas soluciones al cambiar el valor de α para una misma estimación inicial x_0, y que se pueden obtener raíces complejas partiendo de estimaciones iniciales reales.

Nuestra tercera función es $f_3(x) = e^x - 1$, con única raíz real $\bar{x}_1 = 0$. Las derivadas necesarias para esta función son:

$$\begin{aligned} {}_cD^{\alpha}_{0^+} f_3(x) &= x^{1-\alpha} E_{1,2-\alpha}(x), \\ D^{\alpha}_{0^+} f_3(x) &= {}_cD^{\alpha}_{0^+} f_3(x) - \frac{1}{\Gamma(1-\alpha)} x^{-\alpha}, \\ (T^a_{\alpha} f_3)(x) &= (x-a)^{1-\alpha} f_3'(x) = (x-a)^{1-\alpha}(e^x). \end{aligned}$$

En la Tabla 5.3 podemos notar que TeCA requiere más iteraciones cuando el valor de α disminuye, mientras

	Método TeCA					Método TeRL				
α	$\bar{x}$	$\lvert f_4(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_4(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_{12}$	1.14×10^{-12}	1.27×10^{-5}	3	4.18	$\bar{x}_{12}$	1.14×10^{-12}	1.27×10^{-5}	3	4.18
0.9	$\bar{x}_{12}$	1.29×10^{-7}	8.86×10^{-9}	19	0.96	$\bar{x}_{12}$	1.32×10^{-7}	9.05×10^{-9}	19	0.96
0.8	$\bar{x}_{12}$	1.05×10^{-6}	9.54×10^{-9}	73	0.99	$\bar{x}_{12}$	1.08×10^{-6}	9.80×10^{-9}	73	0.99
0.7	$\bar{x}_{12}$	6.49×10^{-6}	9.95×10^{-9}	264	1.00	$\bar{x}_{12}$	6.56×10^{-6}	9.94×10^{-9}	267	1.00
0.6	-	-	-	>500	-	-	-	-	>500	-
0.5	-	-	-	>500	-	-	-	-	>500	-
0.4	-	-	-	>500	-	-	-	-	>500	-
0.3	-	-	-	>500	-	-	-	-	>500	-
0.2	-	-	-	>500	-	-	-	-	>500	-
0.1	-	-	-	>500	-	-	-	-	>500	-

	Método TeCO					Método CKeCO				
α	$\bar{x}$	$\lvert f_4(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_4(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_{12}$	1.13×10^{-12}	1.27×10^{-5}	3	4.18	$\bar{x}_{12}$	3.37×10^{-13}	8.77×10^{-6}	3	4.07
0.9	$\bar{x}_{12}$	1.13×10^{-12}	1.27×10^{-5}	3	4.18	$\bar{x}_{12}$	3.37×10^{-13}	8.72×10^{-6}	3	4.07
0.8	$\bar{x}_{12}$	1.11×10^{-12}	1.26×10^{-5}	3	4.18	$\bar{x}_{12}$	3.27×10^{-13}	8.66×10^{-6}	3	4.07
0.7	$\bar{x}_{12}$	1.06×10^{-12}	1.25×10^{-5}	3	4.18	$\bar{x}_{12}$	2.97×10^{-13}	8.60×10^{-6}	3	4.07
0.6	$\bar{x}_{12}$	1.06×10^{-12}	1.25×10^{-5}	3	4.17	$\bar{x}_{12}$	3.17×10^{-13}	8.54×10^{-6}	3	4.07
0.5	$\bar{x}_{12}$	1.04×10^{-12}	1.24×10^{-5}	3	4.17	$\bar{x}_{12}$	3.07×10^{-13}	8.49×10^{-6}	3	4.07
0.4	$\bar{x}_{12}$	1.04×10^{-12}	1.24×10^{-5}	3	4.17	$\bar{x}_{12}$	3.17×10^{-13}	8.43×10^{-6}	3	4.06
0.3	$\bar{x}_{12}$	1.02×10^{-12}	1.23×10^{-5}	3	4.17	$\bar{x}_{12}$	3.07×10^{-13}	8.37×10^{-6}	3	4.06
0.2	$\bar{x}_{12}$	1.01×10^{-12}	1.22×10^{-5}	3	4.17	$\bar{x}_{12}$	2.77×10^{-13}	8.32×10^{-6}	3	4.06
0.1	$\bar{x}_{12}$	9.81×10^{-13}	1.22×10^{-5}	3	4.17	$\bar{x}_{12}$	2.77×10^{-13}	8.26×10^{-6}	3	4.06

	Método OeCO					Método CeCO				
α	$\bar{x}$	$\lvert f_4(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_4(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_{12}$	7.85×10^{-11}	4.17×10^{-4}	2	-	$\bar{x}_{12}$	4.27×10^{-15}	5.00×10^{-7}	3	4.81
0.9	$\bar{x}_{12}$	7.83×10^{-11}	4.17×10^{-4}	2	-	$\bar{x}_{12}$	1.58×10^{-14}	4.96×10^{-7}	3	4.81
0.8	$\bar{x}_{12}$	7.80×10^{-11}	4.17×10^{-4}	2	-	$\bar{x}_{12}$	1.58×10^{-14}	4.92×10^{-7}	3	4.81
0.7	$\bar{x}_{12}$	7.79×10^{-11}	4.17×10^{-4}	2	-	$\bar{x}_{12}$	4.27×10^{-15}	4.89×10^{-7}	3	4.81
0.6	$\bar{x}_{12}$	7.76×10^{-11}	4.17×10^{-4}	2	-	$\bar{x}_{12}$	4.27×10^{-15}	4.85×10^{-7}	3	4.81
0.5	$\bar{x}_{12}$	7.74×10^{-11}	4.16×10^{-4}	2	-	$\bar{x}_{12}$	4.27×10^{-15}	4.81×10^{-7}	3	4.81
0.4	$\bar{x}_{12}$	7.72×10^{-11}	4.16×10^{-4}	2	-	$\bar{x}_{12}$	1.44×10^{-14}	4.77×10^{-7}	3	4.81
0.3	$\bar{x}_{12}$	7.70×10^{-11}	4.16×10^{-4}	2	-	$\bar{x}_{12}$	2.58×10^{-14}	4.74×10^{-7}	3	4.81
0.2	$\bar{x}_{12}$	7.67×10^{-11}	4.16×10^{-4}	2	-	$\bar{x}_{12}$	5.72×10^{-15}	4.70×10^{-7}	3	4.81
0.1	$\bar{x}_{12}$	7.66×10^{-11}	4.15×10^{-4}	2	-	$\bar{x}_{12}$	4.59×10^{-14}	4.67×10^{-7}	3	4.81

Tabla 5.4: Resultados para $f_4(x)$, con estimación inicial $x_0 = 2$

que TeRL no encuentra solución para $\alpha \neq 1$; ρ es lineal cuando se usa TeCA. Vemos que TeCO, CKeCO, OeCO y CeCO presentan un comportamiento muy similar a sus versiones clásicas, respectivamente, y el orden de convergencia computacional es el esperado en cada caso. Algunos errores son cero para OeCO porque se usa aritmética de precisión doble; un valor muy cercano a cero se podría obtener si se usara aritmética de precisión variable.

Finalmente, la cuarta función es $f_4(x) = \sin 10x - 0.5x + 0.2$, con raíces reales $\bar{x}_1 = -1.4523$, $\bar{x}_2 = -1.3647$, $\bar{x}_3 = -0.87345$, $\bar{x}_4 = -0.6857$, $\bar{x}_5 = -0.27949$, $\bar{x}_6 = -0.021219$, $\bar{x}_7 = 0.31824$, $\bar{x}_8 = 0.64036$, $\bar{x}_9 = 0.91636$, $\bar{x}_{10} = 1.3035$, $\bar{x}_{11} = 1.5118$, $\bar{x}_{12} = 1.9756$ y $\bar{x}_{13} = 2.0977$. Las derivadas necesarias para esta función son:

$$
\begin{aligned}
{}_cD^{\alpha}_{0+} f_4(x) &= 5x^{1-\alpha}\left(E_{1,2-\alpha}(10ix) + E_{1,2-\alpha}(-10ix)\right) - \frac{0.5\Gamma(2)}{\Gamma(2-\alpha)}x^{1-\alpha},\\
D^{\alpha}_{0+} f_4(x) &= {}_cD^{\alpha}_{0+} f_4(x) + 0.2\frac{1}{\Gamma(1-\alpha)}x^{-\alpha},\\
(T^a_{\alpha} f_4)(x) &= (x-a)^{1-\alpha} f_4'(x) = (x-a)^{1-\alpha}(10\cos 10x - 0.5).
\end{aligned}
$$

En la Tabla 5.4 podemos observar que TeCA y TeRL requieren más iteraciones cuando el valor de α disminuye, y que ρ es lineal; no se muestran resultados cuando no se encuentra solución en 500 iteraciones. Notamos que TeCO, CKeCO, OeCO y CeCO presentan un comportamiento muy similar a sus versiones clásicas, respectivamente, y el orden de convergencia computacional es mayor que el esperado cuando se usan TeCO, CKeCO y CeCO; no se proporciona ρ cuando se usa OeCO porque es necesario al menos tres iteraciones para ser calculado.

5.3.1. Estabilidad numérica

Vamos a analizar la dependencia respecto a las estimaciones iniciales de los esquemas diseñados en este capítulo con el uso de los planos de convergencia definidos en [29] (ver ficheros MATLAB en Anexos A.30 a A.35). En éstos, el eje de abscisas corresponde a la estimación inicial x_0, y el orden de la derivada α aparece en el eje de las ordenadas. Se utiliza una malla de 400×400 puntos. Los puntos que no se representan en negro corresponden a los pares de estimaciones iniciales y valores de α que convergen a una de las raíces con una tolerancia de 10^{-3}. Diferentes colores significan convergencia a diferentes raíces. Por lo tanto, cuando un punto se representa en negro, esto muestra que no se encuentra ninguna raíz en un máximo de 500 iteraciones. Además, para todos los planos de convergencia se calcula el porcentaje de pares convergentes (x_0, α), con el objetivo de comparar la eficiencia de tales procedimientos.

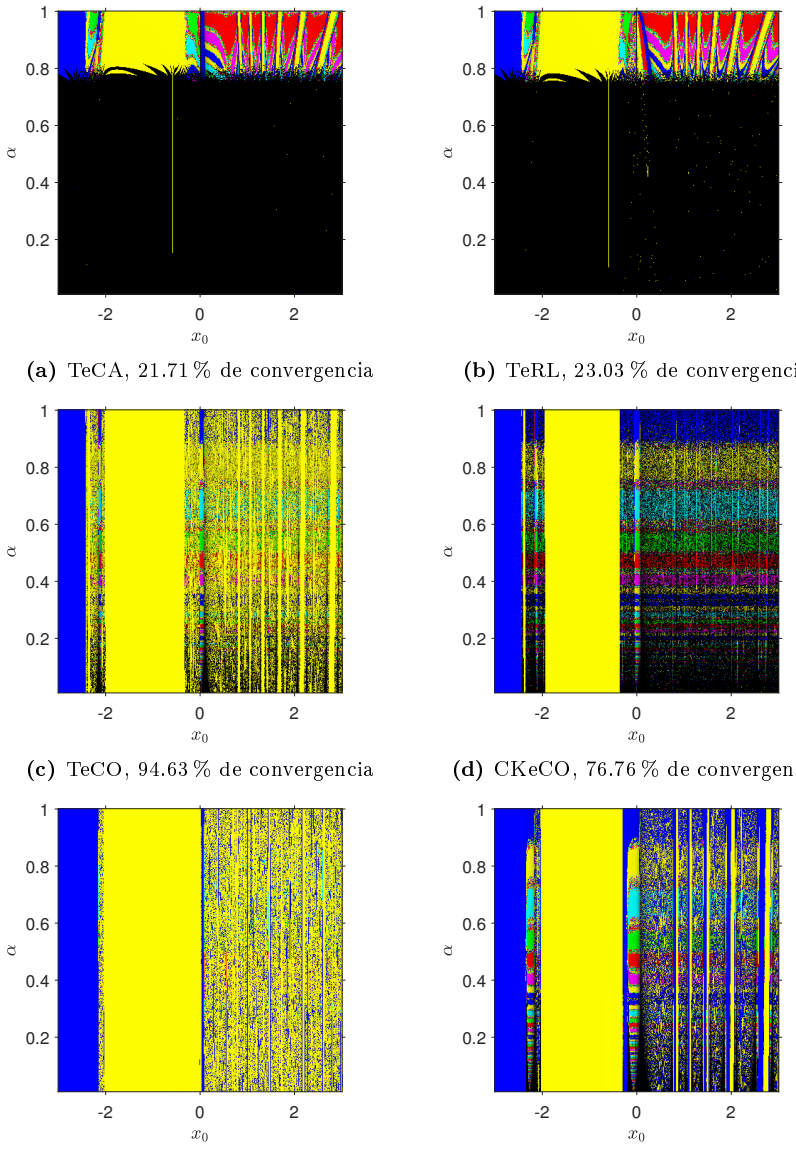
En la Figura 5.1 notamos que TeCA y TeRL alcanzan aproximadamente entre 21 % y 23 % de convergencia, TeCO, OeCO y CeCO consiguen entre el 94 % y 99 % de convergencia, y CKeCO alcanza alrededor del 77 % de convergencia; todas las raíces se encuentran en cada plano.

En la Figura 5.2 podemos ver que TeCA y TeRL consiguen entre un 15 % y un 16 % de convergencia, mientras que TeCO, CKeCO, OeCO y CeCO casi alcanzan el 100 % de convergencia; las dos raíces se encuentran en cada plano.

En la Figura 5.3 observamos que TeCA, TeCO, CKeCO y CeCO consiguen entre 53 % y 59 % de convergencia, TeRL alcanza sólo alrededor del 1 %, y OeCO consigue alrededor del 74 % de convergencia. En el caso de los procedimientos multipunto (TeCO, CKeCO, OeCO y CeCO), considerando que $-10 \leq x_0 \leq 10$, y $a = -10$, señalamos que la estabilidad podría ser ligeramente mejorada si escogemos $a < -10$.

En la Figura 5.4 podemos notar que TeCA y TeRL alcanzan entre 12 % y 13 % de convergencia, TeCO y CeCO consiguen entre 62 % y 66 % de convergencia, CKeCO alcanza alrededor del 48 % de convergencia, y OeCO consigue alrededor del 73 % de convergencia; de nuevo, todas las raíces se encuentran en cada plano.

En este capítulo, hemos diseñado, realizado el análisis de convergencia y estudiado la estabilidad de dos métodos fraccionarios de tipo Traub con derivadas de Caputo y de Riemann-Liouville (TeCA y TeRL, respectivamente). También se ha proporcionado una técnica general para obtener la versión conformable de cualquier esquema clásico, y fue utilizada para diseñar procedimientos de tipo Traub, Chun-Kim, Ostrowski y Chun con derivadas conformables (TeCO, CKeCO, OeCO y CeCO, respectivamente), para luego realizar sus respectivos análisis de convergencia y estudio de la estabilidad. En la teoría, TeCA y TeRL tienen orden de convergencia inferior al caso clásico de Traub (cuando $\alpha = 1$), mientras que TeCO, CKeCO, OeCO y CeCO preservan el orden de sus versiones clásicas, respectivamente. En la práctica, hemos observado que es posible converger con métodos conformables cuando los clásicos fallan, en algunos casos se puede converger en menos iteraciones que en el caso clásico, y ρ puede ser ligeramente superior; también se ha comprobado que, en general, se puede converger a raíces complejas con estimaciones iniciales reales, y que es posible conseguir una raíz diferente al modificar el valor del orden de la derivada α. En el siguiente capítulo, diseñamos, realizamos el análisis de convergencia, y estudiamos la estabilidad de la versión fraccionaria de algunos métodos libres de derivadas clásicos.



(a) TeCA, 21.71 % de convergencia

(b) TeRL, 23.03 % de convergencia

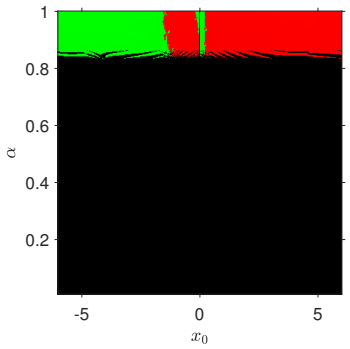
(c) TeCO, 94.63 % de convergencia

(d) CKeCO, 76.76 % de convergencia

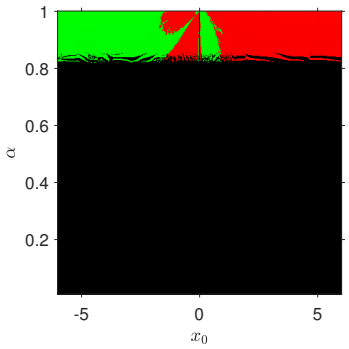
(e) OeCO, 99.46 % de convergencia

(f) CeCO, 94.04 % de convergencia

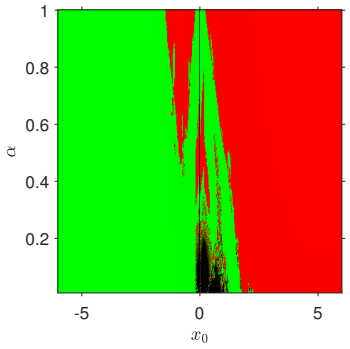
Figura 5.1: Planos de convergencia para $f_1(x)$



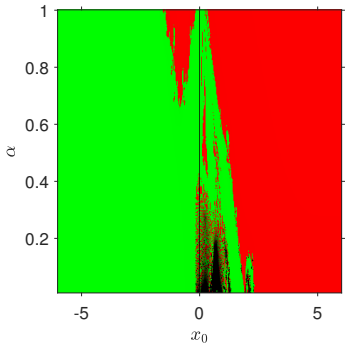
(a) TeCA, 15.27 % de convergencia



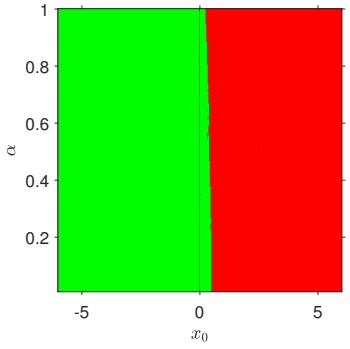
(b) TeRL, 16.45 % de convergencia



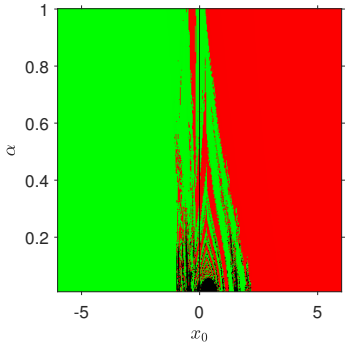
(c) TeCO, 98.91 % de convergencia



(d) CKeCO, 99.01 % de convergencia

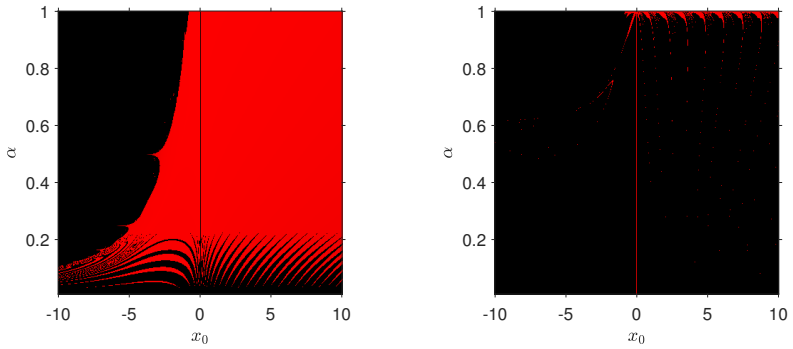


(e) OeCO, 99.75 % de convergencia



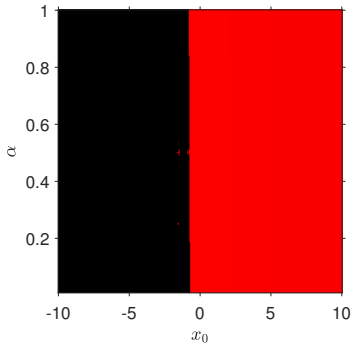
(f) CeCO, 98.72 % de convergencia

Figura 5.2: Planos de convergencia para $f_2(x)$

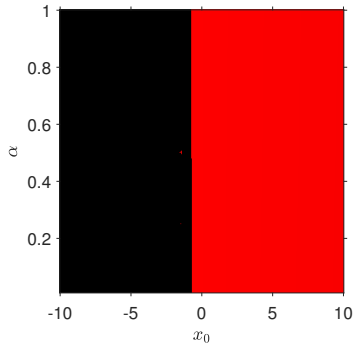


(a) TeCA, 56.27 % de convergencia

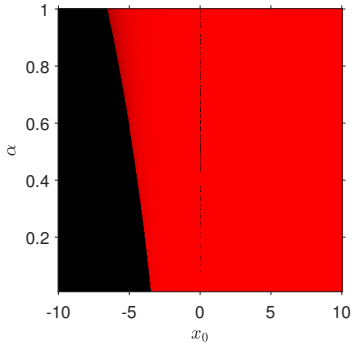
(b) TeRL, 1.34 % de convergencia



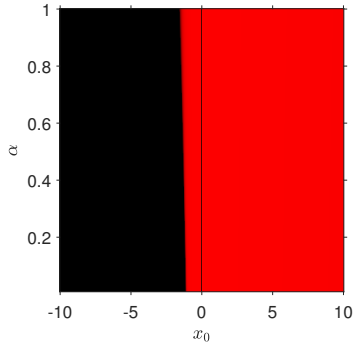
(c) TeCO, 53.88 % de convergencia



(d) CKeCO, 53.50 % de convergencia

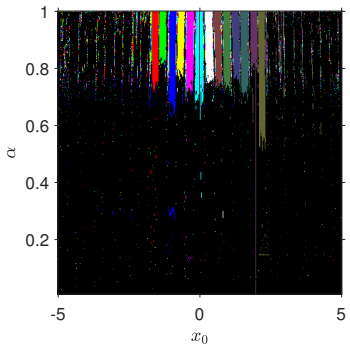


(e) OeCO, 74.04 % de convergencia

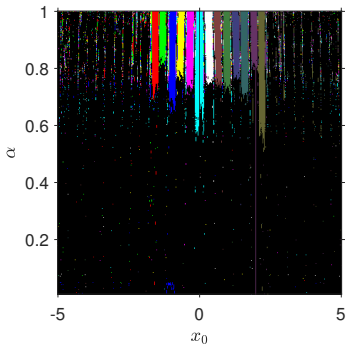


(f) CeCO, 56.31 % de convergencia

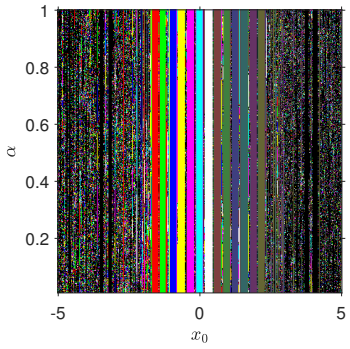
Figura 5.3: Planos de convergencia para $f_3(x)$



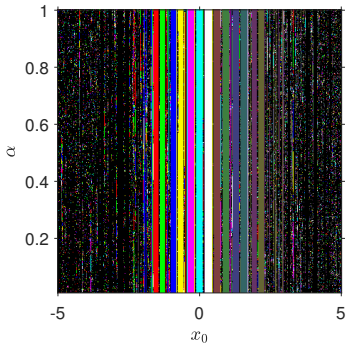
(a) TeCA, 12.98 % de convergencia



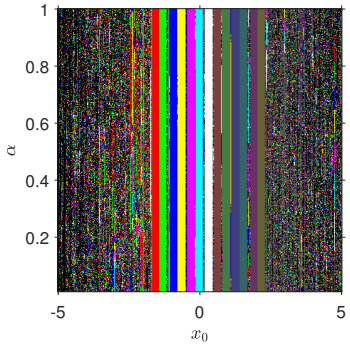
(b) TeRL, 13.31 % de convergencia



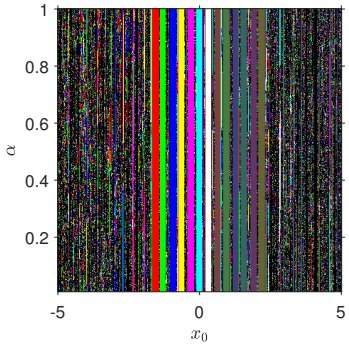
(c) TeCO, 66.19 % de convergencia



(d) CKeCO, 48.23 % de convergencia



(e) OeCO, 73.67 % de convergencia



(f) CeCO, 62.55 % de convergencia

Figura 5.4: Planos de convergencia para $f_4(x)$

Capítulo 6

Métodos iterativos fraccionarios escalares libres de derivadas

"Los encantos de esta ciencia sublime, las matemáticas, sólo se le revelan a aquellos que tienen el valor de profundizar en ella"

Carl Friedrich Gauss (1777 - 1855)

En este capítulo proponemos (hasta donde sabemos) los primeros esquemas fraccionarios libres de derivadas en la literatura: de tipo Steffensen (sin memoria), y de tipo Secante (con memoria), donde aproximamos derivadas conformables. Recordemos que la derivada fraccionaria conformable por la izquierda (véase [1, 23]) de una función $f : [a, \infty) \to \mathbb{R}$, centrada en a, de orden $\alpha \in (0, 1]$, $\alpha, a, x \in \mathbb{R}$, $a < x$, se define como

$$(T_\alpha^a f)(x) = \lim_{\varepsilon \to 0} \frac{f(x + \varepsilon(x-a)^{1-\alpha}) - f(x)}{\varepsilon},$$

la cual podemos aproximar con la siguiente diferencia dividida finita fraccionaria conformable de orden lineal (ver [40]):

$$(T_\alpha^a f)(x) \approx \frac{f(x + \varepsilon(x-a)^{1-\alpha}) - f(x)}{\varepsilon}, \quad \varepsilon \neq 0. \tag{6.1}$$

En los Capítulos 3, 4 y 5 hemos visto que los procedimientos conformables preservan el orden de convergencia teórico de sus versiones clásicas, y que presentan algunas ventajas numéricas sobre éstos, no importando si estos métodos son escalares o vectoriales, o si son punto a punto o multipunto. Nos preguntamos ahora si en las versiones conformables de esquemas libres de derivadas (con o sin memoria) se mantienen estas mismas ventajas. En la siguiente sección, mostramos el diseño y análisis de convergencia de la versión conformable del procedimiento de Steffensen.

6.1. Método de Steffensen fraccionario conformable

Tengamos en cuenta que el esquema clásico de Steffensen es el siguiente [36, 42]:

$$\begin{aligned}\phi_1(x) &= x - \frac{[f(x)]^2}{f(x+f(x)) - f(x)} \\ &= x - \frac{f(x)}{f(x+f(x)) - f(x)} f(x),\end{aligned}$$

donde

$$\frac{f(x+f(x)) - f(x)}{f(x)} \neq 0$$

es una aproximación de la derivada clásica. Veamos el diseño de la versión conformable de este procedimiento.

6.1.1. Deducción del método

Podemos extender la técnica general vista en la Sección 5.2 para el diseño de métodos conformables libres de derivadas. En este caso,

$$g(x) = \frac{f(x)}{f(x+f(x)) - f(x)}.$$

Considerando (6.1),

$$g_\alpha(x) = \frac{f(x)}{f(x+f(x)(x-a)^{1-\alpha}) - f(x)},$$

donde $\varepsilon = f(x)$ en (6.1). Finalmente, la versión conformable del esquema de Steffensen resulta

$$\phi_2(x) = a + \left((x-a)^\alpha - \alpha \frac{[f(x)]^2}{f(x+f(x)(x-a)^{1-\alpha}) - f(x)} \right)^{1/\alpha},$$

y lo denotamos por SeCO; notemos que cuando $\alpha = 1$ obtenemos el procedimiento clásico de Steffensen.

6.1.2. Análisis de convergencia

En el siguiente resultado se establecen las condiciones que aseguran la convergencia de SeCO. Recordemos que utilizamos desarrollos de Taylor clásicos, debido a su equivalencia con los desarrollos de Taylor conformables vista en la Observación 3.3.3; usaremos la notación $x = x_k$ y $\phi(x) = x_{k+1}$.

Teorema 6.1.1 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo un cero $\bar{x}$ de $f(x)$. Si una aproximación inicial x_0 es lo suficientemente cercana a $\bar{x}$, entonces el orden de convergencia local del procedimiento de Steffensen conformable (SeCO)*

$$x_{k+1} = a + \left((x_k - a)^\alpha - \alpha \frac{[f(x_k)]^2}{f(x_k + f(x_k)(x_k-a)^{1-\alpha}) - f(x_k)} \right)^{1/\alpha}, \quad k = 0, 1, 2, \ldots, \tag{6.2}$$

es al menos 2, siendo $0 < \alpha \leq 1$, y la ecuación del error es

$$e_{k+1} = \left(\left(1 + f'(\bar{x})(\bar{x}-a)^{1-\alpha}\right) C_2 + \frac{1}{2}\frac{1-\alpha}{\bar{x}-a} \right) e_k^2 + O\left(e_k^3\right),$$

donde $C_j = \dfrac{f^{(j)}(\bar{x})}{j!f'(\bar{x})}$, *para* $j \geq 2$, *tal que* $a < x_k$, $\forall k$.

Demostración. Sabiendo que $x_k = e_k + \bar{x}$, el desarrollo de Taylor de $f(x_k)$ y $[f(x_k)]^2$ en el entorno de $\bar{x}$ se puede expresar como

$$f(x_k) = f'(\bar{x})\left[e_k + C_2 e_k^2 + C_3 e_k^3\right] + O\left(e_k^4\right)$$

y

$$[f(x_k)]^2 = [f'(\bar{x})]^2\left[e_k^2 + 2C_2 e_k^3\right] + O\left(e_k^4\right),$$

respectivamente, siendo $C_j = \dfrac{f^{(j)}(\bar{x})}{j!f'(\bar{x})}$, para $j \geq 2$.

Usando el Teorema generalizado del binomio (2.52) y el coeficiente binomial fraccionario (2.53),

$$x_k + f(x_k)(x_k-a)^{1-\alpha} = \left(1 + f'(\bar{x})(\bar{x}-a)^{1-\alpha}\right) e_k + f'(\bar{x})\left(\frac{1-\alpha+(\bar{x}-a)C_2}{(\bar{x}-a)^{\alpha}}\right) e_k^2 + O\left(e_k^3\right).$$

Así, el desarrollo de $f(x_k + f(x_k)(x_k-a)^{1-\alpha})$ es

$$\begin{aligned} f(x_k + f(x_k)(x_k-a)^{1-\alpha}) &= f'(\bar{x})\left[\left(1 + f'(\bar{x})(\bar{x}-a)^{1-\alpha}\right) e_k \right. \\ &+ \left. \left(\left(1 + f'(\bar{x})(\bar{x}-a)^{1-\alpha}\right)^2 C_2 + \frac{1-\alpha+(\bar{x}-a)C_2}{(\bar{x}-a)^{\alpha}}\right) e_k^2\right] + O\left(e_k^3\right), \end{aligned}$$

y al restarle $f(x_k)$, resulta

$$\begin{aligned} f(x_k + f(x_k)(x_k-a)^{1-\alpha}) - f(x_k) &= [f'(\bar{x})]^2\left[(\bar{x}-a)^{1-\alpha} e_k \right. \\ &+ \left. \left(\frac{1-\alpha}{(\bar{x}-a)^{\alpha}} + (\bar{x}-a)^{1-2\alpha}\left(f'(\bar{x})(\bar{x}-a) + 3(\bar{x}-a)^{\alpha}\right) C_2\right) e_k^2\right] \\ &+ O\left(e_k^3\right). \end{aligned}$$

El cociente $\alpha\dfrac{[f(x_k)]^2}{f(x_k + f(x_k)(x_k-a)^{1-\alpha}) - f(x_k)}$ da como resultado

$$\begin{aligned} \alpha\frac{[f(x_k)]^2}{f(x_k + f(x_k)(x_k-a)^{1-\alpha}) - f(x_k)} &= \alpha\left[(\bar{x}-a)^{\alpha-1} e_k \right. \\ &+ \left. \left(\frac{\alpha-1}{(\bar{x}-a)^{2-\alpha}} - \frac{\left(f'(\bar{x})(\bar{x}-a) + (\bar{x}-a)^{\alpha}\right) C_2}{\bar{x}-a}\right) e_k^2\right] + O\left(e_k^3\right). \end{aligned}$$

Usando de nuevo el Teorema generalizado del binomio,

$$(x_k-a)^\alpha=(\bar{x}-a)^\alpha+\alpha(\bar{x}-a)^{\alpha-1}e_k+\frac{1}{2}\alpha(\alpha-1)(\bar{x}-a)^{\alpha-2}e_k^2+O\left(e_k^3\right),$$

y de este modo,

$$\begin{aligned}(x_k-a)^\alpha-\alpha\frac{[f(x_k)]^2}{f(x_k+f(x_k)(x_k-a)^{1-\alpha})-f(x_k)} &= (\bar{x}-a)^\alpha\\ &+ \alpha\left(\frac{1-\alpha}{2(\bar{x}-a)^{2-\alpha}}+\frac{(f'(\bar{x})(\bar{x}-a)+(\bar{x}-a)^\alpha)\,C_2}{\bar{x}-a}\right)e_k^2\\ &+ O\left(e_k^3\right).\end{aligned}$$

Usando el Teorema generalizado del binomio una vez más,

$$\begin{aligned}\left((x_k-a)^\alpha-\alpha\frac{[f(x_k)]^2}{f(x_k+f(x_k)(x_k-a)^{1-\alpha})-f(x_k)}\right)^{1/\alpha} &= \bar{x}-a\\ &+ \left(\left(1+f'(\bar{x})(\bar{x}-a)^{1-\alpha}\right)C_2+\frac{1}{2}\frac{1-\alpha}{\bar{x}-a}\right)e_k^2\\ &+ O\left(e_k^3\right).\end{aligned}$$

Finalmente, sabiendo que $x_{k+1}=e_{k+1}+\bar{x}$,

$$e_{k+1}=\left(\left(1+f'(\bar{x})(\bar{x}-a)^{1-\alpha}\right)C_2+\frac{1}{2}\frac{1-\alpha}{\bar{x}-a}\right)e_k^2+O\left(e_k^3\right).$$

Esto completa la prueba. □

Observación 6.1.1 *Dado que la ecuación del error del método clásico de Steffensen es (ver (2.30))*

$$e_{k+1}=\left(1+f'(\bar{x})\right)C_2e_k^2+O\left(e_k^3\right),$$

se comprueba la relación entre las constantes asintóticas del error vista en el Teorema 2.1.3.

Observación 6.1.2 *SeCO es el primer procedimiento fraccionario óptimo libre de derivadas.*

En la siguiente sección, presentamos el diseño y análisis de convergencia de la versión conformable del esquema de la Secante.

6.2. Método de la Secante fraccionario conformable

Recordemos que el procedimiento clásico de la Secante es [36, 42]:

$$\begin{aligned}\phi_3(x;y) &= x-\frac{(y-x)f(x)}{f(y)-f(x)}\\ &= x-\frac{y-x}{f(y)-f(x)}f(x),\end{aligned}$$

donde

$$\frac{f(y)-f(x)}{y-x} \neq 0$$

es una aproximación de la derivada clásica. Veamos el diseño de la versión conformable de este método.

6.2.1. Deducción del método

Usando la técnica general vista en la Sección 5.2, en este caso,

$$g(x) = \frac{y-x}{f(y)-f(x)}.$$

Considerando (6.1),

$$g_\alpha(x) = \frac{y-x}{f(x+(y-x)(x-a)^{1-\alpha})-f(x)},$$

donde $\varepsilon = y - x$ en (6.1). Como consecuencia, la versión conformable del esquema de la Secante es

$$\phi_4(x;y) = a + \left((x-a)^\alpha - \alpha\frac{(y-x)f(x)}{f(x+(y-x)(x-a)^{1-\alpha})-f(x)}\right)^{1/\alpha},$$

y lo denotamos por EeCO; notemos que cuando $\alpha = 1$ obtenemos el procedimiento clásico de la Secante.

6.2.2. Análisis de convergencia

En el siguiente resultado se plantean las condiciones que garantizan la convergencia de EeCO; usaremos la notación $x = x_k$, $y = x_{k-1}$ y $\phi(x;y) = x_{k+1}$.

Teorema 6.2.1 *Sea $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una función suficientemente diferenciable en el intervalo I, conteniendo un cero $\bar{x}$ de $f(x)$. Si las aproximaciones iniciales x_0 y x_{-1} son lo suficientemente cercanas a $\bar{x}$, entonces el orden de convergencia local del método de la Secante conformable (EeCO)*

$$x_{k+1} = a + \left((x_k-a)^\alpha - \alpha\frac{(x_{k-1}-x_k)f(x_k)}{f(x_k+(x_{k-1}-x_k)(x_k-a)^{1-\alpha})-f(x_k)}\right)^{1/\alpha}, \quad k = 0,1,2,\ldots, \tag{6.3}$$

es al menos 1.618*, siendo $0 < \alpha \leq 1$, y la ecuación del error es*

$$e_{k+1} = (\bar{x}-a)^{1-\alpha}C_2 e_k e_{k-1} + O\left(e_k e_{k-1}^2\right),$$

donde $C_j = \dfrac{f^{(j)}(\bar{x})}{j!f'(\bar{x})}$, para $j \geq 2$, tal que $a < x_k$, $a < x_{k-1}$, $\forall k$.

Demostración. Sabiendo que $x_k = e_k + \bar{x}$, podemos expresar el desarrollo de Taylor de $f(x_k)$ en el entorno de $\bar{x}$ como

$$f(x_k) = f'(\bar{x})\left[e_k + C_2 e_k^2 + C_3 e_k^3\right] + O\left(e_k^4\right),$$

siendo $C_j = \dfrac{f^{(j)}(\bar{x})}{j! f'(\bar{x})}$, para $j \geq 2$.

Sabiendo que $x_{k-1} = e_{k-1} + \bar{x}$, entonces $x_{k-1} - x_k = e_{k-1} - e_k$. Así, usando el Teorema generalizado del binomio (2.52) y el coeficiente binomial fraccionario (2.53),

$$\begin{aligned} x_k + (x_{k-1} - x_k)(x_k - a)^{1-\alpha} &= \bar{x} + (\bar{x} - a)^{1-\alpha} e_{k-1} + \left(1 - (\bar{x} - a)^{1-\alpha}\right) e_k + (1-\alpha)(\bar{x} - a)^{-\alpha} e_k e_{k-1} \\ &+ (\alpha - 1)(\bar{x} - a)^{-\alpha} e_k^2 + O\left(e_k^2 e_{k-1}\right). \end{aligned}$$

De esta manera, el desarrollo de $f(x_k + (x_{k-1} - x_k)(x_k - a)^{1-\alpha})$ es

$$\begin{aligned} f(x_k + (x_{k-1} - x_k)(x_k - a)^{1-\alpha}) &= f'(\bar{x})\left[(\bar{x} - a)^{1-\alpha} e_{k-1} + \left(1 - (\bar{x} - a)^{1-\alpha}\right) e_k + (\bar{x} - a)^{2-2\alpha} C_2 e_{k-1}^2 \right. \\ &+ \left. \left((1-\alpha)(\bar{x} - a)^{-\alpha} + 2(\bar{x} - a)^{1-\alpha} C_2 - 2(\bar{x} - a)^{2-2\alpha} C_2\right) e_k e_{k-1}\right] \\ &+ O\left(e_k e_{k-1}^2\right), \end{aligned}$$

y al restarle $f(x_k)$, resulta

$$\begin{aligned} f(x_k + (x_{k-1} - x_k)(x_k - a)^{1-\alpha}) - f(x_k) &= f'(\bar{x})\left[(\bar{x} - a)^{1-\alpha} e_{k-1} - (\bar{x} - a)^{1-\alpha} e_k + (\bar{x} - a)^{2-2\alpha} C_2 e_{k-1}^2 \right. \\ &+ \left. \left((1-\alpha)(\bar{x} - a)^{-\alpha} + 2(\bar{x} - a)^{1-\alpha} C_2 - 2(\bar{x} - a)^{2-2\alpha} C_2\right) e_k e_{k-1}\right] \\ &+ O\left(e_k e_{k-1}^2\right). \end{aligned}$$

De nuevo, sabiendo que $x_{k-1} - x_k = e_{k-1} - e_k$, el cociente $\alpha \dfrac{(x_{k-1} - x_k) f(x_k)}{f(x_k + (x_{k-1} - x_k)(x_k - a)^{1-\alpha}) - f(x_k)}$ da como resultado

$$\alpha \frac{(x_{k-1} - x_k) f(x_k)}{f(x_k + (x_{k-1} - x_k)(x_k - a)^{1-\alpha}) - f(x_k)} = \alpha(\bar{x} - a)^{\alpha-1} e_k - \alpha C_2 e_k e_{k-1} + O\left(e_k e_{k-1}^2\right).$$

Usando de nuevo el Teorema generalizado del binomio,

$$(x_k - a)^\alpha = (\bar{x} - a)^\alpha + \alpha(\bar{x} - a)^{\alpha-1} e_k + \frac{1}{2}\alpha(\alpha - 1)(\bar{x} - a)^{\alpha-2} e_k^2 + O\left(e_k^3\right),$$

y por lo tanto,

$$(x_k - a)^\alpha - \alpha \frac{(x_{k-1} - x_k) f(x_k)}{f(x_k + (x_{k-1} - x_k)(x_k - a)^{1-\alpha}) - f(x_k)} = (\bar{x} - a)^\alpha + \alpha C_2 e_k e_{k-1} + O\left(e_k e_{k-1}^2\right).$$

Usando el Teorema generalizado del binomio una vez más,

$$\left((x_k - a)^\alpha - \alpha \frac{(x_{k-1} - x_k) f(x_k)}{f(x_k + (x_{k-1} - x_k)(x_k - a)^{1-\alpha}) - f(x_k)}\right)^{1/\alpha} = \bar{x} - a + (\bar{x} - a)^{1-\alpha} C_2 e_k e_{k-1} + O\left(e_k e_{k-1}^2\right).$$

Finalmente, sabiendo que $x_{k+1} = e_{k+1} + \bar{x}$,

$$e_{k+1} = (\bar{x} - a)^{1-\alpha} C_2 e_k e_{k-1} + O\left(e_k e_{k-1}^2\right).$$

Usando el Teorema 2.1.1, el polinomio característico que se obtiene es $s^2 - s - 1 = 0$, cuya única raíz positiva es $s \approx 1.618$. Así, la convergencia del método de la Secante conformable es superlineal.

Esto completa la prueba. □

Observación 6.2.1 *Dado que la ecuación del error del esquema clásico de la Secante es (ver (2.32))*

$$e_{k+1} = C_2 e_k e_{k-1} + O\left(e_k e_{k-1}^2\right),$$

se comprueba la relación entre las constantes asintóticas del error vista en el Teorema 2.1.3.

Observación 6.2.2 *EeCO es el primer método fraccionario con memoria libre de derivadas.*

Hemos visto que el orden de convergencia teórico de los esquemas clásicos libres de derivadas (con o sin memoria) también se preserva en la versión conformable de dichos procedimientos. En la siguiente sección, realizamos diversas pruebas numéricas con algunas funciones no lineales, y estudiamos la estabilidad de tales métodos.

6.3. Pruebas numéricas

Para obtener los resultados que se muestran en esta sección, hemos utilizado Matlab R2020a con aritmética de precisión doble, $|f(x_{k+1})| < 10^{-8}$ o $|x_{k+1} - x_k| < 10^{-8}$ como criterios de parada, y a lo sumo 500 iteraciones. También usamos el Orden de Convergencia Computacional Aproximado (ACOC)

$$ACOC = \rho = \frac{\ln(|x_{k+1} - x_k|/|x_k - x_{k-1}|)}{\ln(|x_k - x_{k-1}|/|x_{k-1} - x_{k-2}|)}, \quad k = 2, 3, 4, \ldots,$$

definido en [17], para confirmar que el orden de convergencia teórico también se conserva en la práctica.

Vamos a probar cuatro funciones no lineales con los métodos que hemos diseñado en este capítulo (ver ficheros MATLAB en Anexos A.36 y A.37); en este sentido, comparamos cada esquema con su versión clásica (cuando $\alpha = 1$). Para EeCO seleccionamos $x_{-1} = x_0 + 1$ para realizar la primera iteración, fijamos $a = -10$ para cada método, y $\alpha \in (0, 1]$.

En cada tabla mostramos los resultados que se obtuvieron para cada función de prueba con el uso de los dos esquemas diseñados en este capítulo (SeCO y EeCO), donde x_0 coincide en ambos procedimientos.

La primera función es $f_1(x) = -12.84x^6 - 25.6x^5 + 16.55x^4 - 2.21x^3 + 26.71x^2 - 4.29x - 15.21$, con raíces reales y complejas $\bar{x}_1 = 0.82366 + 0.24769i$, $\bar{x}_2 = 0.82366 - 0.24769i$, $\bar{x}_3 = -2.62297$, $\bar{x}_4 = -0.584$, $\bar{x}_5 = -0.21705 + 0.99911i$ y $\bar{x}_6 = -0.21705 - 0.99911i$.

En la Tabla 6.1 vemos que el procedimiento SeCO puede requerir la misma cantidad de iteraciones que el método clásico de Steffensen (cuando $\alpha = 1$); también el número de iteraciones aumenta cuando α disminuye, y ρ puede ser ligeramente superior a 2 cuando $\alpha \neq 1$. Notemos que SeCO necesita que la estimación inicial x_0 esté muy cercana a $\bar{x}_4$ para converger con cualquier α. Podemos observar que EeCO requiere menos iteraciones que el esquema clásico de la Secante para la mayoría de los valores de α, y el orden de convergencia computacional puede ser ligeramente superior a 1.618, aunque inferior en algunos casos.

	Método SeCO					Método EeCO				
α	$\bar{x}$	$\lvert f_1(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_1(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_4$	1.07×10^{-14}	2.18×10^{-10}	5	2.00	$\bar{x}_4$	9.95×10^{-14}	3.48×10^{-10}	5	2.38
0.9	$\bar{x}_4$	2.06×10^{-13}	4.04×10^{-9}	5	2.01	$\bar{x}_4$	3.60×10^{-10}	6.53×10^{-8}	5	2.47
0.8	$\bar{x}_4$	3.48×10^{-11}	6.31×10^{-8}	5	2.03	$\bar{x}_4$	1.50×10^{-10}	3.71×10^{-8}	5	2.23
0.7	$\bar{x}_4$	6.78×10^{-9}	7.87×10^{-7}	5	2.07	$\bar{x}_4$	6.02×10^{-12}	6.64×10^{-9}	4	1.48
0.6	$\bar{x}_4$	2.04×10^{-12}	1.22×10^{-8}	6	2.02	$\bar{x}_4$	1.88×10^{-9}	3.22×10^{-7}	4	0.90
0.5	$\bar{x}_4$	6.98×10^{-9}	6.37×10^{-7}	6	2.07	$\bar{x}_4$	3.39×10^{-9}	4.64×10^{-7}	4	0.80
0.4	$\bar{x}_4$	9.73×10^{-11}	6.72×10^{-8}	7	2.04	$\bar{x}_4$	5.25×10^{-9}	5.85×10^{-7}	4	0.74
0.3	$\bar{x}_4$	5.16×10^{-12}	1.39×10^{-8}	8	2.02	$\bar{x}_4$	8.13×10^{-9}	7.30×10^{-7}	4	0.70
0.2	$\bar{x}_4$	6.36×10^{-13}	4.01×10^{-9}	9	2.02	$\bar{x}_4$	2.24×10^{-13}	2.09×10^{-10}	5	2.53
0.1	$\bar{x}_4$	2.16×10^{-11}	2.24×10^{-8}	9	2.03	$\bar{x}_4$	6.54×10^{-13}	3.29×10^{-10}	5	2.63

Tabla 6.1: Resultados para $f_1(x)$, con estimaciones iniciales $x_0 = -0.58$ para SeCO, y $x_{-1} = 0.42$ y $x_0 = -0.58$ para EeCO

	Método SeCO					Método EeCO				
α	$\bar{x}$	$\lvert f_2(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_2(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_2$	3.66×10^{-15}	7.76×10^{-9}	6	2.00	$\bar{x}_2$	7.82×10^{-14}	6.38×10^{-9}	6	1.65
0.9	$\bar{x}_2$	5.81×10^{-13}	4.07×10^{-7}	6	2.00	$\bar{x}_2$	7.90×10^{-11}	3.89×10^{-7}	6	1.85
0.8	$\bar{x}_2$	6.09×10^{-10}	1.78×10^{-5}	6	2.01	$\bar{x}_2$	6.72×10^{-10}	1.37×10^{-6}	6	1.87
0.7	$\bar{x}_2$	1.99×10^{-15}	3.19×10^{-8}	7	2.00	$\bar{x}_2$	3.20×10^{-14}	2.82×10^{-9}	7	1.53
0.6	$\bar{x}_2$	8.03×10^{-11}	3.49×10^{-6}	7	2.00	$\bar{x}_2$	3.30×10^{-11}	2.22×10^{-7}	7	1.51
0.5	$\bar{x}_2$	1.78×10^{-15}	2.44×10^{-8}	8	2.00	$\bar{x}_2$	3.29×10^{-10}	6.08×10^{-7}	8	2.28
0.4	$\bar{x}_2$	5.14×10^{-10}	7.27×10^{-6}	8	2.00	$\bar{x}_2$	5.94×10^{-11}	2.33×10^{-7}	10	1.82
0.3	$\bar{x}_2$	1.48×10^{-12}	3.54×10^{-7}	9	1.99	$\bar{x}_1$	9.28×10^{-14}	3.11×10^{-9}	218	1.62
0.2	$\bar{x}_2$	1.42×10^{-14}	2.51×10^{-8}	10	2.00	$\bar{x}_2$	1.90×10^{-13}	6.36×10^{-9}	23	1.62
0.1	$\bar{x}_2$	1.65×10^{-14}	4.57×10^{-9}	11	2.00	$\bar{x}_1$	2.78×10^{-11}	1.02×10^{-7}	25	1.61

Tabla 6.2: Resultados para $f_2(x)$, con estimaciones iniciales $x_0 = -3.5$ para SeCO, y $x_{-1} = -2.5$ y $x_0 = -3.5$ para EeCO

	Método SeCO					Método EeCO				
α	$\bar{x}$	$\lvert f_3(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_3(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_1$	1.91×10^{-11}	4.38×10^{-6}	5	2.02	$\bar{x}_1$	9.81×10^{-11}	8.66×10^{-7}	6	1.59
0.9	$\bar{x}_1$	6.70×10^{-10}	2.43×10^{-5}	5	2.05	$\bar{x}_1$	1.45×10^{-9}	4.22×10^{-6}	6	1.57
0.8	$\bar{x}_1$	1.78×10^{-15}	2.77×10^{-8}	6	2.01	$\bar{x}_1$	2.59×10^{-13}	1.77×10^{-8}	7	1.66
0.7	$\bar{x}_1$	2.22×10^{-12}	1.21×10^{-6}	6	2.02	$\bar{x}_1$	1.61×10^{-11}	1.69×10^{-7}	7	1.69
0.6	$\bar{x}_1$	4.12×10^{-9}	4.82×10^{-5}	6	2.07	$\bar{x}_1$	3.11×10^{-10}	1.58×10^{-6}	7	1.75
0.5	$\bar{x}_1$	3.46×10^{-11}	4.05×10^{-6}	7	2.03	$\bar{x}_1$	4.60×10^{-9}	5.46×10^{-6}	7	1.85
0.4	$\bar{x}_1$	2.74×10^{-11}	3.30×10^{-6}	8	2.03	$\bar{x}_1$	1.27×10^{-12}	3.67×10^{-8}	8	1.50
0.3	$\bar{x}_1$	1.07×10^{-14}	5.68×10^{-8}	10	2.01	$\bar{x}_1$	1.71×10^{-11}	1.73×10^{-7}	8	1.44
0.2	$\bar{x}_1$	3.55×10^{-15}	1.06×10^{-8}	13	2.01	$\bar{x}_1$	1.23×10^{-10}	5.54×10^{-7}	8	1.36
0.1	$\bar{x}_1$	5.86×10^{-14}	1.17×10^{-7}	19	2.01	$\bar{x}_1$	5.50×10^{-10}	1.36×10^{-6}	8	1.26

Tabla 6.3: Resultados para $f_3(x)$, con estimaciones iniciales $x_0 = 0.5$ para SeCO, y $x_{-1} = 1.5$ y $x_0 = 0.5$ para EeCO

Nuestra segunda función es $f_2(x) = ix^{1.8} - x^{0.9} - 16$, con raíces complejas $\bar{x}_1 = 2.90807 - 4.24908i$ y $\bar{x}_2 = -3.85126 + 1.74602i$.

En la Tabla 6.2 notamos que para ambos procedimientos (SeCO y EeCO) el número de iteraciones puede ser igualado a sus respectivos casos clásicos; en general, el número de iteraciones va en aumento cuando el valor de α decrece. El orden de convergencia computacional es similar al caso clásico para cada método, y con cada esquema se encuentran raíces complejas partiendo de estimaciones iniciales reales. Con EeCO es posible encontrar distintas soluciones al cambiar el valor de α.

	Método SeCO					Método EeCO				
α	$\bar{x}$	$\lvert f_4(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ	$\bar{x}$	$\lvert f_4(x_{k+1})\rvert$	$\lvert x_{k+1}-x_k\rvert$	iter	ρ
1	$\bar{x}_3$	6.38×10^{-15}	5.41×10^{-9}	20	2.00	$\bar{x}_{10}$	4.48×10^{-12}	2.41×10^{-8}	17	1.20
0.9	$\bar{x}_{13}$	1.57×10^{-12}	7.60×10^{-8}	16	2.01	$\bar{x}_7$	1.46×10^{-10}	9.06×10^{-7}	10	0.74
0.8	$\bar{x}_4$	1.85×10^{-12}	7.12×10^{-8}	18	1.99	$\bar{x}_2$	1.60×10^{-11}	3.61×10^{-8}	11	1.31
0.7	$\bar{x}_1$	6.41×10^{-12}	1.39×10^{-7}	28	2.02	$\bar{x}_9$	6.89×10^{-10}	4.16×10^{-7}	24	1.38
0.6	$\bar{x}_1$	6.33×10^{-13}	3.88×10^{-8}	23	2.01	$\bar{x}_9$	6.14×10^{-11}	1.03×10^{-7}	23	1.14
0.5	$\bar{x}_{11}$	1.09×10^{-9}	1.17×10^{-6}	21	1.95	$\bar{x}_{10}$	2.32×10^{-12}	6.76×10^{-9}	20	1.81
0.4	$\bar{x}_1$	8.15×10^{-9}	3.49×10^{-6}	95	2.05	$\bar{x}_5$	4.45×10^{-13}	2.23×10^{-9}	14	1.98
0.3	-	-	-	-	-	$\bar{x}_4$	8.39×10^{-13}	3.87×10^{-9}	104	1.37
0.2	$\bar{x}_5$	7.17×10^{-12}	8.35×10^{-8}	66	1.47	$\bar{x}_{13}$	2.58×10^{-10}	8.57×10^{-8}	175	1.67
0.1	$\bar{x}_3$	7.92×10^{-14}	4.44×10^{-9}	58	2.00	-	-	-	>500	-

Tabla 6.4: Resultados para $f_4(x)$, con estimaciones iniciales $x_0=3$ para SeCO, y $x_{-1}=4$ y $x_0=3$ para EeCO

La tercera función es $f_3(x)=e^x-1$, con única raíz real $\bar{x}_1=0$.

En la Tabla 6.3 observamos que para ambos procedimientos el número de iteraciones puede ser igual al de sus respectivos casos clásicos; el número de iteraciones aumenta cuando α disminuye. El orden de convergencia computacional es similar al caso clásico para cada método y todo α.

Finalmente, nuestra cuarta función es $f_4(x)=\sin 10x-0.5x+0.2$, con raíces reales $\bar{x}_1=-1.4523$, $\bar{x}_2=-1.3647$, $\bar{x}_3=-0.87345$, $\bar{x}_4=-0.6857$, $\bar{x}_5=-0.27949$, $\bar{x}_6=-0.021219$, $\bar{x}_7=0.31824$, $\bar{x}_8=0.64036$, $\bar{x}_9=0.91636$, $\bar{x}_{10}=1.3035$, $\bar{x}_{11}=1.5118$, $\bar{x}_{12}=1.9756$ y $\bar{x}_{13}=2.0977$.

En la Tabla 6.4 vemos que SeCO y EeCO requieren menos iteraciones que el caso clásico, respectivamente para algunos vlores de α, y que ρ es similar al caso clásico en cada caso. No se muestran resultados para SeCO cuando $\alpha=0.3$ porque éste convergen a un punto que no es solución de $f_4(x)$; esto sucede porque a pesar de que $|x_{k+1}-x_k|<10^{-8}$, es decir, cumple con el criterio de parada, $|f(x_{k+1})|\gg10^{-8}$. Tampoco se muestran resultados para EeCO cuando $\alpha=0.1$, ya que éste requiere más de 500 iteraciones para converger. Podemos observar que con cada procedimiento se obtienen distintas raíces al modificar el valor de α.

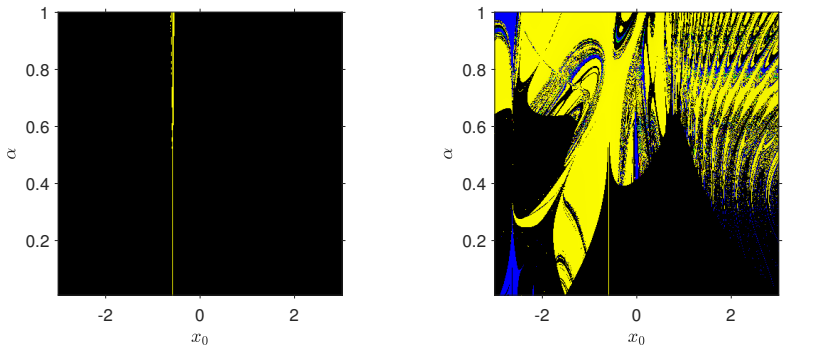
6.3.1. Estabilidad numérica

Ahora vamos a analizar la dependencia respecto a las estimaciones iniciales de los métodos diseñados en este capítulo; para ello, utilizamos los planos de convergencia definidos en [29] (ver ficheros MATLAB en Anexos A.38 y A.39). En estos planos, el eje de abscisas corresponde a la estimación inicial x_0 (para ambos esquemas), y el orden fraccionario de la derivada (α) aparece en el eje de las ordenadas. Se utiliza una malla de 400×400 puntos. Los puntos que no se representan en negro corresponden a los pares de estimaciones iniciales y valores de α que convergen a una de las raíces, con una tolerancia de 10^{-3}. Distintos colores significan convergencia a distintas raíces. Por lo tanto, cuando un punto se representa en negro, esto significa que el proceso iterativo no converge a ninguna raíz en un máximo de 500 iteraciones. Para todos los planos de convergencia calculamos el porcentaje de pares convergentes (x_0,α), con el fin de comparar la eficiencia de dichos procedimientos. Para cada método fijamos $a=-10$ en cada plano, y $\alpha\in(0,1]$. En el caso de EeCO seleccionamos $x_{-1}=x_0+1$ para realizar la primera iteración, tal y como hicimos en las pruebas numéricas de la Sección 6.3.

En la Figura 6.1 podemos observar que SeCO alcanza aproximadamente sólo un 0.5 % de convergencia, y se obtiene sólo una raíz, mientras que EeCO consigue alrededor de un 44 % de convergencia, y todas las raíces se encuentran.

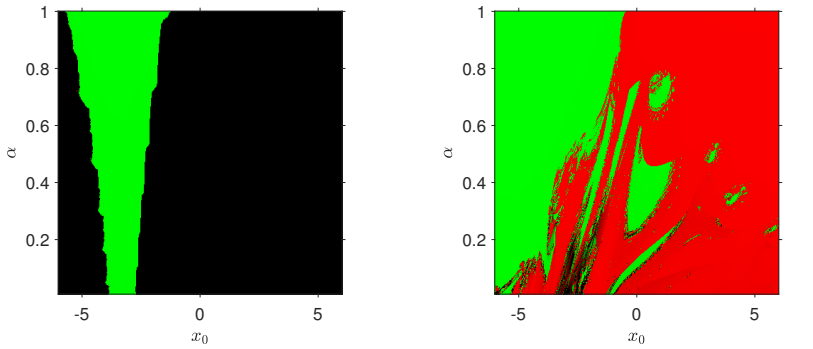
En la Figura 6.2 notamos que SeCO alcanza aproximadamente un 20 % de convergencia, mientras que EeCO consigue casi el 100 % de convergencia; ambas raíces se encuentran en cada plano.

En la Figura 6.3 podemos ver que SeCO alcanza un 10 % de convergencia, mientras que EeCO consigue alrededor de un 47 % de convergencia. En cada caso, considerando que $-10\le x_0\le10$, y que $a=-10$,



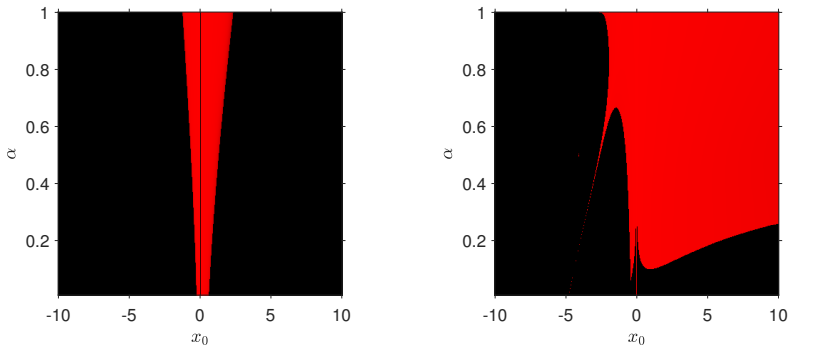
(a) SeCO, 0.43 % de convergencia **(b)** EeCO, 43.76 % de convergencia

Figura 6.1: Planos de convergencia para $f_1(x)$



(a) SeCO, 19.94 % de convergencia **(b)** EeCO, 99.29 % de convergencia

Figura 6.2: Planos de convergencia para $f_2(x)$



(a) SeCO, 10.00 % de convergencia **(b)** EeCO, 46.75 % de convergencia

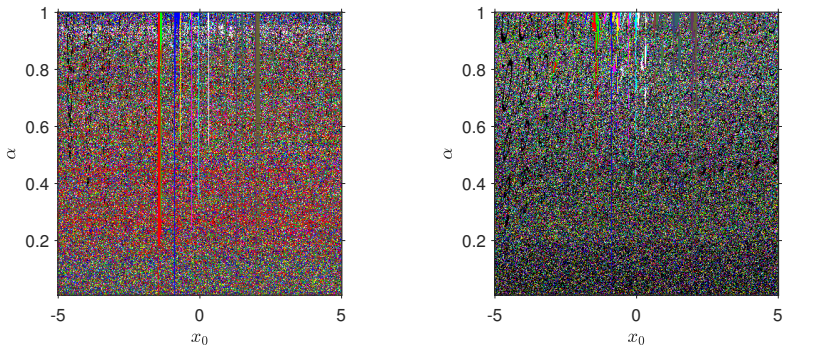
Figura 6.3: Planos de convergencia para $f_3(x)$

indicamos que la estabilidad podría ser ligeramente mejorada si escogemos $a < -10$.

En la Figura 6.4 observamos que SeCO alcanza aproximadamente un 93 % de convergencia, mientras que EeCO consigue alrededor de un 74 % de convergencia; todas las raíces se encuentran en cada plano.

Hemos visto que en su versión conformable, los esquemas libres de derivadas mantienen en líneas generales la estabilidad del procedimiento original, aunque con mejoras en algunos aspectos.

En este capítulo, vimos el diseño, análisis de convergencia y estudio de la estabilidad de los primeros esquemas fraccionarios libres de derivadas (de tipos Steffensen y Secante), donde se aproximan derivadas conforma-



(a) SeCO, 93.07 % de convergencia **(b)** EeCO, 73.69 % de convergencia

Figura 6.4: Planos de convergencia para $f_4(x)$

bles; fue posible utilizar la técnica general de la Sección 5.2 para conseguir la versión conformable de tales procedimientos clásicos. En la teoría, SeCO y EeCO preservan el orden de convergencia teórico de sus versiones clásicas, respectivamente. En la práctica, hemos observado que en algunos casos se puede converger en menos iteraciones que en el caso clásico, y ρ puede ser ligeramente superior. También hemos visto que, en general, es posible conseguir una raíz distinta al variar el valor del índice fraccionario α, y se puede converger a raíces complejas partiendo de estimaciones iniciales reales. En el siguiente capítulo, se proporcionan las conclusiones generales del trabajo desarrollado en esta memoria, y las líneas futuras de investigación.

Capítulo 7

Conclusiones y líneas futuras

"Siempre parece imposible... hasta que se hace"

Nelson Mandela (1918 - 2013)

En esta Tesis Doctoral, se han diseñado métodos fraccionarios con derivadas de Caputo, de Riemann-Liouville y conformable (o su aproximación) escalares; también se ha propuesto una técnica general para la obtención de la versión conformable de cualquier esquema clásico escalar.

En esta memoria se ha diseñado:

- El primer procedimiento fraccionario óptimo (Newton-Raphson conformable escalar) según la conjetura de Kung y Traub.
- Los primeros métodos fraccionarios de tipo Newton-Raphson (con derivada conformable escalar) de mayor orden.
- El primer esquema fraccionario vectorial (Newton-Raphson conformable vectorial).
- Los primeros procedimientos fraccionarios multipunto (Traub con derivadas de Caputo y Riemann-Liouvllie escalares, y Traub, Chun-Kim, Ostrowski y Chun conformables escalares).
- Los primeros métodos fraccionarios multipunto óptimos (Ostrowski y Chun conformables escalares).
- Los primeros esquemas fraccionarios libres de derivadas (Steffensen y Secante conformables escalares).
- El primer procedimiento fraccionario con memoria (Secante conformable escalar).

Se ha realizado el análisis de convergencia de todos estos procedimientos fraccionarios, donde se ha obtenido orden de convergencia inferior al caso clásico cuando se usan las derivadas de Caputo o Riemann-iouville en los esquemas, mientras que con el uso de derivadas conformables (o su aproximación) se conserva el orden de convergencia teórico de los métodos clásicos.

En la práctica, los procedimientos con derivadas conformables (o su aproximación) mostraron mucha superioridad comparados con aquellos con derivadas de Caputo o Riemann-Liouville en términos de cantidad de iteraciones requeridas, el orden de convergencia computacional y la amplitud de las cuencas de atracción de las raíces. Estos métodos con derivadas conformables también presentaron algunas ventajas frente a sus respectivos esquemas clásicos (cuando $\alpha = 1$):

- Éstos pueden encontrar solución cuando los métodos clásicos fallan (con $\alpha \neq 1$).
- En algunos casos es posible converger en menos iteraciones que en el caso clásico.

- El orden de convergencia computacional puede ser ligeramente superior.
- Éstos pueden conseguir una raíz diferente al modificar el valor del orden fraccionario α.
- También, pueden converger a raíces complejas partiendo de estimaciones iniciales reales.

Otras investigaciones que se pueden llevar a cabo son:

- El desarrollo de procedimientos fraccionarios con funciones peso: Nos preguntamos si con los métodos fraccionarios será posible obtener nuevas funciones peso que permitan aumentar la convergencia sin añadir nuevas evaluaciones funcionales.
- El estudio dinámico de los esquemas desarrollados en esta memoria: Nos planteamos poner a prueba diversos polinomios reales y complejos con diferentes valores de α, para estudiar el comportamiento asintótico de sus puntos fijos y ver cuál es el efecto resultante en la amplitud de las cuencas de atracción asociadas a dichos procedimientos.
- Aplicar métodos fraccionarios a la resolución de ecuaciones diferenciales en derivadas parciales fraccionarias: Si tal ecuación diferencial es no lineal, resolver el sistema no linieal generado con el uso de la versión vectorial del esquema conformable de Newton-Raphson.
- Adaptar estos procedimientos a la resolución de ecuaciones matriciales: En el caso de ser posible diseñar métodos fraccionarios para resolver ecuaciones matriciales particulares, podría ser necesario el uso del producto/potencia de Hadamard, y generalizar el Teorema del binomio a valores matriciales.
- El desarrollo de esquemas con derivadas conformables por la derecha: Dado que todos los procedimientos conformables desarrollados en esta memoria incluyen derivadas conformables por la izquierda, nos planteamos diseñarlos con derivadas conformables por la derecha, donde antes será necesario establecer una serie de Taylor con el uso de esta derivada.
- Demostrar que el orden de convergencia teórico siempre se mantiene en la versión conformable de los métodos clásicos: Estamos interesados en probar, en caso de ser posible, que con el uso de la técnica general que diseñamos para obtener la versión conformable de cualquier esquema clásico de orden p, podemos conseguir un procedimiento conformable de orden p.

Apéndice A

Anexos: Ficheros en MATLAB

A.1. Función Gamma

```
function [f] = Gamma(z)
% GAMMA  Gamma function valid in the entire complex plane.
%        Accuracy is 15 significant digits along the real axis
%        and 13 significant digits elsewhere.
%        This routine uses a superb Lanczos series
%        approximation for the complex Gamma function.
%
%        z may be complex and of any size.
%        Also  n! = prod(1:n) = gamma(n+1)
%
%usage: [f] = gamma(z)
%
%tested on versions 6.0 and 5.3.1 under Sun Solaris 5.5.1
%
%References: C. Lanczos, SIAM JNA  1, 1964. pp. 86-96
%            Y. Luke, "The Special ... approximations", 1969 pp. 29-31
%            Y. Luke, "Algorithms ... functions", 1977
%            J. Spouge,  SIAM JNA 31, 1994. pp. 931-944
%            W. Press,  "Numerical Recipes"
%            S. Chang, "Computation of special functions", 1996
%            W. J. Cody "An Overview of Software Development for Special
%            Functions", 1975
%
%see also:   GAMMA GAMMALN GAMMAINC PSI
%see also:   mhelp GAMMA
%
%Paul Godfrey
%pgodfrey@intersil.com
%http://winnie.fit.edu/~gabdo/gamma.txt
%Sept 11, 2001

siz = size(z);
z=z(:);
zz=z;

f = 0.*z; % reserve space in advance

p=find(real(z)<0);
if ~isempty(p)
    z(p)=-z(p);
end
```

```
% 15 sig. digits for 0<=real(z)<=171
% coeffs should sum to about g*g/2+23/24

g=607/128; % best results when 4<=g<=5

c = [  0.99999999999999709182;
      57.156235665862923517;
     -59.597960355475491248;
      14.136097974741747174;
      -0.49191381609762019978;
        .33994649984811888699e-4;
        .46523628927048575665e-4;
       -.98374475304879564677e-4;
        .15808870322491248884e-3;
       -.21026444172410488319e-3;
        .21743961811521264320e-3;
       -.16431810653676389022e-3;
        .84418223983852743293e-4;
       -.26190838401581408670e-4;
        .36899182659531622704e-5];

%Num Recipes used g=5 with 7 terms
%for a less effective approximation

z=z-1;
zh =z+0.5;
zgh=zh+g;
%trick for avoiding FP overflow above z=141
zp=zgh.^(zh*0.5);

ss=0.0;
for pp=size(c,1)-1:-1:1
    ss=ss+c(pp+1)./(z+pp);
end

%sqrt(2Pi)
sq2pi=  2.50662827463100050241576528481l0;
f=(sq2pi*(c(1)+ss)).*((zp.*exp(-zgh)).*zp);

f(z==0 | z==1) = 1.0;

%adjust for negative real parts
if ~isempty(p)
   f(p)=-pi./(zz(p).*f(p).*sin(pi*zz(p)));
end

%adjust for negative poles
p=find(round(zz)==zz & imag(zz)==0 & real(zz)<=0);
if ~isempty(p)
   f(p)=Inf;
end

f=reshape(f,siz);

return
```

A.2. Función de Mittag-Leffler

```
function [e]=mlf(alf,bet,c,fi)
%
% MLF -- Mittag-Leffler function.
%        MLF(alpha,beta,Z,P) is the Mittag-Leffler function E_{alpha,beta}(Z)
%        evaluated with accuracy 10^(-P) for each element of Z.
%        alpha and beta are scalars, P is integer, Z can be a vector or
%        a two-dimensional array. The output is of the same size as Z.
% (C) 2001-2012 Igor Podlubny, Martin Kacenak
% Last update: 2012-09-07


[cRows, cCols] = size(c);
c=m2v(c);

if nargin<4, fi=6; end
if nargin<3 || alf<=0 || fi<=0
else
   [r,s]=size(c); [r1,s1]=size(alf); [r2,s2]=size(bet);
   mx=max([r,s]); mx1=max([r1,s1]); mx2=max([r2,s2]);
   if (r>1 && s>1) || (r1>1 && s1>1) || (r2>1 && s2>1) || (mx1>1 && mx2>1)
      sprintf('wrong number of input parameters')
   else
      if mx1>mx2, mxx=mx1; e=zeros(mx,mx1);
      else mxx=mx2; e=zeros(mx,mx2);end;
      for i1= 1:mx
         for i2=1:mxx

            if r>s, z=c(i1,1); else z=c(1,i1); end
            if mx1>mx2, if r1>s1, alfa=alf(i2,1); else alfa=alf(1,i2);end, beta=bet;
            else if r2>s2 ,beta=bet(i2,1); else beta=bet(1,i2); end, alfa=alf; end
            if beta<0, rc=(-2*log(10^(-fi)*pi/(6*(abs(beta)+2)*(2*abs(beta))^...
                  (abs(beta)))))^alfa;
               else rc=(-2*log(10^(-fi)*pi/6))^alfa; end
            r0=max([1,2*abs(z),rc]);
            if (alfa==1 && beta==1)
               e(i1,i2)=exp(z);
            else
               if (alfa<1 && abs(z)<=1) || ( (1<=alfa && alfa <2) && abs(z)<=...
                     floor(20/(2.1-alfa)^(5.5-2*alfa))) || (alfa>=2 && abs(z)<=50)
                  oldsum=0;
                  k=0;
                  while (alfa*k+beta)<=0
                     k=k+1;
                  end
                  newsum=z^k/Gamma(alfa*k+beta);
                  while newsum~=oldsum
                     oldsum=newsum;
                     k=k+1;
                     term=z^k/Gamma(alfa*k+beta);
                     newsum=newsum+term;
                     k=k+1;
                     term=z^k/Gamma(alfa*k+beta);
                     newsum=newsum+term;
                  end
                  e(i1,i2)=newsum;
               else
                  if (alfa<=1 && abs(z)<=fix(5*alfa+10))
                     if ((abs(angle(z))>pi*alfa) && (abs(abs(angle(z))-...
```

```
                    (pi*alfa))>10^(-fi)))
                if beta<=1
                    e(i1,i2)=rombint('K',0,r0,fi,alfa,beta,z);
                else
                  eps=1;
                  e(i1,i2)=rombint('K',eps,r0,fi,alfa,beta,z)+ ...
                     rombint('P',-pi*alfa,pi*alfa,fi,alfa,beta,z,eps);
               end
            elseif (abs(angle(z))<pi*alfa && abs(abs(angle(z))-...
                    (pi*alfa))>10^(-fi))
                if beta<=1
                    e(i1,i2)=rombint('K',0,r0,fi,alfa,beta,z)+ ...
                       (z^((1-beta)/alfa))*(exp(z^(1/alfa))/alfa);
                else
                    eps=abs(z)/2;
                    e(i1,i2)=rombint('K',eps,r0,fi,alfa,beta,z)+ ...
                       rombint('P',-pi*alfa,pi*alfa,fi,alfa,beta,z,eps)+ ...
                       (z^((1-beta)/alfa))*(exp(z^(1/alfa))/alfa);
                end
            else
                eps=abs(z)+0.5;
                e(i1,i2)=rombint('K',eps,r0,fi,alfa,beta,z)+ ...
                    rombint('P',-pi*alfa,pi*alfa,fi,alfa,beta,z,eps);
            end
        else
            if alfa<=1
                if (abs(angle(z))<(pi*alfa/2+min(pi,pi*alfa))/2)
                    % alfa
                    newsum=(z^((1-beta)/alfa))*exp(z^(1/alfa))/alfa;
                    for k=1:floor(fi/log10(abs(z)))
                        newsum=newsum-((z^(-k))/Gamma(beta-alfa*k));
                        % k
                    end
                    e(i1,i2)=newsum;
                else
                    newsum=0;
                    for k=1:floor(fi/log10(abs(z)))
                        newsum=newsum-((z^-k)/Gamma(beta-alfa*k));
                    end
                    e(i1,i2)=newsum;
                end
            else
                if alfa>=2
                    m=floor(alfa/2);
                    sum=0;
                    for h=0:m
                        zn=(z^(1/(m+1)))*exp((2*pi*1i*h)/(m+1));
                        sum=sum+mlf(alfa/(m+1),beta,zn,fi);
                    end
                    e(i1,i2)=(1/(m+1))*sum;
                else
                    e(i1,i2)=(mlf(alfa/2,beta,z^(1/2),fi)+...
                        mlf(alfa/2,beta,-z^(1/2),fi))/2;
                end
            end
        end
    end
  end
end
```

```
end

if isreal(c)
    e = real(e);
end

e = v2m(e,cRows,cCols);


function [res]=rombint(funfcn,a,b,order,varargin)
if nargin<4 ,order=6; end
if nargin<3
   Warning ('Error in input format')
else
   rom=zeros(2,order);
   h=b-a;
   rom(1,1)=h*(feval(funfcn,a,varargin{:})+feval(funfcn,b,varargin{:}))/2;

   ipower=1;
   for i= 2:order
      sum=0;
      for j=1:ipower
        sum=sum+feval(funfcn,(a+h*(j-0.5)),varargin{:});
      end
      rom(2,1)=(rom(1,1)+h*sum)/2;
      for k=1:i-1
         rom(2,k+1)=((4^k)*rom(2,k)-rom(1,k))/((4^k)-1);
      end

      for j=0:i-1
         rom(1,j+1)=rom(2,j+1);
      end
      ipower=ipower*2;
      h=h/2;
   end
   res=rom(1,order);
end


function res=K(r,alfa,beta,z)
res=r.^((1-beta)/alfa).*exp(-r.^(1/alfa)).*(r*sin(pi*(1-beta))-...
z*sin(pi*(1-beta+alfa)))/(pi*alfa*(r.^2-2*r*z*cos(pi*alfa)+z.^2));


function res=P(r,alfa,beta,z,eps)
w=(eps^(1/alfa))*sin(r/alfa)+r*(1+(1-beta)/alfa);
res=((eps^(1+(1-beta)/alfa))/(2*pi*alfa))*((exp((eps^(1/alfa))*cos(r/alfa)).*...
   (cos(w)+1i*sin(w))))/(eps*exp(1i*r)-z);

function A = v2m(V, M, N)
if numel(V)==M*N,
    A = reshape(V, [N, M]);
    A = A' ;
else
    warning('Wrong dimensions of the output in V2M.')
end

function V = m2v(A)
M = A'; V = M(:);
```

A.3. Gráfica de la función Gamma

```
clf, clear all, close all, clc
a = -4;
b = 4;
z = a:0.01:b;
G = Gamma(z);
x = zeros(length(z));
plot(z, G, 'b', z, x, 'k', x, z, 'k--', x - 1, z, 'k--', x - 2, z, 'k--',...
    x - 3, z, 'k--')
xlim([a, b])
ylim([a, b])
xlabel('$z$','Interpreter','latex');
ylabel('$\Gamma(z)$','Interpreter','latex');
savefig('C2_1_fun_gamma_fig.fig')
```

A.4. Gráfica de la función de Mittag-Leffler

```
clf, clear all, close all, clc
a = -10;
b = 10;
z = a:0.01:b;
x = zeros(length(z));
for alpha = 1:0.5:2
    E = mlf(alpha, 1, z, 9);
    plot(z, E)
    hold on
end
plot(z, x, 'k', x, z, 'k')
xlim([a, 5])
ylim([-2, 5])
legend('E_1(z)', 'E_{1.5}(z)', 'E_2(z)')
xlabel('$z$','Interpreter','latex');
ylabel('$E_{\mu}(z)$','Interpreter','latex');
hold off
savefig('C2_2_fun_ML_fig.fig')
```

A.5. Plano de convergencia con el método de Traub paramétrico

```
% Plano de convergencia: f(x) = x^2 - 1
clf, clear all, close all, clc
tic
numNodosx = 400; numNodosy = 400;
maxIter = 500;
tol = 0.001;

f = @(x) x^2 - 1;
fp = @(x) 2*x;

r1 = 1; r2 = -1;

n = @(x) x - f(x)/fp(x);
traub = @(x, theta) n(x) - theta*f(n(x))/fp(x);

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3; xfin = 3; yini = 0.01; yfin = 1;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;

[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);
R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));
[rows, cols] = size(X);
noConv = 0; % inicio contador de puntos que no convergieron

for i = 1:rows
    for j = 1:cols
        x = X(i, j);
        theta = Y(i, j);
        iter = 0;
        e = Inf;
        while iter < maxIter && e > tol
            an = x;
            x = traub(x, theta); % traub_amortiguado
            e = abs(an - x);
            iter = iter + 1;
        end
        d1 = abs(r1 - x); % distancia a las raices
        d2 = abs(r2 - x);
        if d1 < tol % convergio a la primera raiz
            R(i, j) = 1 - iter/maxIter;
        elseif d2 < tol % convergio a la segunda raiz
            G(i, j) = 1 - iter/maxIter;
        else
            noConv = noConv + 1;
        end
    end
end
I(:, :, 1) = R(:, :);
I(:, :, 2) = G(:, :);
I(:, :, 3) = B(:, :);

figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
xlabel('$x_0$','Interpreter','latex');
ylabel('$\theta$','Interpreter','latex');
axis on, axis xy, axis square
densidad = ((numPuntos - noConv)/numPuntos)*100; % porciento exito
t = toc;
```

```
savefig('C2_3_traub_param_fig.fig')
save('C2_3_traub_param_mat.mat')
```

A.6. Método NeCA

```
function [x, err1, err2, iter, ACOC] = pruebas_NeCA(x0, alpha)
f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % Funcion 1
df = @(x) ((Gamma(7)*-12.84*x^(6 - alpha))/Gamma(7 - alpha)) +...
    ((Gamma(6)*-25.6*x^(5 - alpha))/Gamma(6 - alpha)) +...
    ((Gamma(5)*16.55*x^(4 - alpha))/Gamma(5 - alpha)) +...
    ((Gamma(4)*-2.21*x^(3 - alpha))/Gamma(4 - alpha)) +...
    ((Gamma(3)*26.71*x^(2 - alpha))/Gamma(3 - alpha)) +...
    ((Gamma(2)*-4.29*x^(1 - alpha))/Gamma(2 - alpha)); % Derivada de Caputo 1
% f = @(x) 1i*x^1.8 - x^0.9 - 16; % Funcion 2
% df = @(x) ((Gamma(2.8)*1i*x^(1.8 - alpha))/Gamma(2.8 - alpha)) +...
%     ((Gamma(1.9)*-1*x^(0.9 - alpha))/Gamma(1.9 - alpha)); % Derivada de Caputo 2
% f = @(x) exp(x) - 1; % Funcion 3
% df = @(x) x^(1 - alpha)*mlf(1, 2 - alpha, x, 9); % Derivada de Caputo 3
% f = @(x) sin(10*x) - 0.5*x + 0.2; % Funcion 4
% df = @(x) 5*x^(1 - alpha)*(mlf(1, 2 - alpha, 10i*x, 9) +...
%     mlf(1, 2 - alpha, -10i*x, 9)) +...
%     ((Gamma(2)*-0.5*x^(1 - alpha))/Gamma(2 - alpha)); % Derivada de Caputo 4
err1 = Inf;
err2 = Inf;
x = x0*ones(1);
ACOC = Inf*ones(1);
iter = 0;
while iter < 500 && err1 > 1e-8 && err2 > 1e-8
    x(iter + 2) = x(end) - (Gamma(alpha + 1)*(f(x(end))/df(x(end))))^(1/alpha);
    err1 = abs(f(x(end)));
    err2 = abs(x(end) - x(end - 1));
    if iter >= 2
        ACOC(iter + 2) = (log(abs((x(end) - x(end - 1)))/abs((x(end - 1)...
            - x(end - 2)))))/(log(abs((x(end - 1) - x(end - 2)))...
            /abs((x(end - 2) - x(end - 3)))));
    elseif iter < 2
        ACOC(iter + 2) = Inf;
    end
    iter = iter + 1;
end
x = x(end);
ACOC(1) = [];
t = 1:iter;
plot(t, ACOC), grid
xlabel('Iteraciones')
ylabel('Orden de convergencia')
end
% Ejemplo de uso:
% [x, err1, err2, iter, ACOC] = pruebas_NeCA(1, 0.5)
```

A.7. Método NeRL

```
function [x, err1, err2, iter, ACOC] = pruebas_NeRL(x0, alpha)
f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % Funcion 1
df = @(x) ((Gamma(7)*-12.84*x^(6 - alpha))/Gamma(7 - alpha)) +...
    ((Gamma(6)*-25.6*x^(5 - alpha))/Gamma(6 - alpha)) +...
    ((Gamma(5)*16.55*x^(4 - alpha))/Gamma(5 - alpha)) +...
    ((Gamma(4)*-2.21*x^(3 - alpha))/Gamma(4 - alpha)) +...
    ((Gamma(3)*26.71*x^(2 - alpha))/Gamma(3 - alpha)) +...
    ((Gamma(2)*-4.29*x^(1 - alpha))/Gamma(2 - alpha)) +...
    (-15.21/Gamma(1 - alpha))*x^-alpha; % Derivada de Riemann-Liouville 1
% f = @(x) 1i*x^1.8 - x^0.9 - 16; % Funcion 2
% df = @(x) ((Gamma(2.8)*1i*x^(1.8 - alpha))/Gamma(2.8 - alpha)) +...
%     ((Gamma(1.9)*-1*x^(0.9 - alpha))/Gamma(1.9 - alpha)) +...
%     (-16/Gamma(1 - alpha))*x^-alpha; % Derivada de Riemann-Liouville 2
% f = @(x) exp(x) - 1; % Funcion 3
% df = @(x) x^(1 - alpha)*mlf(1, 2 - alpha, x, 9) +...
%     (-1/Gamma(1 - alpha))*x^-alpha; % Derivada de Riemann-Liouville 3
% f = @(x) sin(10*x) - 0.5*x + 0.2; % Funcion 4
% df = @(x) 5*x^(1 - alpha)*(mlf(1, 2 - alpha, 10i*x, 9) +...
%     mlf(1, 2 - alpha, -10i*x, 9)) +...
%     ((Gamma(2)*-0.5*x^(1 - alpha))/Gamma(2 - alpha)) +...
%     (0.2/Gamma(1 - alpha))*x^-alpha; % Derivada de Riemann-Liouville 4
err1 = Inf;
err2 = Inf;
x = x0*ones(1);
ACOC = Inf*ones(1);
iter = 0;
while iter < 500 && err1 > 1e-8 && err2 > 1e-8
    x(iter + 2) = x(end) - (Gamma(alpha + 1)*(f(x(end))/df(x(end))))^(1/alpha);
    err1 = abs(f(x(end)));
    err2 = abs(x(end) - x(end - 1));
    if iter >= 2
        ACOC(iter + 2) = (log(abs((x(end) - x(end - 1)))/abs((x(end - 1)...
            - x(end - 2)))))/(log(abs((x(end - 1) - x(end - 2)))...
            /abs((x(end - 2) - x(end - 3)))));
    elseif iter < 2
        ACOC(iter + 2) = Inf;
    end
    iter = iter + 1;
end
x = x(end);
ACOC(1) = [];
t = 1:iter;
plot(t, ACOC), grid
xlabel('Iteraciones')
ylabel('Orden de convergencia')
end
% Ejemplo de uso:
% [x, err1, err2, iter, ACOC] = pruebas_NeRL(1, 0.5)
```

A.8. Método NeCO

```
function [x, err1, err2, iter, ACOC] = pruebas_NeCO(x0, alpha, a)
f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % Funcion 1
df = @(x) (x - a)^(1 - alpha)*(-(1926*x^5)/25 - 128*x^4 + (331*x^3)/5 -...
    (663*x^2)/100 + (2671*x)/50 - 429/100); % Derivada conformable 1
% f = @(x) 1i*x^1.8 - x^0.9 - 16; % Funcion 2
% df = @(x) (x - a)^(1 - alpha)*(1.8*1i*x^0.8 - 0.9*x^-0.1);
% Derivada conformable 2
% f = @(x) exp(x) - 1; % Funcion 3
% df = @(x) (x - a)^(1-alpha)*exp(x); % Derivada conformable 3
% f = @(x) sin(10*x) - 0.5*x + 0.2; % Funcion 4
% df = @(x) (x - a)^(1-alpha)*(10*cos(10*x)-0.5); % Derivada conformable 4
err1 = Inf;
err2 = Inf;
x = x0*ones(1);
ACOC = Inf*ones(1);
iter = 0;
while iter < 500 && err1 > 1e-8 && err2 > 1e-8
    x(iter + 2) = a + ((x(end) - a)^alpha - alpha*f(x(end))/df(x(end)))^(1/alpha);
    err1 = abs(f(x(end)));
    err2 = abs(x(end) - x(end - 1));
    if iter >= 2
        ACOC(iter + 2) = (log(abs((x(end) - x(end - 1)))/abs((x(end - 1)...
            - x(end - 2)))))/(log(abs((x(end - 1) - x(end - 2)))...
            /abs((x(end - 2) - x(end - 3)))));
    elseif iter < 2
        ACOC(iter + 2) = Inf;
    end
    iter = iter + 1;
end
x = x(end);
ACOC(1) = [];
t = 1:iter;
plot(t, ACOC), grid
xlabel('Iteraciones')
ylabel('Orden_de_convergencia')
end
% Ejemplo de uso:
% [x, err1, err2, iter, ACOC] = pruebas_NeCO(1, 0.5, -10)
```

A.9. Método NeA3

```
function [iter, x, err1, err2, ACOC] = pruebas_NeA3(x0, alpha)
f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % Funcion 1
fp = @(x) -(1926*x^5)/25 - 128*x^4 + (331*x^3)/5 -...
    (663*x^2)/100 + (2671*x)/50 - 429/100; % Primera derivada 1
fpp = @(x) -(1926*x^4)/5 - 512*x^3 + (993*x^2)/5 - (663*x)/50 + 2671/50;
% Segunda derivada 1
df = @(x, a) (x - a)^(1 - alpha)*(-(1926*x^5)/25 - 128*x^4 + (331*x^3)/5 -...
    (663*x^2)/100 + (2671*x)/50 - 429/100); % Derivada conformable 1
% f = @(x) 1i*x^1.8 - x^0.9 - 16; % Funcion 2
% fp = @(x) 1.8*1i*x^0.8 - 0.9*x^-0.1; % Primera derivada 2
% fpp = @(x) 0.8*1.8*1i*x^-0.2 + 0.1*0.9*x^-1.1; % Segunda derivada 2
% df = @(x, a) (x - a)^(1 - alpha)*(1.8*1i*x^0.8 - 0.9*x^-0.1);
% Derivada conformable 2
% f = @(x) exp(x) - 1; % Funcion 3
% fp = @(x) exp(x); % Primera derivada 3
% fpp = @(x) exp(x); % Segunda derivada 3
% df = @(x, a) (x - a)^(1-alpha)*exp(x); % Derivada conformable 3
% f = @(x) sin(10*x) - 0.5*x + 0.2; % Funcion 4
% fp = @(x) 10*cos(10*x)-0.5; % Primera derivada 4
% fpp = @(x) -100*sin(10*x); % Segunda derivada 4
% df = @(x, a) (x - a)^(1-alpha)*(10*cos(10*x)-0.5);
% Derivada conformable 4
err1 = Inf;
err2 = Inf;
x = x0*ones(1);
ACOC = Inf*ones(1);
iter = 0;
while iter < 500 && err1 > 1e-8 && err2 > 1e-8
    fx = f(x(end));
    a = x(end) + (1 - alpha)*fp(x(end))/fpp(x(end));
    x(iter + 2) = a + ((x(end) - a)^alpha - alpha*fx/df(x(end), a))^(1/alpha);
    err1 = abs(f(x(end)));
    err2 = abs(x(end) - x(end - 1));
    if iter >= 2
        ACOC(iter + 2) = (log(abs((x(end) - x(end - 1)))/abs((x(end - 1)...
            - x(end - 2)))))/(log(abs((x(end - 1) - x(end - 2)))...
            /abs((x(end - 2) - x(end - 3)))));
    elseif iter < 2
        ACOC(iter + 2) = Inf;
    end
    iter = iter + 1;
end
x = x(end);
ACOC(1) = [];
t = 1:iter;
plot(t, ACOC), grid
xlabel('Iteraciones')
ylabel('Orden de convergencia')
end
% Ejemplo de uso:
% [iter, x, err1, err2, ACOC] = pruebas_NeA3(1, 0.5)
```

A.10. Método NeL3

```
function [iter, x, err1, err2, ACOC] = pruebas_NeL3(x0, a)
f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % Funcion 1
fp = @(x) -(1926*x^5)/25 - 128*x^4 + (331*x^3)/5 -...
    (663*x^2)/100 + (2671*x)/50 - 429/100; % Primera derivada 1
fpp = @(x) -(1926*x^4)/5 - 512*x^3 + (993*x^2)/5 - (663*x)/50 + 2671/50;
% Segunda derivada 1
df = @(x, alpha) (x - a)^(1 - alpha)*(-(1926*x^5)/25 - 128*x^4 + (331*x^3)/5 -...
    (663*x^2)/100 + (2671*x)/50 - 429/100); % Derivada conformable 1
% f = @(x) 1i*x^1.8 - x^0.9 - 16; % Funcion 2
% fp = @(x) 1.8*1i*x^0.8 - 0.9*x^-0.1; % Primera derivada 2
% fpp = @(x) 0.8*1.8*1i*x^-0.2 + 0.1*0.9*x^-1.1; % Segunda derivada 2
% df = @(x, alpha) (x - a)^(1 - alpha)*(1.8*1i*x^0.8 - 0.9*x^-0.1);
% Derivada conformable 2
% f = @(x) exp(x) - 1; % Funcion 3
% fp = @(x) exp(x); % Primera derivada 3
% fpp = @(x) exp(x); % Segunda derivada 3
% df = @(x, alpha) (x - a)^(1-alpha)*exp(x); % Derivada conformable 3
% f = @(x) sin(10*x) - 0.5*x + 0.2; % Funcion 4
% fp = @(x) 10*cos(10*x)-0.5; % Primera derivada 4
% fpp = @(x) -100*sin(10*x); % Segunda derivada 4
% df = @(x, alpha) (x - a)^(1-alpha)*(10*cos(10*x)-0.5);
% Derivada conformable 4
err1 = Inf;
err2 = Inf;
x = x0*ones(1);
ACOC = Inf*ones(1);
iter = 0;
while iter < 500 && err1 > 1e-8 && err2 > 1e-8
    fx = f(x(end));
    alpha = 1 + (x(end) - a)*fpp(x(end))/fp(x(end));
    x(iter + 2) = a + ((x(end) - a)^alpha - alpha*fx/df(x(end),alpha))^(1/alpha);
    err1 = abs(f(x(end)));
    err2 = abs(x(end) - x(end - 1));
    if iter >= 2
        ACOC(iter + 2) = (log(abs((x(end) - x(end - 1)))/abs((x(end - 1)...
            - x(end - 2)))))/(log(abs((x(end - 1) - x(end - 2)))...
            /abs((x(end - 2) - x(end - 3)))));
    elseif iter < 2
        ACOC(iter + 2) = Inf;
    end
    iter = iter + 1;
end
x = x(end);
ACOC(1) = [];
t = 1:iter;
plot(t, ACOC), grid
xlabel('Iteraciones')
ylabel('Orden de convergencia')
end
% Ejemplo de uso:
% [iter, x, err1, err2, ACOC] = pruebas_NeL3(1, -10)
```

A.11. Método NeLA4

```
function [iter, x, err1, err2, ACOC] = pruebas_NeLA4(x0)
f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % Funcion 1
fp = @(x) -(1926*x^5)/25 - 128*x^4 + (331*x^3)/5 -...
    (663*x^2)/100 + (2671*x)/50 - 429/100; % Primera derivada 1
fpp = @(x) -(1926*x^4)/5 - 512*x^3 + (993*x^2)/5 -...
    (663*x)/50 + 2671/50; % Segunda derivada 1
fppp = @(x) -4*(1926*x^3)/5 - 3*512*x^2 + 2*(993*x)/5 - 663/50;
% Tercera derivada 1
df = @(x, alpha, a) (x - a)^(1 - alpha)*(-(1926*x^5)/25 -...
    128*x^4 + (331*x^3)/5 - (663*x^2)/100 + (2671*x)/50 - 429/100);
% Derivada conformable 1
% f = @(x) 1i*x^1.8 - x^0.9 - 16; % Funcion 2
% fp = @(x) 1.8*1i*x^0.8 - 0.9*x^-0.1; % Primera derivada 2
% fpp = @(x) 0.8*1.8*1i*x^-0.2 + 0.1*0.9*x^-1.1; % Segunda derivada 2
% fppp = @(x) -0.2*0.8*1.8*1i*x^-1.2 - 1.1*0.9*x^-2.1; % Tercera derivada 2
% df = @(x, alpha, a) (x - a)^(1 - alpha)*(1.8*1i*x^0.8 - 0.9*x^-0.1);
% Derivada conformable 2
% f = @(x) exp(x) - 1; % Funcion 3
% fp = @(x) exp(x); % Primera derivada 3
% fpp = @(x) exp(x); % Segunda derivada 3
% fppp = @(x) exp(x); % Tercera derivada 3
% df = @(x, alpha, a) (x - a)^(1-alpha)*exp(x); % Derivada conformable 3
% f = @(x) sin(10*x) - 0.5*x + 0.2; % Funcion 4
% fp = @(x) 10*cos(10*x)-0.5; % Primera derivada 4
% fpp = @(x) -100*sin(10*x); % Segunda derivada 4
% fppp = @(x) -1000*cos(10*x); % Tercera derivada 4
% df = @(x, alpha, a) (x - a)^(1-alpha)*(10*cos(10*x)-0.5);
% Derivada conformable 4
err1 = Inf;
err2 = Inf;
x = x0*ones(1);
ACOC = Inf*ones(1);
iter = 0;
while iter < 500 && err1 > 1e-8 && err2 > 1e-8
    fx = f(x(end));
    c2 = (1/2)*fpp(x(end))/fp(x(end));
    c3 = (1/6)*fppp(x(end))/fp(x(end));
    alpha = (3*c3 - 4*c2^2)/(3*c3 - 2*c2^2);
    a = x(end) + c2/(3*c3 - 2*c2^2);
    x(iter + 2) = a + ((x(end) - a)^alpha - alpha*fx/df(x(end),alpha,a))^(1/alpha);
    err1 = abs(f(x(end)));
    err2 = abs(x(end) - x(end - 1));
    if iter >= 2
        ACOC(iter + 2) = (log(abs((x(end) - x(end - 1)))/abs((x(end - 1)...
            - x(end - 2)))))/(log(abs((x(end - 1) - x(end - 2)))...
            /abs((x(end - 2) - x(end - 3)))));
    elseif iter < 2
        ACOC(iter + 2) = Inf;
    end
    iter = iter + 1;
end
x = x(end);
ACOC(1) = [];
t = 1:iter;
plot(t, ACOC), grid
xlabel('Iteraciones')
ylabel('Orden de convergencia')
```

```
end
% Ejemplo de uso:
% [iter, x, err1, err2, ACOC] = pruebas_NeLA4(1)
```

A.12. Plano de convergencia para el método NeCA (ejemplo con $f_1(x)$)

```
% Plano de convergencia: f(x) = -12.84*x^6 - 25.6*x^5 + 16.55*x^4 -
% 2.21*x^3 + 26.71*x^2 - 4.29*x - 15.21
clf, clear all, close all, clc
tic
numNodosx = 400; numNodosy = 400;
maxIter = 500;
tol = 0.001;

f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % problema a resolver
fp = @(x, alpha) ((Gamma(7)*-12.84*x^(6 - alpha))/Gamma(7 - alpha)) +...
    ((Gamma(6)*-25.6*x^(5 - alpha))/Gamma(6 - alpha)) +...
    ((Gamma(5)*16.55*x^(4 - alpha))/Gamma(5 - alpha)) +...
    ((Gamma(4)*-2.21*x^(3 - alpha))/Gamma(4 - alpha)) +...
    ((Gamma(3)*26.71*x^(2 - alpha))/Gamma(3 - alpha)) +...
    ((Gamma(2)*-4.29*x^(1 - alpha))/Gamma(2 - alpha));

r1 = 0.82366 + 0.24769i; r2 = 0.82366 - 0.24769i;
r3 = -2.62297; r4 = -0.584;
r5 = -0.21705 + 0.99911i; r6 = -0.21705 - 0.99911i;

n = @(x, alpha) x - (Gamma(alpha + 1)*(f(x)/fp(x, alpha)))^(1/alpha);

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3; xfin = 3; yini = 0.01; yfin = 1;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;

[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);
R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));
[rows, cols] = size(X);
noConv = 0; % inicio contador de puntos que no convergieron

for i = 1:rows
    for j = 1:cols
        x = X(i, j);
        alpha = Y(i, j);
        iter = 0;
        e = Inf;
        while iter < maxIter && e > tol
            an = x;
            x = n(x, alpha); % Newton_Caputo
            e = abs(an - x);
            iter = iter + 1;
        end
        d1 = abs(r1 - x); % distancia a las raices
        d2 = abs(r2 - x);
        d3 = abs(r3 - x);
        d4 = abs(r4 - x);
        d5 = abs(r5 - x);
        d6 = abs(r6 - x);
        if d1 < tol % convergio a la primera raiz
            R(i, j) = 1 - iter/maxIter;
        elseif d2 < tol % convergio a la segunda raiz
            G(i, j) = 1 - iter/maxIter;
        elseif d3 < tol % convergio a la tercera raiz
```

```
            B(i, j) = 1 - iter/maxIter;
        elseif d4 < tol % convergio a la cuarta raiz
            R(i, j) = 1 - iter/maxIter;
            G(i, j) = 1 - iter/maxIter;
        elseif d5 < tol % convergio a la quinta raiz
            R(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        elseif d6 < tol % convergio a la sexta raiz
            G(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        else
            noConv = noConv + 1;
        end
    end
end
I(:, :, 1) = R(:, :);
I(:, :, 2) = G(:, :);
I(:, :, 3) = B(:, :);

figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
xlabel('$x_0$','Interpreter','latex');
ylabel('$\alpha$','Interpreter','latex');
axis on, axis xy, axis square
densidad = ((numPuntos - noConv)/numPuntos)*100; % porciento exito
t = toc;
savefig('C3_1_newton_caputo_f1_fig.fig')
save('C3_1_newton_caputo_f1_mat.mat')
```

A.13. Plano de convergencia para el método NeRL (ejemplo con $f_1(x)$)

```
% Plano de convergencia: f(x) = -12.84*x^6 - 25.6*x^5 + 16.55*x^4 -
% 2.21*x^3 + 26.71*x^2 - 4.29*x - 15.21
clf, clear all, close all, clc
tic
numNodosx = 400; numNodosy = 400;
maxIter = 500;
tol = 0.001;

f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % problema a resolver
fp = @(x, alpha) ((Gamma(7)*-12.84*x^(6 - alpha))/Gamma(7 - alpha)) +...
    ((Gamma(6)*-25.6*x^(5 - alpha))/Gamma(6 - alpha)) +...
    ((Gamma(5)*16.55*x^(4 - alpha))/Gamma(5 - alpha)) +...
    ((Gamma(4)*-2.21*x^(3 - alpha))/Gamma(4 - alpha)) +...
    ((Gamma(3)*26.71*x^(2 - alpha))/Gamma(3 - alpha)) +...
    ((Gamma(2)*-4.29*x^(1 - alpha))/Gamma(2 - alpha)) +...
    (-15.21/Gamma(1 - alpha))*x^-alpha;

r1 = 0.82366 + 0.24769i; r2 = 0.82366 - 0.24769i;
r3 = -2.62297; r4 = -0.584;
r5 = -0.21705 + 0.99911i; r6 = -0.21705 - 0.99911i;

n = @(x, alpha) x - (Gamma(alpha + 1)*(f(x)/fp(x, alpha)))^(1/alpha);

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3; xfin = 3; yini = 0.01; yfin = 1;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;

[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);
R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));
[rows, cols] = size(X);
noConv = 0; % inicio contador de puntos que no convergieron

for i = 1:rows
    for j = 1:cols
        x = X(i, j);
        alpha = Y(i, j);
        iter = 0;
        e = Inf;
        while iter < maxIter && e > tol
            an = x;
            x = n(x, alpha); % Newton_RL
            e = abs(an - x);
            iter = iter + 1;
        end
        d1 = abs(r1 - x); % distancia a las raices
        d2 = abs(r2 - x);
        d3 = abs(r3 - x);
        d4 = abs(r4 - x);
        d5 = abs(r5 - x);
        d6 = abs(r6 - x);
        if d1 < tol % convergio a la primera raiz
            R(i, j) = 1 - iter/maxIter;
        elseif d2 < tol % convergio a la segunda raiz
            G(i, j) = 1 - iter/maxIter;
```

```
        elseif d3 < tol % convergio a la tercera raiz
            B(i, j) = 1 - iter/maxIter;
        elseif d4 < tol % convergio a la cuarta raiz
            R(i, j) = 1 - iter/maxIter;
            G(i, j) = 1 - iter/maxIter;
        elseif d5 < tol % convergio a la quinta raiz
            R(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        elseif d6 < tol % convergio a la sexta raiz
            G(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        else
            noConv = noConv + 1;
        end
    end
end
I(:, :, 1) = R(:, :);
I(:, :, 2) = G(:, :);
I(:, :, 3) = B(:, :);

figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
xlabel('$x_0$','Interpreter','latex');
ylabel('$\alpha$','Interpreter','latex');
axis on, axis xy, axis square
densidad = ((numPuntos - noConv)/numPuntos)*100; % porciento exito
t = toc;
savefig('C3_5_newton_RL_f1_fig.fig')
save('C3_5_newton_RL_f1_mat.mat')
```

A.14. Plano de convergencia para el método NeCO (ejemplo con $f_1(x)$)

```
% Plano de convergencia: f(x) = -12.84*x^6 - 25.6*x^5 + 16.55*x^4 -
% 2.21*x^3 + 26.71*x^2 - 4.29*x - 15.21
clf, clear all, close all, clc
tic
numNodosx = 400; numNodosy = 400;
maxIter = 500;
tol = 0.001;
a = -10;

f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % problema a resolver
fp = @(x, alpha) (x - a)^(1-alpha)*(-(1926*x^5)/25 - 128*x^4 +...
    (331*x^3)/5 - (663*x^2)/100 + (2671*x)/50 - 429/100);

r1 = 0.82366 + 0.24769i; r2 = 0.82366 - 0.24769i;
r3 = -2.62297; r4 = -0.584;
r5 = -0.21705 + 0.99911i; r6 = -0.21705 - 0.99911i;

n = @(x, alpha) a + ((x - a)^alpha - alpha*f(x)/fp(x, alpha))^(1/alpha);

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3; xfin = 3; yini = 0.01; yfin = 1;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;

[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);
R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));
[rows, cols] = size(X);
noConv = 0; % inicio contador de puntos que no convergieron

for i = 1:rows
    for j = 1:cols
        x = X(i, j);
        alpha = Y(i, j);
        iter = 0;
        e = Inf;
        while iter < maxIter && e > tol
            an = x;
            x = n(x, alpha); % Newton_conf
            e = abs(an - x);
            iter = iter + 1;
        end
        d1 = abs(r1 - x); % distancia a las raices
        d2 = abs(r2 - x);
        d3 = abs(r3 - x);
        d4 = abs(r4 - x);
        d5 = abs(r5 - x);
        d6 = abs(r6 - x);
        if d1 < tol % convergio a la primera raiz
            R(i, j) = 1 - iter/maxIter;
        elseif d2 < tol % convergio a la segunda raiz
            G(i, j) = 1 - iter/maxIter;
        elseif d3 < tol % convergio a la tercera raiz
            B(i, j) = 1 - iter/maxIter;
        elseif d4 < tol % convergio a la cuarta raiz
            R(i, j) = 1 - iter/maxIter;
```

```
            G(i, j) = 1 - iter/maxIter;
        elseif d5 < tol % convergio a la quinta raiz
            R(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        elseif d6 < tol % convergio a la sexta raiz
            G(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        else
            noConv = noConv + 1;
        end
    end
end
I(:, :, 1) = R(:, :);
I(:, :, 2) = G(:, :);
I(:, :, 3) = B(:, :);

figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
xlabel('$x_0$','Interpreter','latex');
ylabel('$\alpha$','Interpreter','latex');
axis on, axis xy, axis square
densidad = ((numPuntos - noConv)/numPuntos)*100; % porciento exito
t = toc;
savefig('C3_9_newton_conf_f1_fig.fig')
save('C3_9_newton_conf_f1_mat.mat')
```

A.15. Plano de convergencia para el método NeA3 (ejemplo con $f_1(x)$)

```
% Plano de convergencia: f(x) = -12.84*x^6 - 25.6*x^5 + 16.55*x^4 -
% 2.21*x^3 + 26.71*x^2 - 4.29*x - 15.21
clf, clear all, close all, clc
tic
numNodosx = 400; numNodosy = 400;
maxIter = 500;
tol = 0.001;

f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % problema a resolver
fp = @(x) -(1926*x^5)/25 - 128*x^4 +...
    (331*x^3)/5 - (663*x^2)/100 + (2671*x)/50 - 429/100;
fpp = @(x) -(1926*x^4)/5 - 512*x^3 + (993*x^2)/5 - (663*x)/50 + 2671/50;
a = @(x, alpha) x + (1 - alpha)*fp(x)/fpp(x);
df = @(x, alpha) (x - a(x, alpha))^(1-alpha)*(-(1926*x^5)/25 - 128*x^4 +...
    (331*x^3)/5 - (663*x^2)/100 + (2671*x)/50 - 429/100);

r1 = 0.82366 + 0.24769i; r2 = 0.82366 - 0.24769i;
r3 = -2.62297; r4 = -0.584;
r5 = -0.21705 + 0.99911i; r6 = -0.21705 - 0.99911i;

n = @(x, alpha) a(x, alpha) + ((x - a(x, alpha))^alpha -...
    alpha*f(x)/df(x, alpha))^(1/alpha);

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3; xfin = 3; yini = 0.01; yfin = 1;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;

[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);
R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));
[rows, cols] = size(X);
noConv = 0; % inicio contador de puntos que no convergieron

for i = 1:rows
    for j = 1:cols
        x = X(i, j);
        alpha = Y(i, j);
        iter = 0;
        e = Inf;
        while iter < maxIter && e > tol
            an = x;
            x = n(x, alpha); % Newton_conf_a3
            e = abs(an - x);
            iter = iter + 1;
        end
        d1 = abs(r1 - x); % distancia a las raices
        d2 = abs(r2 - x);
        d3 = abs(r3 - x);
        d4 = abs(r4 - x);
        d5 = abs(r5 - x);
        d6 = abs(r6 - x);
        if d1 < tol % convergio a la primera raiz
            R(i, j) = 1 - iter/maxIter;
        elseif d2 < tol % convergio a la segunda raiz
            G(i, j) = 1 - iter/maxIter;
```

```
        elseif d3 < tol % convergio a la tercera raiz
            B(i, j) = 1 - iter/maxIter;
        elseif d4 < tol % convergio a la cuarta raiz
            R(i, j) = 1 - iter/maxIter;
            G(i, j) = 1 - iter/maxIter;
        elseif d5 < tol % convergio a la quinta raiz
            R(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        elseif d6 < tol % convergio a la sexta raiz
            G(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        else
            noConv = noConv + 1;
        end
    end
end
I(:, :, 1) = R(:, :);
I(:, :, 2) = G(:, :);
I(:, :, 3) = B(:, :);

figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
xlabel('$x_0$','Interpreter','latex');
ylabel('$\alpha$','Interpreter','latex');
axis on, axis xy, axis square
densidad = ((numPuntos - noConv)/numPuntos)*100; % porciento exito
t = toc;
savefig('C3_13_newton_conf_a3_f1_fig.fig')
save('C3_13_newton_conf_a3_f1_mat.mat')
```

A.16. Plano dinámico para el método NeCO (ejemplo con $p_1(z)$)

```
% Plano dinamico: f(z) = z^2 - 1
tic
numNodosx = 400; numNodosy = 400;
maxIter = 100;
tol = 0.001;
a = -10;

f = @(x) x^2 - 1;
df = @(x) (x - a)^(1-alpha)*(2*x);
r1 = 1; r2 = -1;

N = @(x) a + ((x - a)^alpha - alpha*f(x)/df(x))^(1/alpha);

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3; xfin = 3; yini = -3; yfin = 3;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;
[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);

R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));

[rows, cols] = size(X);

    for m = 1:rows
        for n = 1:cols
            x = complex(X(m, n), Y(m, n));
            iter = 0;
            e = Inf;
            while iter < maxIter && e > tol
                an = x;
                x = N(x);
                e = abs(an - x);
                iter = iter + 1;
            end
            d1 = abs(r1 - x);
            d2 = abs(r2 - x);
            if d2 < tol
                R(m, n) = (1 - iter/maxIter);
            elseif d1 < tol
                B(m, n) = (1 - iter/maxIter);
            else
                R(m, n) = 0;
                G(m, n) = 0;
                B(m, n) = 0;
            end
        end
    end
      I(:, :, 1) = R(:, :);
      I(:, :, 2) = G(:, :);
      I(:, :, 3) = B(:, :);
      figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
      xlabel('Re'); ylabel('Im');
      hold on, axis on, axis xy, grid
      plot(real(r1), imag(r1), 'w+', 'MarkerSize', 5);
      plot(real(r2), imag(r2), 'w+', 'MarkerSize', 5);
      t = toc;
```

A.17. Plano dinámico para el método NeL3 (ejemplo con $p_1(z)$)

```
% Plano dinamico: f(z) = z^2 - 1
clf, clear all, close all, clc
tic
numNodosx = 400; numNodosy = 400;
maxIter = 100;
tol = 0.001;
a = -10;

f = @(x) x^2 - 1;
fp = @(x) 2*x;
fpp = @(x) 2;
alpha = @(x) 1 + (x - a)*fpp(x)/fp(x);
df = @(x) (x - a)^(1 - alpha(x))*(2*x);
r1 = 1; r2 = -1;

N = @(x) a + ((x - a)^alpha(x) - alpha(x)*f(x)/df(x))^(1/alpha(x));

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3; xfin = 3; yini = -3; yfin = 3;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;
[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);

R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));

[rows, cols] = size(X);

    for m = 1:rows
        for n = 1:cols
            x = complex(X(m, n), Y(m, n));
            iter = 0;
            e = Inf;
            while iter < maxIter && e > tol
                an = x;
                x = N(x);
                e = abs(an - x);
                iter = iter + 1;
            end
            d1 = abs(r1 - x);
            d2 = abs(r2 - x);
            if d2 < tol
                R(m, n) = (1 - iter/maxIter);
            elseif d1 < tol
                B(m, n) = (1 - iter/maxIter);
            else
                R(m, n) = 0;
                G(m, n) = 0;
                B(m, n) = 0;
            end
        end
    end
      I(:, :, 1) = R(:, :);
      I(:, :, 2) = G(:, :);
      I(:, :, 3) = B(:, :);
      figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
      xlabel('Re'); ylabel('Im');
      hold on, axis on, axis xy, grid
      plot(real(r1), imag(r1), 'w+', 'MarkerSize', 5);
```

```
plot(real(r2), imag(r2), 'w+', 'MarkerSize', 5);
t = toc;
savefig('C3_17_newton_conf_alpha3_p1_fig.fig')
save('C3_17_newton_conf_alpha3_p1_mat.mat')
```

A.18. Plano dinámico para el método NeLA4 (ejemplo con $p_1(z)$)

```
% Plano dinamico: f(z) = z^2 - 1
clf, clear all, close all, clc
tic
numNodosx = 400; numNodosy = 400;
maxIter = 100;
tol = 0.001;

f = @(x) x^2 - 1;
fp = @(x) 2*x;
fpp = @(x) 2;
fppp = @(x) 0;
c2 = @(x) (1/2)*fpp(x)/fp(x);
c3 = @(x) (1/6)*fppp(x)/fp(x);
alpha = @(x) (3*c3(x) - 4*c2(x)^2)/(3*c3(x) - 2*c2(x)^2);
a = @(x) x + c2(x)/(3*c3(x) - 2*c2(x)^2);
df = @(x) (x - a(x))^(1 - alpha(x))*(2*x);
r1 = 1; r2 = -1;

N = @(x) a(x) + ((x - a(x))^alpha(x) - alpha(x)*f(x)/df(x))^(1/alpha(x));

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3; xfin = 3; yini = -3; yfin = 3;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;
[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);

R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));

[rows, cols] = size(X);

    for m = 1:rows
        for n = 1:cols
            x = complex(X(m, n), Y(m, n));
            iter = 0;
            e = Inf;
            while iter < maxIter && e > tol
                an = x;
                x = N(x);
                e = abs(an - x);
                iter = iter + 1;
            end
            d1 = abs(r1 - x);
            d2 = abs(r2 - x);
            if d2 < tol
                R(m, n) = (1 - iter/maxIter);
            elseif d1 < tol
                B(m, n) = (1 - iter/maxIter);
            else
                R(m, n) = 0;
                G(m, n) = 0;
                B(m, n) = 0;
            end
        end
    end
      I(:, :, 1) = R(:, :);
      I(:, :, 2) = G(:, :);
      I(:, :, 3) = B(:, :);
      figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
```

```
xlabel('Re'); ylabel('Im');
hold on, axis on, axis xy, grid
plot(real(r1), imag(r1), 'w+', 'MarkerSize', 5);
plot(real(r2), imag(r2), 'w+', 'MarkerSize', 5);
t = toc;
savefig('C3_21_newton_conf_aalpha4_p1_fig.fig')
save('C3_21_newton_conf_aalpha4_p1_mat.mat')
```

A.19. Método NvCO (sistemas 2×2)

```
function [iter, z, err1, err2, ACOC] = pruebas_NvCO_a(z, alpha, a)
F = @(x) [x(1)^2 - 2*x(1) - x(2) + 0.5; x(1)^2 + 4*x(2)^2 - 4]; % funcion 1
J = @(x) [(x(1) - a(1))^(1 - alpha)*(2*x(1) - 2), (x(2) - a(2))^(1 - alpha)*(-1);...
    (x(1) - a(1))^(1 - alpha)*(2*x(1)), (x(2) - a(2))^(1 - alpha)*(8*x(2))]; % jacobiana 1
% F = @(x) [x(1)^2 + x(2)^2 - 1; x(1)^2 - x(2)^2 - 1/2]; % funcion 2
% J = @(x) [(x(1) - a(1))^(1 - alpha)*(2*x(1)), (x(2) - a(2))^(1 - alpha)*(2*x(2));...
%     (x(1) - a(1))^(1 - alpha)*(2*x(1)), (x(2) - a(2))^(1 - alpha)*(-2*x(2))]; % jacobiana 2
% F = @(x) [x(1)^2 - x(1) - x(2)^2 - 1; -sin(x(1)) + x(2)]; % funcion 3
% J = @(x) [(x(1) - a(1))^(1 - alpha)*(2*x(1) - 1), (x(2) - a(2))^(1 - alpha)*(-2*x(2));...
%     (x(1) - a(1))^(1 - alpha)*(-cos(x(1))), (x(2) - a(2))^(1 - alpha)*(1)]; % jacobiana 3
% F = @(x) [x(1)^2 + x(2)^2 - 4; exp(x(1)) + x(2) - 1]; % funcion 4
% J = @(x) [(x(1) - a(1))^(1 - alpha)*(2*x(1)), (x(2) - a(2))^(1 - alpha)*(2*x(2));...
%     (x(1) - a(1))^(1 - alpha)*(exp(x(1))), (x(2) - a(2))^(1 - alpha)*(1)]; % jacobiana 4
err1 = Inf;
err2 = Inf;
z1 = z';
ACOC = Inf*ones(1);
iter = 0;
while iter < 500 && err1 > 1e-8 && err2(end) > 1e-8
    zant = z;
    z = a + ((z - a).^alpha - alpha*(J(z)\F(z))).^(1/alpha);
    err1 = norm(F(z));
    err2 = [err2 norm(z - zant)];
    z1(iter + 1,:) = z';
    if iter >= 3
        ACOC0 = (log(norm((z1(end,:) - z1(end - 1,:)))/norm((z1(end - 1,:)...
            - z1(end - 2,:)))))/(log(norm((z1(end - 1,:) - z1(end - 2,:)))...
            /norm((z1(end - 2,:) - z1(end - 3,:)))));
    elseif iter < 3
        ACOC0 = Inf;
    end
    ACOC(iter + 1) = ACOC0;
    iter = iter + 1;
end
t = 1:iter;
err2(1) = [];
semilogy(t, err2), grid
xlabel('Iteraciones')
ylabel('$\|x^{(k+1)}-x^{(k)}\|$','Interpreter','latex')
end
% [iter, z, err1, err2, ACOC] = pruebas_NvCO_a([-2; -1.5], 0.5, -10*ones(2, 1))
```

A.20. Sistemas de ecuaciones, y Jacobianas conformables 15×15 y 10×10

```
function [F,dF] = sistemas_dT(x, a, alpha)

%global num
num=1; % Seleccionar caso 1 o 2
x=x(:);
switch num

    case 1

            n=length(x);
        for ii=1:n-1
            y(ii)= x(ii)*x(ii+1)-1;
        end
        y(n)=x(n)*x(1)-1;
        y=y(:);
        % conformable
        M = ones(n);
        for k=1:n
            M(:,k)=(x(k)-a(k))^(1-alpha);
        end
        %size(x)
        a=x(2:n); a=a(:);
        a=[a; x(1)]; %size(a),
        b=x(1:n-1); %size(b)
        dy=diag(a)+diag(b,1);
        dy(n,1)=x(n);
        F=y;
        dF=M.*dy;

    case 2

        n=length(x);
        for i=1:n
            y(i)=x(i)-1.5*sin(sum(x)-x(i));
        end
        y=y(:);
        % conformable
        M = ones(n);
        for k=1:n
            M(:,k)=(x(k)-a(k))^(1-alpha);
        end
        %size(x)
        dy=zeros(n,n);
        for i=1:n
        dy(:,i)=-1.5*cos(sum(x)-x(i));
        dy(i,i)=1;
        end
        F=y;
        dF=M.*dy;

end
```

A.21. Método NvCO (sistemas 15×15 y 10×10)

```
function [iter, z, err1, err2, ACOC] = pruebas_NvCO_b(z, alpha, a)
err1 = Inf;
err2 = Inf;
z1 = z';
ACOC = Inf*ones(1);
iter = 0;
while iter < 500 && err1 > 1e-8 && err2(end) > 1e-8
    zant = z;
    [F,J] = sistemas_dT(z, a, alpha);
    z = a + ((z - a).^alpha - alpha*(J\F)).^(1/alpha);
    err1 = norm(F);
    err2 = [err2 norm(z - zant)];
    z1(iter + 1,:) = z';
    if iter >= 3
        ACOC0 = (log(norm((z1(end,:) - z1(end - 1,:)))/norm((z1(end - 1,:)...
            - z1(end - 2,:)))))/(log(norm((z1(end - 1,:) - z1(end - 2,:)))...
            /norm((z1(end - 2,:) - z1(end - 3,:)))));
    elseif iter < 3
        ACOC0 = Inf;
    end
    ACOC(iter + 1) = ACOC0;
    iter = iter + 1;
end
t = 1:iter;
err2(1) = [];
semilogy(t, err2), grid
xlabel('Iteraciones')
ylabel('$\|x^{(k+1)}-x^{(k)}\|$','Interpreter','latex')
% z = double(z);
% err1 = double(err1);
% err2 = double(err2);
end
% [iter, z, err1, err2, ACOC] = pruebas_NvCO_b(2*ones(5, 1), 0.5, -10*ones(5, 1))
```

A.22. Curvas de error para el método NvCO

```
% Tabla 1
clear all
clc
for k = [0.6 0.3 0.2] % 0.1:0.1:0.9
    pruebas_NvCO_a([-2; -1.5], k, -10*ones(2, 1));
    hold on
end
legend('$\alpha=0.6$','$\alpha=0.3$','$\alpha=0.2$','Interpreter','latex')
% legend('$\alpha=0.1$','$\alpha=0.2$','$\alpha=0.3$','$\alpha=0.4$',...
%     '$\alpha=0.5$','$\alpha=0.6$','$\alpha=0.7$','$\alpha=0.8$',...
%     '$\alpha=0.9$','Interpreter','latex')
hold off
%%
% Tabla 2
clear all
clc
for k = [1 0.6 0.2] % 0.1:0.1:1
    pruebas_NvCO_a([-2; 1.5], k, -10*ones(2, 1));
    hold on
end
legend('$\alpha=1$','$\alpha=0.6$','$\alpha=0.2$','Interpreter','latex')
% legend('$\alpha=0.1$','$\alpha=0.2$','$\alpha=0.3$','$\alpha=0.4$',...
%     '$\alpha=0.5$','$\alpha=0.6$','$\alpha=0.7$','$\alpha=0.8$',...
%     '$\alpha=0.9$','$\alpha=1$','Interpreter','latex')
hold off
%%
% Tabla 3
clear all
clc
for k = [1 0.5 0.1] % 0.1:0.1:1
    pruebas_NvCO_a([2; -2.5], k, -10*ones(2, 1));
    hold on
end
legend('$\alpha=1$','$\alpha=0.5$','$\alpha=0.1$','Interpreter','latex')
% legend('$\alpha=0.1$','$\alpha=0.2$','$\alpha=0.3$','$\alpha=0.4$',...
%     '$\alpha=0.5$','$\alpha=0.6$','$\alpha=0.7$','$\alpha=0.8$',...
%     '$\alpha=0.9$','$\alpha=1$','Interpreter','latex')
hold off
%%
% Tabla 4
clear all
clc
for k = [1 0.7 0.4] % 0.1:0.1:1
    pruebas_NvCO_a([2; 2.5], k, -10*ones(2, 1));
    hold on
end
legend('$\alpha=1$','$\alpha=0.7$','$\alpha=0.4$','Interpreter','latex')
% legend('$\alpha=0.1$','$\alpha=0.2$','$\alpha=0.3$','$\alpha=0.4$',...
%     '$\alpha=0.5$','$\alpha=0.6$','$\alpha=0.7$','$\alpha=0.8$',...
%     '$\alpha=0.9$','$\alpha=1$','Interpreter','latex')
hold off
%%
% Tabla 5
clear all
clc
for k = [1 0.4 0.2] % 0.1:0.1:1
    pruebas_NvCO_a([2.5; -0.5], k, -10*ones(2, 1));
    hold on
```

```
end
legend('$\alpha=1$','$\alpha=0.4$','$\alpha=0.2$','Interpreter','latex')
% legend('$\alpha=0.1$','$\alpha=0.2$','$\alpha=0.3$','$\alpha=0.4$',...
%       '$\alpha=0.5$','$\alpha=0.6$','$\alpha=0.7$','$\alpha=0.8$',...
%       '$\alpha=0.9$','$\alpha=1$','Interpreter','latex')
hold off
%%
% Tabla 6
clear all
clc
for k = [1 0.6 0.2] % 0.1:0.1:1
    pruebas_NvCO_a([2.5; 0.5], k, -10*ones(2, 1));
    hold on
end
legend('$\alpha=1$','$\alpha=0.6$','$\alpha=0.2$','Interpreter','latex')
% legend('$\alpha=0.1$','$\alpha=0.2$','$\alpha=0.3$','$\alpha=0.4$',...
%       '$\alpha=0.5$','$\alpha=0.6$','$\alpha=0.7$','$\alpha=0.8$',...
%       '$\alpha=0.9$','$\alpha=1$','Interpreter','latex')
hold off
%%
% Tabla 7
clear all
clc
for k = [1 0.5 0.3] % 0.1:0.1:1
    pruebas_NvCO_a([-2.5; -3.5], k, -10*ones(2, 1));
    hold on
end
legend('$\alpha=1$','$\alpha=0.5$','$\alpha=0.3$','Interpreter','latex')
% legend('$\alpha=0.1$','$\alpha=0.2$','$\alpha=0.3$','$\alpha=0.4$',...
%       '$\alpha=0.5$','$\alpha=0.6$','$\alpha=0.7$','$\alpha=0.8$',...
%       '$\alpha=0.9$','$\alpha=1$','Interpreter','latex')
hold off
%%
% Tabla 8
clear all
clc
for k = [1 0.9 0.4] % 0.1:0.1:1
    pruebas_NvCO_a([-2.5; 3.5], k, -10*ones(2, 1));
    hold on
end
legend('$\alpha=1$','$\alpha=0.9$','$\alpha=0.4$','Interpreter','latex')
% legend('$\alpha=0.1$','$\alpha=0.2$','$\alpha=0.3$','$\alpha=0.4$',...
%       '$\alpha=0.5$','$\alpha=0.6$','$\alpha=0.7$','$\alpha=0.8$',...
%       '$\alpha=0.9$','$\alpha=1$','Interpreter','latex')
hold off
%%
% Tabla 9
clear all
clc
for k = [1 0.5 0.1] % 0.1:0.1:1
    pruebas_NvCO_b(-1.5*ones(15, 1), k, -10*ones(15, 1));
    hold on
end
legend('$\alpha=1$','$\alpha=0.5$','$\alpha=0.1$','Interpreter','latex')
% legend('$\alpha=0.1$','$\alpha=0.2$','$\alpha=0.3$','$\alpha=0.4$',...
%       '$\alpha=0.5$','$\alpha=0.6$','$\alpha=0.7$','$\alpha=0.8$',...
%       '$\alpha=0.9$','$\alpha=1$','Interpreter','latex')
hold off
%%
% Tabla 10
clear all
```

```
clc
for k = [1 0.8 0.5] % 0.1:0.1:1
    pruebas_NvCO_b(2.5*ones(15, 1), k, -10*ones(15, 1));
    hold on
end
legend('$\alpha=1$','$\alpha=0.8$','$\alpha=0.5$','Interpreter','latex')
% legend('$\alpha=0.1$','$\alpha=0.2$','$\alpha=0.3$','$\alpha=0.4$',...
%     '$\alpha=0.5$','$\alpha=0.6$','$\alpha=0.7$','$\alpha=0.8$',...
%     '$\alpha=0.9$','$\alpha=1$','Interpreter','latex')
hold off
%%
% Tabla 11
clear all
clc
for k = [1 0.5 0.3] % [0.1:0.1:0.8 1]
    pruebas_NvCO_b(2*ones(10, 1), k, -10*ones(10, 1));
    hold on
end
legend('$\alpha=1$','$\alpha=0.5$','$\alpha=0.3$','Interpreter','latex')
% legend('$\alpha=0.1$','$\alpha=0.2$','$\alpha=0.3$','$\alpha=0.4$',...
%     '$\alpha=0.5$','$\alpha=0.6$','$\alpha=0.7$','$\alpha=0.8$',...
%     '$\alpha=1$','Interpreter','latex')
hold off
%%
% Tabla 12
clear all
clc
for k = [0.6 0.3 0.1] % [0.1 0.3 0.4 0.5 0.6 0.8]
    pruebas_NvCO_b(3*ones(10, 1), k, -10*ones(10, 1));
    hold on
end
legend('$\alpha=0.6$','$\alpha=0.3$','$\alpha=0.1$','Interpreter','latex')
% legend('$\alpha=0.1$','$\alpha=0.3$','$\alpha=0.4$','$\alpha=0.5$',...
%     '$\alpha=0.6$','$\alpha=0.8$','Interpreter','latex')
hold off
```

A.23. Plano de convergencia para el método NvCO (ejemplo con $F_1(x,y)$)

```
% Plano de convergencia: F(x,y) = [x^2 - 2*x - y + 0.5; x^2 + 4*y^2 - 4]
tic
numNodosx = 400; numNodosy = 400;
maxIter = 500;
tol = 1e-3;
a = -10*[1; 1];

F = @(z) [z(1)^2 - 2*z(1) - z(2) + 0.5; z(1)^2 + 4*z(2)^2 - 4];
J = @(z, alpha) [(z(1) - a(1))^(1 - alpha)*(2*z(1) - 2), (z(2) - a(2))^...
    (1 - alpha)*(-1);(z(1) - a(1))^(1 - alpha)*(2*z(1)), (z(2) - a(2))^...
    (1 - alpha)*(8*z(2))];

r1 = [1.9007; 0.3112];
r2 = [-0.2222; 0.9938];
r3 = [1.1608 - 0.6545i; -0.9025 - 0.2104i];

n = @(z, alpha) a + ((z - a).^alpha - alpha*(J(z, alpha)\F(z))).^(1/alpha);

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3.5; xfin = 3.5; yini = 0.01; yfin = 1;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;

[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);
R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));
[rows, cols] = size(X);
noConv = 0; % inicio contador de puntos que no convergieron

for i = 1:rows
    for j = 1:cols
        x = X(i, j);
        alpha = Y(i, j);
        z = [x; y];
        iter = 0;
        e = Inf;
        while iter < maxIter && e > tol
            zant = z;
            z = n(z, alpha); % Newton_sistemas
            e = norm(zant - z);
            iter = iter + 1;
        end
        dist1 = norm(r1 - z); % distancia a la primera raiz
        dist2 = norm(r2 - z); % distancia a la segunda raiz
        dist3 = norm(r3 - z); % distancia a la tercera raiz
        if dist1 < tol % convergio a la primera raiz
            R(i, j) = 1 - iter/maxIter;
        elseif dist2 < tol % convergio a la segunda raiz
            G(i, j) = 1 - iter/maxIter;
        elseif dist3 < tol % convergio a la tercera raiz
            B(i, j) = 1 - iter/maxIter;
        else
            noConv = noConv + 1;
        end
    end
end
I(:, :, 1) = R(:, :);
```

```
I(:, :, 2) = G(:, :);
I(:, :, 3) = B(:, :);

figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
xlabel('$x_0$','Interpreter','latex');
ylabel('$\alpha$','Interpreter','latex');
axis on, axis xy, axis square
densidad = ((numPuntos - noConv)/numPuntos)*100; % porciento exito
t = toc;
```

A.24. Método TeCA

```
function [x, err1, err2, iter, ACOC] = pruebas_TeCA(x0, alpha)
f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % Funcion 1
df = @(x) ((Gamma(7)*-12.84*x^(6 - alpha))/Gamma(7 - alpha)) +...
    ((Gamma(6)*-25.6*x^(5 - alpha))/Gamma(6 - alpha)) +...
    ((Gamma(5)*16.55*x^(4 - alpha))/Gamma(5 - alpha)) +...
    ((Gamma(4)*-2.21*x^(3 - alpha))/Gamma(4 - alpha)) +...
    ((Gamma(3)*26.71*x^(2 - alpha))/Gamma(3 - alpha)) +...
    ((Gamma(2)*-4.29*x^(1 - alpha))/Gamma(2 - alpha)); % Derivada de Caputo 1
% f = @(x) 1i*x^1.8 - x^0.9 - 16; % Funcion 2
% df = @(x) ((Gamma(2.8)*1i*x^(1.8 - alpha))/Gamma(2.8 - alpha)) +...
%     ((Gamma(1.9)*-1*x^(0.9 - alpha))/Gamma(1.9 - alpha)); % Derivada de Caputo 2
% f = @(x) exp(x) - 1; % Funcion 3
% df = @(x) x^(1 - alpha)*mlf(1, 2 - alpha, x, 9); % Derivada de Caputo 3
% f = @(x) sin(10*x) - 0.5*x + 0.2; % Funcion 4
% df = @(x) 5*x^(1 - alpha)*(mlf(1, 2 - alpha, 10i*x, 9) +...
%     mlf(1, 2 - alpha, -10i*x, 9)) +...
%     ((Gamma(2)*-0.5*x^(1 - alpha))/Gamma(2 - alpha)); % Derivada de Caputo 4
err1 = Inf;
err2 = Inf;
x = x0*ones(1);
ACOC = Inf*ones(1);
iter = 0;
while iter < 500 && err1 > 1e-8 && err2 > 1e-8
    yk = x(end) - (Gamma(alpha + 1)*(f(x(end))/df(x(end))))^(1/alpha);
    x(iter + 2) = yk - (Gamma(alpha + 1)*(f(yk)/df(x(end))))^(1/alpha);
    err1 = abs(f(x(end)));
    err2 = abs(x(end) - x(end - 1));
    if iter >= 2
        ACOC(iter + 2) = (log(abs((x(end) - x(end - 1)))/abs((x(end - 1)...
            - x(end - 2)))))/(log(abs((x(end - 1) - x(end - 2)))...
            /abs((x(end - 2) - x(end - 3)))));
    elseif iter < 2
        ACOC(iter + 2) = Inf;
    end
    iter = iter + 1;
end
x = x(end);
ACOC(1) = [];
t = 1:iter;
plot(t, ACOC), grid
xlabel('Iteraciones')
ylabel('Orden_de_convergencia')
end
% Ejemplo de uso:
% [x, err1, err2, iter, ACOC] = pruebas_TeCA(-2.2, 0.9)
```

A.25. Método TeRL

```
function [x, err1, err2, iter, ACOC] = pruebas_TeRL(x0, alpha)
f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % Funcion 1
df = @(x) ((Gamma(7)*-12.84*x^(6 - alpha))/Gamma(7 - alpha)) +...
    ((Gamma(6)*-25.6*x^(5 - alpha))/Gamma(6 - alpha)) +...
    ((Gamma(5)*16.55*x^(4 - alpha))/Gamma(5 - alpha)) +...
    ((Gamma(4)*-2.21*x^(3 - alpha))/Gamma(4 - alpha)) +...
    ((Gamma(3)*26.71*x^(2 - alpha))/Gamma(3 - alpha)) +...
    ((Gamma(2)*-4.29*x^(1 - alpha))/Gamma(2 - alpha)) +...
    (-15.21/Gamma(1 - alpha))*x^-alpha; % Derivada de Riemann-Liouville 1
% f = @(x) 1i*x^1.8 - x^0.9 - 16; % Funcion 2
% df = @(x) ((Gamma(2.8)*1i*x^(1.8 - alpha))/Gamma(2.8 - alpha)) +...
%     ((Gamma(1.9)*-1*x^(0.9 - alpha))/Gamma(1.9 - alpha)) +...
%     (-16/Gamma(1 - alpha))*x^-alpha; % Derivada de Riemann-Liouville 2
% f = @(x) exp(x) - 1; % Funcion 3
% df = @(x) x^(1 - alpha)*mlf(1, 2 - alpha, x, 9) +...
%     (-1/Gamma(1 - alpha))*x^-alpha; % Derivada de Riemann-Liouville 3
% f = @(x) sin(10*x) - 0.5*x + 0.2; % Funcion 4
% df = @(x) 5*x^(1 - alpha)*(mlf(1, 2 - alpha, 10i*x, 9) +...
%     mlf(1, 2 - alpha, -10i*x, 9)) +...
%     ((Gamma(2)*-0.5*x^(1 - alpha))/Gamma(2 - alpha)) +...
%     (0.2/Gamma(1 - alpha))*x^-alpha; % Derivada de Riemann-Liouville 4
err1 = Inf;
err2 = Inf;
x = x0*ones(1);
ACOC = Inf*ones(1);
iter = 0;
while iter < 500 && err1 > 1e-8 && err2 > 1e-8
    yk = x(end) - (Gamma(alpha + 1)*(f(x(end))/df(x(end))))^(1/alpha);
    x(iter + 2) = yk - (Gamma(alpha + 1)*(f(yk)/df(x(end))))^(1/alpha);
    err1 = abs(f(x(end)));
    err2 = abs(x(end) - x(end - 1));
    if iter >= 2
        ACOC(iter + 2) = (log(abs((x(end) - x(end - 1)))/abs((x(end - 1)...
            - x(end - 2)))))/(log(abs((x(end - 1) - x(end - 2)))...
            /abs((x(end - 2) - x(end - 3)))));
    elseif iter < 2
        ACOC(iter + 2) = Inf;
    end
    iter = iter + 1;
end
x = x(end);
ACOC(1) = [];
t = 1:iter;
plot(t, ACOC), grid
xlabel('Iteraciones')
ylabel('Orden de convergencia')
end
% Ejemplo de uso:
% [x, err1, err2, iter, ACOC] = pruebas_TeRL(-2.2, 0.9)
```

A.26. Método TeCO

```
function [x, err1, err2, iter, ACOC] = pruebas_TeCO(x0, alpha, a)
f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % Funcion 1
df = @(x) (x - a)^(1 - alpha)*(-(1926*x^5)/25 - 128*x^4 + (331*x^3)/5 -...
    (663*x^2)/100 + (2671*x)/50 - 429/100); % Derivada conformable 1
% f = @(x) 1i*x^1.8 - x^0.9 - 16; % Funcion 2
% df = @(x) (x - a)^(1 - alpha)*(1.8*1i*x^0.8 - 0.9*x^-0.1); % Derivada
% conformable 2
% f = @(x) exp(x) - 1; % Funcion 3
% df = @(x) (x - a)^(1-alpha)*exp(x); % Derivada conformable 3
% f = @(x) sin(10*x) - 0.5*x + 0.2; % Funcion 4
% df = @(x) (x - a)^(1-alpha)*(10*cos(10*x)-0.5); % Derivada conformable 4
err1 = Inf;
err2 = Inf;
x = x0*ones(1);
ACOC = Inf*ones(1);
iter = 0;
while iter < 500 && err1 > 1e-8 && err2 > 1e-8
    yk = a + ((x(end) - a)^alpha - alpha*f(x(end))/df(x(end)))^(1/alpha);
    x(iter + 2) = a + ((yk - a)^alpha - alpha*f(yk)/df(x(end)))^(1/alpha);
    err1 = abs(f(x(end)));
    err2 = abs(x(end) - x(end - 1));
    if iter >= 2
        ACOC(iter + 2) = (log(abs((x(end) - x(end - 1)))/abs((x(end - 1)...
            - x(end - 2)))))/(log(abs((x(end - 1) - x(end - 2)))...
            /abs((x(end - 2) - x(end - 3)))));
    elseif iter < 2
        ACOC(iter + 2) = Inf;
    end
    iter = iter + 1;
end
x = x(end);
ACOC(1) = [];
t = 1:iter;
plot(t, ACOC), grid
xlabel('Iteraciones')
ylabel('Orden_de_convergencia')
end
% Ejemplo de uso:
% [x, err1, err2, iter, ACOC] = pruebas_TeCO(-2.2, 0.5, -10)
```

A.27. Método CKeCO

```
function [x, err1, err2, iter, ACOC] = pruebas_CKeCO(x0, alpha, a)
f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % Funcion 1
df = @(x) (x - a)^(1 - alpha)*(-(1926*x^5)/25 - 128*x^4 + (331*x^3)/5 -...
    (663*x^2)/100 + (2671*x)/50 - 429/100); % Derivada conformable 1
% f = @(x) 1i*x^1.8 - x^0.9 - 16; % Funcion 2
% df = @(x) (x - a)^(1 - alpha)*(1.8*1i*x^0.8 - 0.9*x^-0.1); % Derivada
% conformable 2
% f = @(x) exp(x) - 1; % Funcion 3
% df = @(x) (x - a)^(1-alpha)*exp(x); % Derivada conformable 3
% f = @(x) sin(10*x) - 0.5*x + 0.2; % Funcion 4
% df = @(x) (x - a)^(1-alpha)*(10*cos(10*x)-0.5); % Derivada conformable 4
err1 = Inf;
err2 = Inf;
x = x0*ones(1);
ACOC = Inf*ones(1);
iter = 0;
while iter < 500 && err1 > 1e-8 && err2 > 1e-8
    yk = a + ((x(end) - a)^alpha - alpha*f(x(end))/df(x(end)))^(1/alpha);
    x(iter + 2) = a + ((x(end) - a)^alpha - alpha/2*(3 - df(yk)/df(x(end)))*...
        f(x(end))/df(x(end)))^(1/alpha);
    err1 = abs(f(x(end)));
    err2 = abs(x(end) - x(end - 1));
    if iter >= 2
        ACOC(iter + 2) = (log(abs((x(end) - x(end - 1)))/abs((x(end - 1)...
            - x(end - 2)))))/(log(abs((x(end - 1) - x(end - 2)))...
            /abs((x(end - 2) - x(end - 3)))));
    elseif iter < 2
        ACOC(iter + 2) = Inf;
    end
    iter = iter + 1;
end
x = x(end);
ACOC(1) = [];
t = 1:iter;
plot(t, ACOC), grid
xlabel('Iteraciones')
ylabel('Orden de convergencia')
end
% Ejemplo de uso:
% [x, err1, err2, iter, ACOC] = pruebas_CKeCO(-2.2, 0.5, -10)
```

A.28. Método OeCO

```
function [x, err1, err2, iter, ACOC] = pruebas_OeCO(x0, alpha, a)
f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % Funcion 1
df = @(x) (x - a)^(1 - alpha)*(-(1926*x^5)/25 - 128*x^4 + (331*x^3)/5 -...
    (663*x^2)/100 + (2671*x)/50 - 429/100); % Derivada conformable 1
% f = @(x) 1i*x^1.8 - x^0.9 - 16; % Funcion 2
% df = @(x) (x - a)^(1 - alpha)*(1.8*1i*x^0.8 - 0.9*x^-0.1); % Derivada
% conformable 2
% f = @(x) exp(x) - 1; % Funcion 3
% df = @(x) (x - a)^(1-alpha)*exp(x); % Derivada conformable 3
% f = @(x) sin(10*x) - 0.5*x + 0.2; % Funcion 4
% df = @(x) (x - a)^(1-alpha)*(10*cos(10*x)-0.5); % Derivada conformable 4
err1 = Inf;
err2 = Inf;
x = x0*ones(1);
ACOC = Inf*ones(1);
iter = 0;
while iter < 500 && err1 > 1e-8 && err2 > 1e-8
    yk = a + ((x(end) - a)^alpha - alpha*f(x(end))/df(x(end)))^(1/alpha);
    x(iter + 2) = a + ((yk - a)^alpha - alpha*(f(x(end))/(f(x(end)) -...
        2*f(yk)))*f(yk)/df(x(end)))^(1/alpha);
    err1 = abs(f(x(end)));
    err2 = abs(x(end) - x(end - 1));
    if iter >= 2
        ACOC(iter + 2) = (log(abs((x(end) - x(end - 1)))/abs((x(end - 1)...
            - x(end - 2)))))/(log(abs((x(end - 1) - x(end - 2)))...
            /abs((x(end - 2) - x(end - 3)))));
    elseif iter < 2
        ACOC(iter + 2) = Inf;
    end
    iter = iter + 1;
end
x = x(end);
ACOC(1) = [];
t = 1:iter;
plot(t, ACOC), grid
xlabel('Iteraciones')
ylabel('Orden de convergencia')
end
% Ejemplo de uso:
% [x, err1, err2, iter, ACOC] = pruebas_OeCO(-2.2, 0.5, -10)
```

A.29. Método CeCO

```
function [x, err1, err2, iter, ACOC] = pruebas_CeCO(x0, alpha, a)
f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % Funcion 1
df = @(x) (x - a)^(1 - alpha)*(-(1926*x^5)/25 - 128*x^4 + (331*x^3)/5 -...
    (663*x^2)/100 + (2671*x)/50 - 429/100); % Derivada conformable 1
% f = @(x) 1i*x^1.8 - x^0.9 - 16; % Funcion 2
% df = @(x) (x - a)^(1 - alpha)*(1.8*1i*x^0.8 - 0.9*x^-0.1); % Derivada
% conformable 2
% f = @(x) exp(x) - 1; % Funcion 3
% df = @(x) (x - a)^(1-alpha)*exp(x); % Derivada conformable 3
% f = @(x) sin(10*x) - 0.5*x + 0.2; % Funcion 4
% df = @(x) (x - a)^(1-alpha)*(10*cos(10*x)-0.5); % Derivada conformable 4
err1 = Inf;
err2 = Inf;
x = x0*ones(1);
ACOC = Inf*ones(1);
iter = 0;
while iter < 500 && err1 > 1e-8 && err2 > 1e-8
    yk = a + ((x(end) - a)^alpha - alpha*f(x(end))/df(x(end)))^(1/alpha);
    x(iter + 2) = a + ((yk - a)^alpha - alpha*((f(x(end)) + 2*f(yk))/f(x(end)))*...
        f(yk)/df(x(end)))^(1/alpha);
    err1 = abs(f(x(end)));
    err2 = abs(x(end) - x(end - 1));
    if iter >= 2
        ACOC(iter + 2) = (log(abs((x(end) - x(end - 1)))/abs((x(end - 1)...
            - x(end - 2)))))/(log(abs((x(end - 1) - x(end - 2)))...
            /abs((x(end - 2) - x(end - 3)))));
    elseif iter < 2
        ACOC(iter + 2) = Inf;
    end
    iter = iter + 1;
end
x = x(end);
ACOC(1) = [];
t = 1:iter;
plot(t, ACOC), grid
xlabel('Iteraciones')
ylabel('Orden de convergencia')
end
% Ejemplo de uso:
% [x, err1, err2, iter, ACOC] = pruebas_CeCO(-2.2, 0.5, -10)
```

A.30. Plano de convergencia para el método TeCA (ejemplo con $f_1(x)$)

```
% Plano de convergencia: f(x) = -12.84*x^6 - 25.6*x^5 + 16.55*x^4 -
% 2.21*x^3 + 26.71*x^2 - 4.29*x - 15.21
clf, clear all, close all, clc
tic
numNodosx = 400; numNodosy = 400;
maxIter = 500;
tol = 0.001;

f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % problema a resolver
fp = @(x, alpha) ((Gamma(7)*-12.84*x^(6 - alpha))/Gamma(7 - alpha)) +...
    ((Gamma(6)*-25.6*x^(5 - alpha))/Gamma(6 - alpha)) +...
    ((Gamma(5)*16.55*x^(4 - alpha))/Gamma(5 - alpha)) +...
    ((Gamma(4)*-2.21*x^(3 - alpha))/Gamma(4 - alpha)) +...
    ((Gamma(3)*26.71*x^(2 - alpha))/Gamma(3 - alpha)) +...
    ((Gamma(2)*-4.29*x^(1 - alpha))/Gamma(2 - alpha));

r1 = 0.82366 + 0.24769i; r2 = 0.82366 - 0.24769i;
r3 = -2.62297; r4 = -0.584;
r5 = -0.21705 + 0.99911i; r6 = -0.21705 - 0.99911i;

n = @(x, alpha) x - (Gamma(alpha + 1)*(f(x)/fp(x, alpha)))^(1/alpha);
traub = @(x, alpha) n(x, alpha) - (Gamma(alpha + 1)*(f(n(x, alpha))/...
    fp(x, alpha)))^(1/alpha);

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3; xfin = 3; yini = 0.01; yfin = 1;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;

[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);
R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));
[rows, cols] = size(X);
noConv = 0; % inicio contador de puntos que no convergieron

for i = 1:rows
    for j = 1:cols
        x = X(i, j);
        alpha = Y(i, j);
        iter = 0;
        e = Inf;
        while iter < maxIter && e > tol
            an = x;
            x = traub(x, alpha); % Traub_Caputo
            e = abs(an - x);
            iter = iter + 1;
        end
        d1 = abs(r1 - x); % distancia a las raices
        d2 = abs(r2 - x);
        d3 = abs(r3 - x);
        d4 = abs(r4 - x);
        d5 = abs(r5 - x);
        d6 = abs(r6 - x);
        if d1 < tol % convergio a la primera raiz
            R(i, j) = 1 - iter/maxIter;
        elseif d2 < tol % convergio a la segunda raiz
```

```
            G(i, j) = 1 - iter/maxIter;
        elseif d3 < tol % convergio a la tercera raiz
            B(i, j) = 1 - iter/maxIter;
        elseif d4 < tol % convergio a la cuarta raiz
            R(i, j) = 1 - iter/maxIter;
            G(i, j) = 1 - iter/maxIter;
        elseif d5 < tol % convergio a la quinta raiz
            R(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        elseif d6 < tol % convergio a la sexta raiz
            G(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        else
            noConv = noConv + 1;
        end
    end
end
I(:, :, 1) = R(:, :);
I(:, :, 2) = G(:, :);
I(:, :, 3) = B(:, :);

figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
xlabel('$x_0$','Interpreter','latex');
ylabel('$\alpha$','Interpreter','latex');
axis on, axis xy, axis square
densidad = ((numPuntos - noConv)/numPuntos)*100; % porciento exito
t = toc;
savefig('C5_1_traub_caputo_f1_fig.fig')
save('C5_1_traub_caputo_f1_mat.mat')
```

A.31. Plano de convergencia para el método TeRL (ejemplo con $f_1(x)$)

```
% Plano de convergencia: f(x) = -12.84*x^6 - 25.6*x^5 + 16.55*x^4 -
% 2.21*x^3 + 26.71*x^2 - 4.29*x - 15.21
clf, clear all, close all, clc
tic
numNodosx = 400; numNodosy = 400;
maxIter = 500;
tol = 0.001;

f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % problema a resolver
fp = @(x, alpha) ((Gamma(7)*-12.84*x^(6 - alpha))/Gamma(7 - alpha)) +...
    ((Gamma(6)*-25.6*x^(5 - alpha))/Gamma(6 - alpha)) +...
    ((Gamma(5)*16.55*x^(4 - alpha))/Gamma(5 - alpha)) +...
    ((Gamma(4)*-2.21*x^(3 - alpha))/Gamma(4 - alpha)) +...
    ((Gamma(3)*26.71*x^(2 - alpha))/Gamma(3 - alpha)) +...
    ((Gamma(2)*-4.29*x^(1 - alpha))/Gamma(2 - alpha)) +...
    (-15.21/Gamma(1 - alpha))*x^-alpha;

r1 = 0.82366 + 0.24769i; r2 = 0.82366 - 0.24769i;
r3 = -2.62297; r4 = -0.584;
r5 = -0.21705 + 0.99911i; r6 = -0.21705 - 0.99911i;

n = @(x, alpha) x - (Gamma(alpha + 1)*(f(x)/fp(x, alpha)))^(1/alpha);
traub = @(x, alpha) n(x, alpha) - (Gamma(alpha + 1)*(f(n(x, alpha))/...
    fp(x, alpha)))^(1/alpha);

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3; xfin = 3; yini = 0.01; yfin = 1;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;

[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);
R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));
[rows, cols] = size(X);
noConv = 0; % inicio contador de puntos que no convergieron

for i = 1:rows
    for j = 1:cols
        x = X(i, j);
        alpha = Y(i, j);
        iter = 0;
        e = Inf;
        while iter < maxIter && e > tol
            an = x;
            x = traub(x, alpha); % Traub_RL
            e = abs(an - x);
            iter = iter + 1;
        end
        d1 = abs(r1 - x); % distancia a las raices
        d2 = abs(r2 - x);
        d3 = abs(r3 - x);
        d4 = abs(r4 - x);
        d5 = abs(r5 - x);
        d6 = abs(r6 - x);
        if d1 < tol % convergio a la primera raiz
            R(i, j) = 1 - iter/maxIter;
```

```
        elseif d2 < tol % convergio a la segunda raiz
            G(i, j) = 1 - iter/maxIter;
        elseif d3 < tol % convergio a la tercera raiz
            B(i, j) = 1 - iter/maxIter;
        elseif d4 < tol % convergio a la cuarta raiz
            R(i, j) = 1 - iter/maxIter;
            G(i, j) = 1 - iter/maxIter;
        elseif d5 < tol % convergio a la quinta raiz
            R(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        elseif d6 < tol % convergio a la sexta raiz
            G(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        else
            noConv = noConv + 1;
        end
    end
end
I(:, :, 1) = R(:, :);
I(:, :, 2) = G(:, :);
I(:, :, 3) = B(:, :);

figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
xlabel('$x_0$','Interpreter','latex');
ylabel('$\alpha$','Interpreter','latex');
axis on, axis xy, axis square
densidad = ((numPuntos - noConv)/numPuntos)*100; % porciento exito
t = toc;
savefig('C5_5_traub_RL_f1_fig.fig')
save('C5_5_traub_RL_f1_mat.mat')
```

A.32. Plano de convergencia para el método TeCO (ejemplo con $f_1(x)$)

```
% Plano de convergencia: f(x) = -12.84*x^6 - 25.6*x^5 + 16.55*x^4 -
% 2.21*x^3 + 26.71*x^2 - 4.29*x - 15.21
clf, clear all, close all, clc
tic
numNodosx = 400; numNodosy = 400;
maxIter = 500;
tol = 0.001;
a = -10;

f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % problema a resolver
fp = @(x, alpha) (x - a)^(1-alpha)*(-(1926*x^5)/25 - 128*x^4 +...
    (331*x^3)/5 - (663*x^2)/100 + (2671*x)/50 - 429/100);

r1 = 0.82366 + 0.24769i; r2 = 0.82366 - 0.24769i;
r3 = -2.62297; r4 = -0.584;
r5 = -0.21705 + 0.99911i; r6 = -0.21705 - 0.99911i;

n = @(x, alpha) a + ((x - a)^alpha - alpha*f(x)/fp(x, alpha))^(1/alpha);
traub = @(x, alpha) a + ((n(x, alpha) - a)^alpha - alpha*f(n(x, alpha))/...
    fp(x, alpha))^(1/alpha);

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3; xfin = 3; yini = 0.01; yfin = 1;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;

[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);
R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));
[rows, cols] = size(X);
noConv = 0; % inicio contador de puntos que no convergieron

for i = 1:rows
    for j = 1:cols
        x = X(i, j);
        alpha = Y(i, j);
        iter = 0;
        e = Inf;
        while iter < maxIter && e > tol
            an = x;
            x = traub(x, alpha); % Traub_conf
            e = abs(an - x);
            iter = iter + 1;
        end
        d1 = abs(r1 - x); % distancia a las raices
        d2 = abs(r2 - x);
        d3 = abs(r3 - x);
        d4 = abs(r4 - x);
        d5 = abs(r5 - x);
        d6 = abs(r6 - x);
        if d1 < tol % convergio a la primera raiz
            R(i, j) = 1 - iter/maxIter;
        elseif d2 < tol % convergio a la segunda raiz
            G(i, j) = 1 - iter/maxIter;
        elseif d3 < tol % convergio a la tercera raiz
            B(i, j) = 1 - iter/maxIter;
```

```
        elseif d4 < tol % convergio a la cuarta raiz
            R(i, j) = 1 - iter/maxIter;
            G(i, j) = 1 - iter/maxIter;
        elseif d5 < tol % convergio a la quinta raiz
            R(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        elseif d6 < tol % convergio a la sexta raiz
            G(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        else
            noConv = noConv + 1;
        end
    end
end
I(:, :, 1) = R(:, :);
I(:, :, 2) = G(:, :);
I(:, :, 3) = B(:, :);

figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
xlabel('$x_0$','Interpreter','latex');
ylabel('$\alpha$','Interpreter','latex');
axis on, axis xy, axis square
densidad = ((numPuntos - noConv)/numPuntos)*100; % porciento exito
t = toc;
savefig('C5_9_traub_conf_f1_fig.fig')
save('C5_9_traub_conf_f1_mat.mat')
```

A.33. Plano de convergencia para el método CKeCO (ejemplo con $f_1(x)$)

```
% Plano de convergencia: f(x) = -12.84*x^6 - 25.6*x^5 + 16.55*x^4 -
% 2.21*x^3 + 26.71*x^2 - 4.29*x - 15.21
clf, clear all, close all, clc
tic
numNodosx = 400; numNodosy = 400;
maxIter = 500;
tol = 0.001;
a = -10;

f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % problema a resolver
fp = @(x, alpha) (x - a)^(1-alpha)*(-(1926*x^5)/25 - 128*x^4 +...
    (331*x^3)/5 - (663*x^2)/100 + (2671*x)/50 - 429/100);

r1 = 0.82366 + 0.24769i; r2 = 0.82366 - 0.24769i;
r3 = -2.62297; r4 = -0.584;
r5 = -0.21705 + 0.99911i; r6 = -0.21705 - 0.99911i;

n = @(x, alpha) a + ((x - a)^alpha - alpha*f(x)/fp(x, alpha))^(1/alpha);
chk = @(x, alpha) a + ((x - a)^alpha - alpha/2*(3 - fp(n(x, alpha), alpha)/...
    fp(x, alpha))*(f(x)/fp(x, alpha)))^(1/alpha);

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3; xfin = 3; yini = 0.01; yfin = 1;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;

[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);
R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));
[rows, cols] = size(X);
noConv = 0; % inicio contador de puntos que no convergieron

for i = 1:rows
    for j = 1:cols
        x = X(i, j);
        alpha = Y(i, j);
        iter = 0;
        e = Inf;
        while iter < maxIter && e > tol
            an = x;
            x = chk(x, alpha); % Chun_Kim_conf
            e = abs(an - x);
            iter = iter + 1;
        end
        d1 = abs(r1 - x); % distancia a las raices
        d2 = abs(r2 - x);
        d3 = abs(r3 - x);
        d4 = abs(r4 - x);
        d5 = abs(r5 - x);
        d6 = abs(r6 - x);
        if d1 < tol % convergio a la primera raiz
            R(i, j) = 1 - iter/maxIter;
        elseif d2 < tol % convergio a la segunda raiz
            G(i, j) = 1 - iter/maxIter;
        elseif d3 < tol % convergio a la tercera raiz
            B(i, j) = 1 - iter/maxIter;
```

```
        elseif d4 < tol % convergio a la cuarta raiz
            R(i, j) = 1 - iter/maxIter;
            G(i, j) = 1 - iter/maxIter;
        elseif d5 < tol % convergio a la quinta raiz
            R(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        elseif d6 < tol % convergio a la sexta raiz
            G(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        else
            noConv = noConv + 1;
        end
    end
end
I(:, :, 1) = R(:, :);
I(:, :, 2) = G(:, :);
I(:, :, 3) = B(:, :);

figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
xlabel('$x_0$','Interpreter','latex');
ylabel('$\alpha$','Interpreter','latex');
axis on, axis xy, axis square
densidad = ((numPuntos - noConv)/numPuntos)*100; % porciento exito
t = toc;
savefig('C5_21_chun_kim_conf_f1_fig.fig')
save('C5_21_chun_kim_conf_f1_mat.mat')
```

A.34. Plano de convergencia para el método OeCO (ejemplo con $f_1(x)$)

```
% Plano de convergencia: f(x) = -12.84*x^6 - 25.6*x^5 + 16.55*x^4 -
% 2.21*x^3 + 26.71*x^2 - 4.29*x - 15.21
clf, clear all, close all, clc
tic
numNodosx = 400; numNodosy = 400;
maxIter = 500;
tol = 0.001;
a = -10;

f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % problema a resolver
fp = @(x, alpha) (x - a)^(1-alpha)*(-(1926*x^5)/25 - 128*x^4 +...
    (331*x^3)/5 - (663*x^2)/100 + (2671*x)/50 - 429/100);

r1 = 0.82366 + 0.24769i; r2 = 0.82366 - 0.24769i;
r3 = -2.62297; r4 = -0.584;
r5 = -0.21705 + 0.99911i; r6 = -0.21705 - 0.99911i;

n = @(x, alpha) a + ((x - a)^alpha - alpha*f(x)/fp(x, alpha))^(1/alpha);
ostrowski = @(x, alpha) a + ((n(x, alpha) - a)^alpha - alpha*f(x)/(f(x) -...
    2*f(n(x, alpha)))*f(n(x, alpha))/fp(x, alpha))^(1/alpha);

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3; xfin = 3; yini = 0.01; yfin = 1;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;

[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);
R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));
[rows, cols] = size(X);
noConv = 0; % inicio contador de puntos que no convergieron

for i = 1:rows
    for j = 1:cols
        x = X(i, j);
        alpha = Y(i, j);
        iter = 0;
        e = Inf;
        while iter < maxIter && e > tol
            an = x;
            x = ostrowski(x, alpha); % Ostrowski_conf
            e = abs(an - x);
            iter = iter + 1;
        end
        d1 = abs(r1 - x); % distancia a las raices
        d2 = abs(r2 - x);
        d3 = abs(r3 - x);
        d4 = abs(r4 - x);
        d5 = abs(r5 - x);
        d6 = abs(r6 - x);
        if d1 < tol % convergio a la primera raiz
            R(i, j) = 1 - iter/maxIter;
        elseif d2 < tol % convergio a la segunda raiz
            G(i, j) = 1 - iter/maxIter;
        elseif d3 < tol % convergio a la tercera raiz
            B(i, j) = 1 - iter/maxIter;
```

```
        elseif d4 < tol % convergio a la cuarta raiz
            R(i, j) = 1 - iter/maxIter;
            G(i, j) = 1 - iter/maxIter;
        elseif d5 < tol % convergio a la quinta raiz
            R(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        elseif d6 < tol % convergio a la sexta raiz
            G(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        else
            noConv = noConv + 1;
        end
    end
end
I(:, :, 1) = R(:, :);
I(:, :, 2) = G(:, :);
I(:, :, 3) = B(:, :);

figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
xlabel('$x_0$','Interpreter','latex');
ylabel('$\alpha$','Interpreter','latex');
axis on, axis xy, axis square
densidad = ((numPuntos - noConv)/numPuntos)*100; % porciento exito
t = toc;
savefig('C5_13_ostrowski_conf_f1_fig.fig')
save('C5_13_ostrowski_conf_f1_mat.mat')
```

A.35. Plano de convergencia para el método CeCO (ejemplo con $f_1(x)$)

```
% Plano de convergencia: f(x) = -12.84*x^6 - 25.6*x^5 + 16.55*x^4 -
% 2.21*x^3 + 26.71*x^2 - 4.29*x - 15.21
clf, clear all, close all, clc
tic
numNodosx = 400; numNodosy = 400;
maxIter = 500;
tol = 0.001;
a = -10;

f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % problema a resolver
fp = @(x, alpha) (x - a)^(1-alpha)*(-(1926*x^5)/25 - 128*x^4 +...
    (331*x^3)/5 - (663*x^2)/100 + (2671*x)/50 - 429/100);

r1 = 0.82366 + 0.24769i; r2 = 0.82366 - 0.24769i;
r3 = -2.62297; r4 = -0.584;
r5 = -0.21705 + 0.99911i; r6 = -0.21705 - 0.99911i;

n = @(x, alpha) a + ((x - a)^alpha - alpha*f(x)/fp(x, alpha))^(1/alpha);
chun = @(x, alpha) a + ((n(x, alpha) - a)^alpha - alpha*(f(x) + 2*f(n(x, alpha)))/...
    f(x)*f(n(x, alpha))/fp(x, alpha))^(1/alpha);

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3; xfin = 3; yini = 0.01; yfin = 1;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;

[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);
R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));
[rows, cols] = size(X);
noConv = 0; % inicio contador de puntos que no convergieron

for i = 1:rows
    for j = 1:cols
        x = X(i, j);
        alpha = Y(i, j);
        iter = 0;
        e = Inf;
        while iter < maxIter && e > tol
            an = x;
            x = chun(x, alpha); % Chun_conf
            e = abs(an - x);
            iter = iter + 1;
        end
        d1 = abs(r1 - x); % distancia a las raices
        d2 = abs(r2 - x);
        d3 = abs(r3 - x);
        d4 = abs(r4 - x);
        d5 = abs(r5 - x);
        d6 = abs(r6 - x);
        if d1 < tol % convergio a la primera raiz
            R(i, j) = 1 - iter/maxIter;
        elseif d2 < tol % convergio a la segunda raiz
            G(i, j) = 1 - iter/maxIter;
        elseif d3 < tol % convergio a la tercera raiz
            B(i, j) = 1 - iter/maxIter;
```

```
        elseif d4 < tol % convergio a la cuarta raiz
            R(i, j) = 1 - iter/maxIter;
            G(i, j) = 1 - iter/maxIter;
        elseif d5 < tol % convergio a la quinta raiz
            R(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        elseif d6 < tol % convergio a la sexta raiz
            G(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        else
            noConv = noConv + 1;
        end
    end
end
I(:, :, 1) = R(:, :);
I(:, :, 2) = G(:, :);
I(:, :, 3) = B(:, :);

figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
xlabel('$x_0$','Interpreter','latex');
ylabel('$\alpha$','Interpreter','latex');
axis on, axis xy, axis square
densidad = ((numPuntos - noConv)/numPuntos)*100; % porciento exito
t = toc;
savefig('C5_17_chun_conf_f1_fig.fig')
save('C5_17_chun_conf_f1_mat.mat')
```

A.36. Método SeCO

```
function [x, err1, err2, iter, ACOC] = pruebas_SeCO(x0, alpha, a)
f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % Funcion 1
% f = @(x) 1i*x^1.8 - x^0.9 - 16; % Funcion 2
% f = @(x) exp(x) - 1; % Funcion 3
% f = @(x) sin(10*x) - 0.5*x + 0.2; % Funcion 4
err1 = Inf;
err2 = Inf;
x = x0*ones(1);
ACOC = Inf*ones(1);
iter = 0;
while iter < 500 && err1 > 1e-8 && err2 > 1e-8
    fx = f(x(end));
    x(iter + 2) = a + ((x(end) - a)^alpha - alpha*fx^2/...
        (f(x(end) + fx*(x(end) - a)^(1 - alpha)) - fx))^(1/alpha);
    err1 = abs(f(x(end)));
    err2 = abs(x(end) - x(end - 1));
    if iter >= 2
        ACOC(iter + 2) = (log(abs((x(end) - x(end - 1)))/abs((x(end - 1)...
            - x(end - 2)))))/(log(abs((x(end - 1) - x(end - 2)))...
            /abs((x(end - 2) - x(end - 3)))));
    elseif iter < 2
        ACOC(iter + 2) = Inf;
    end
    iter = iter + 1;
end
x = x(end);
ACOC(1) = [];
t = 1:iter;
plot(t, ACOC), grid
xlabel('Iteraciones')
ylabel('Orden_de_convergencia')
end
% Ejemplo de uso:
% [x, err1, err2, iter, ACOC] = pruebas_SeCO(-2.2, 0.5, -10)
```

A.37. Método EeCO

```
function [iter, x, err1, err2, ACOC] = pruebas_EeCO(x0, x1, alpha, a)
f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % Funcion 1
% f = @(x) 1i*x^1.8 - x^0.9 - 16; % Funcion 2
% f = @(x) exp(x) - 1; % Funcion 3
% f = @(x) sin(10*x) - 0.5*x + 0.2; % Funcion 4
x = x0*ones(1);
err1 = Inf;
err2 = Inf;
ACOC = Inf*ones(1);
iter = 0;
while iter < 500 && err1 > 1e-8 && err2 > 1e-8
    fx = f(x(end));
    x(iter + 2) = a + ((x(end) - a)^alpha - alpha*((x1 - x(end))*fx)/...
        (f(x(end) + (x1 - x(end))*(x(end) - a)^(1 - alpha)) - fx))^(1/alpha);
    x1 = x(end - 1);
    err1 = abs(f(x(end)));
    err2 = abs(x(end) - x(end - 1));
    if iter >= 2
        ACOC(iter + 2) = (log(abs((x(end) - x(end - 1)))/abs((x(end - 1)...
            - x(end - 2)))))/(log(abs((x(end - 1) - x(end - 2)))...
            /abs((x(end - 2) - x(end - 3)))));
    elseif iter < 2
        ACOC(iter + 2) = Inf;
    end
    iter = iter + 1;
end
x = x(end);
ACOC(1) = [];
t = 1:iter;
plot(t, ACOC), grid
xlabel('Iteraciones')
ylabel('Orden de convergencia')
end
% Ejemplos de uso:
% [iter, x, err1, err2, ACOC] = pruebas_EeCO(0.8, 1, 1, -10)
% [iter, x, err1, err2, ACOC] = pruebas_EeCO(-2.2, -2, 0.5, -10)
```

A.38. Plano de convergencia para el método SeCO (ejemplo con $f_1(x)$)

```
% Plano de convergencia: f(x) = -12.84*x^6 - 25.6*x^5 + 16.55*x^4 -
% 2.21*x^3 + 26.71*x^2 - 4.29*x - 15.21
clf, clear all, close all, clc
tic
numNodosx = 400; numNodosy = 400;
maxIter = 500;
tol = 0.001;
a = -10;

f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % problema a resolver

r1 = 0.82366 + 0.24769i; r2 = 0.82366 - 0.24769i;
r3 = -2.62297; r4 = -0.584;
r5 = -0.21705 + 0.99911i; r6 = -0.21705 - 0.99911i;

s = @(x, alpha) a + ((x - a)^alpha - alpha*f(x)^2/(f(x + f(x)*(x - a)^...
    (1 - alpha)) - f(x)))^(1/alpha);

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3; xfin = 3; yini = 0.01; yfin = 1;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;

[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);
R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));
[rows, cols] = size(X);
noConv = 0; % inicio contador de puntos que no convergieron

for i = 1:rows
    for j = 1:cols
        x = X(i, j);
        alpha = Y(i, j);
        iter = 0;
        e = Inf;
        while iter < maxIter && e > tol
            an = x;
            x = s(x, alpha); % Steffensen_conf
            e = abs(an - x);
            iter = iter + 1;
        end
        d1 = abs(r1 - x); % distancia a las raices
        d2 = abs(r2 - x);
        d3 = abs(r3 - x);
        d4 = abs(r4 - x);
        d5 = abs(r5 - x);
        d6 = abs(r6 - x);
        if d1 < tol % convergio a la primera raiz
            R(i, j) = 1 - iter/maxIter;
        elseif d2 < tol % convergio a la segunda raiz
            G(i, j) = 1 - iter/maxIter;
        elseif d3 < tol % convergio a la tercera raiz
            B(i, j) = 1 - iter/maxIter;
        elseif d4 < tol % convergio a la cuarta raiz
            R(i, j) = 1 - iter/maxIter;
            G(i, j) = 1 - iter/maxIter;
```

```
        elseif d5 < tol % convergio a la quinta raiz
            R(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        elseif d6 < tol % convergio a la sexta raiz
            G(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        else
            noConv = noConv + 1;
        end
    end
end
I(:, :, 1) = R(:, :);
I(:, :, 2) = G(:, :);
I(:, :, 3) = B(:, :);

figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
xlabel('$x_0$','Interpreter','latex');
ylabel('$\alpha$','Interpreter','latex');
axis on, axis xy, axis square
densidad = ((numPuntos - noConv)/numPuntos)*100; % porciento exito
t = toc;
savefig('C6_9_steffensen_conf_f1_fig.fig')
save('C6_9_steffensen_conf_f1_mat.mat')
```

A.39. Plano de convergencia para el método EeCO (ejemplo con $f_1(x)$)

```
% Plano de convergencia: f(x) = -12.84*x^6 - 25.6*x^5 + 16.55*x^4 -
% 2.21*x^3 + 26.71*x^2 - 4.29*x - 15.21
clf, clear all, close all, clc
tic
numNodosx = 400; numNodosy = 400;
maxIter = 500;
tol = 0.001;
a = -10;

f = @(x) -12.84*x^6 - 25.6*x^5 + 16.55*x^4 - 2.21*x^3 +...
    26.71*x^2 - 4.29*x - 15.21; % problema a resolver

r1 = 0.82366 + 0.24769i; r2 = 0.82366 - 0.24769i;
r3 = -2.62297; r4 = -0.584;
r5 = -0.21705 + 0.99911i; r6 = -0.21705 - 0.99911i;

secante = @(x, x1, alpha) a + ((x - a)^alpha - alpha*(x1 - x)*f(x)/(f(x +...
    (x1 - x)*(x - a)^(1 - alpha)) - f(x)))^(1/alpha);

numPuntos = (numNodosx + 1)*(numNodosy + 1);
xini = -3; xfin = 3; yini = 0.01; yfin = 1;
dx = (xfin - xini)/numNodosx;
dy = (yfin - yini)/numNodosy;

[X, Y] = meshgrid(xini:dx:xfin, yini:dy:yfin);
R = zeros(size(X)); G = zeros(size(X)); B = zeros(size(X));
[rows, cols] = size(X);
noConv = 0; % inicio contador de puntos que no convergieron

for i = 1:rows
    for j = 1:cols
        x = X(i, j);
        x1 = x + 1;
        alpha = Y(i, j);
        iter = 0;
        e = Inf;
        while iter < maxIter && e > tol
            an = x;
            x = secante(x, x1, alpha); % Secante_conf
            x1 = an;
            e = abs(an - x);
            iter = iter + 1;
        end
        d1 = abs(r1 - x); % distancia a las raices
        d2 = abs(r2 - x);
        d3 = abs(r3 - x);
        d4 = abs(r4 - x);
        d5 = abs(r5 - x);
        d6 = abs(r6 - x);
        if d1 < tol % convergio a la primera raiz
            R(i, j) = 1 - iter/maxIter;
        elseif d2 < tol % convergio a la segunda raiz
            G(i, j) = 1 - iter/maxIter;
        elseif d3 < tol % convergio a la tercera raiz
            B(i, j) = 1 - iter/maxIter;
        elseif d4 < tol % convergio a la cuarta raiz
```

```
            R(i, j) = 1 - iter/maxIter;
            G(i, j) = 1 - iter/maxIter;
        elseif d5 < tol % convergio a la quinta raiz
            R(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        elseif d6 < tol % convergio a la sexta raiz
            G(i, j) = 1 - iter/maxIter;
            B(i, j) = 1 - iter/maxIter;
        else
            noConv = noConv + 1;
        end
    end
end
I(:, :, 1) = R(:, :);
I(:, :, 2) = G(:, :);
I(:, :, 3) = B(:, :);

figura = imshow(I, 'X', [xini, xfin], 'Y', [yini, yfin]);
xlabel('$x_0$','Interpreter','latex');
ylabel('$\alpha$','Interpreter','latex');
axis on, axis xy, axis square
densidad = ((numPuntos - noConv)/numPuntos)*100; % porciento exito
t = toc;
savefig('C6_13_secante_conf_f1_fig.fig')
save('C6_13_secante_conf_f1_mat.mat')
```

Bibliografía

[1] Abdeljawad, T., *On conformable fractional calculus*, Journal of Computational and Applied Mathematics, 279 (2015), 57–66.

[2] Abramowitz, M., Stegun, I. A., *Handbook of Mathematical Functions*, Dover Publications, New York, 1970.

[3] Akgül, A., Cordero, A., Torregrosa, J. R., *A fractional Newton method with 2αth-order of convergence and its stability*, Applied Mathematics Letters, 98 (2019), 344–351.

[4] Amiri, A.R., Cordero, A., Darvishi, M. T., Torregrosa, J. R., *Preserving the order of convergence: Low-complexity Jacobian-free iterative schemes for solving nonlinear systems*, Journal of Computational and Applied Mathematics, 337 (2018), 87–97.

[5] Atanackovic, T. M., Pilipovic, S., Stankovic, B., Zorica, D., *Fractional Calculus with Applications in Mechanics: Wave Propagation, Impact and Variational Principles*, Wiley, London, 2014.

[6] Atangana, A., Baleanu, D., Alsaedi, A., *New properties of conformable derivative*, Open Mathematics, 13 (2015), 889–898.

[7] Benjemaa, M., *Taylor's formula involving generalized fractional derivatives*, Applied Mathematics and Computation, 335 (2018), 182–195.

[8] Brambila, F., Torres, A., *Fractional Newton-Raphson Method*, arXiv 2017, arXiv:1710.07634v3.

[9] Bronshtein, I. N., Semendyayev, K. A., Musiol, G., Muehlig, H., *Handbook of Mathematics*, Springer, Berlin, 2007.

[10] Candelario, G., Cordero, A., Torregrosa, J. R., *Multipoint Fractional Iterative Methods with (2α + 1)th-Order of Convergence for Solving Nonlinear Problems*, Mathematics, 452 (2020), https://doi.org/10.3390/math8030452.

[11] Candelario, G., Cordero, A., Torregrosa, J. R., Vassileva, M. P., *An optimal and low computational cost fractional Newton-type method for solving nonlinear equations*, Applied Mathematics Letters, 124 (2022), https://doi.org/10.1016/j.aml.2021.107650.

[12] Candelario, G., Cordero, A., Torregrosa, J. R., Vassileva, M. P., *Generalized conformable fractional Newton-type method for solving nonlinear systems*, Numerical Algorithms, (2023), https://doi.org/10.1007/s11075-022-01463-z.

[13] Candelario, G., Cordero, A., Torregrosa, J. R., Vassileva, M. P., *On a new technique to generate optimal multipoint fractional methods for solving nonlinear equations*, Journal of Mathematical Analysis and Applications, (2023).

[14] Chun, C., Kim, Y. I., *Several New Third-Order Iterative Methods for Solving Nonlinear Equations*, Acta applicandae mathematicae, 109 (2010), 1053—1063.

[15] Cordero, A., Girona, I., Torregrosa, J. R., *A Variant of Chebyshev's Method with 3αth-Order of Convergence by Using Fractional Derivatives*, Symmetry, 11 (2019), doi:10.3390/sym11081017.

[16] Cordero, A., Hueso, J. L., Martínez, E., Torregrosa, J. R., *A modified Newton-Jarrat's composition*, Numerical Algorithms, 55 (2009), 87–99.

[17] Cordero, A., Torregrosa, J. R., *Variants of Newton's method using fifth order quadrature formulas*, Applied Mathematics and Computation, 190 (2007), 686–698.

[18] Curiel, R. P., *El cálculo generalizado y las funciones fraccionarias*, Tesis de Maestría (2009), Instituto Tecnológico Autónomo de México.

[19] Graham, R. L., Knuth, D. E., Patashnik, O., *Concrete Mathematics*, Addison-Wesley Longman Publishing, Boston MA, 1994.

[20] Horn, R. A., Johnson, C. R., *Topics in matrix analysis*, Cambridge University Press, New York, 1991.

[21] Ishteva, M. K., *Properties and Applications of the Caputo Fractional Operator*, Master Thesis (2005), Department of Mathematics, Universität Karlsruhe (TH).

[22] Jumarie, G., *Modified Riemann-Liouville Derivative and Fractional Taylor Series of Nondifferentiable Functions Further Results*, Computers and Mathematics with Applications, 51 (2006), 1367–1376.

[23] Khalil, R., Al Horani, M., Yousef, A., Sababheh, M., *A new definition of fractional derivative*, Journal of Computational and Applied Mathematics, 264 (2014), 65–70.

[24] Khan, M. A., Ullah, S., Farhan, M., *The dynamics of Zika virus with Caputo fractional derivative*, AIMS Mathematics, 4 (2019), 134–146.

[25] King, R. F., *A Family of Fourth Order Methods for Nonlinear Equations*, SIAM Journal on Numerical Analysis, 10 (1973), 876—879.

[26] Kung, H. T., Traub, J. F., *Optimal Order of One-Pont and Multipoint Iteration*, Journal of the Association for Computing Machinery, 21 (1974), 643–651.

[27] Lanczos, C., *A precision approximation of the gamma function*, SIAM Journal on Numerical Analysis, 1 (1964), 86–96.

[28] Lazarević, M., *Advanced Topics on Applications of Fractional Calculus on Control Problems, System Stability and Modeling*, WSEAS Press, Belgrade, 2014.

[29] Magreñan, Á. A., *A new tool to study real dynamics: The Convergence Plane*, Applied Mathematics and Computation, 248 (2014), 215–224.

[30] Mathai, A. M., Haubold, H.J., *Fractional and Multivariable Calculus, Model Building and Optimization Problems*, Springer, Berlin, 2017.

[31] Miller, K. S., Ross, B., *An introduction to the fractional calculus and fractional differential equations*, John Wiley & Sons, New York, 1993.

[32] Odibat, Z. M., Shawagfeh, N. T., *Generalized Taylor's formula*, Applied Mathematics and Computation, 186 (2007), 286–293.

[33] Oldham, K. B., Spanier, J., *The fractional calculus*, Academic Press, New York-London, 1974.

[34] Ortega, J. M., Rheinboldt, W. C., *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[35] Pertz, G. H., Gerhardt, C. J., *Leibnizens gesammelte Werke*, Wentworth Press, 2018.

[36] Petković, M. S., Neta, B., Petković, L. D., Džunić, J., *Multipoint Methods for Solving Nonlinear Equations*, Elsevier, USA, 2013.

[37] Podlubny, I., *Fractional differential equations*, Academic Press, San Diego, 1999.

[38] Radwan, A. G., Khanday, F. A., Said, L. A., *Fractional-Order Modeling of Dynamic Systems with Application in Optimization, Signal Processing, and Control*, Academic Press, London UK, 2022.

[39] Ross, B., *A brief history and exposition of the fundamental theory of fractional calculus*, Springer, 1975.

[40] Toprakseven, Ş., *Numerical Solutions of Conformable Fractional Differential Equations by Taylor and Finite Difference Methods*, Journal of Natural and Applied Sciences, 23 (2019), 850–863.

[41] Torres, A., Brambila, F., *Fractional Newton-Raphson Method and Some Variants for the Solution of Non-linear Systems*, arXiv 2020, arXiv:1908.01453v6.

[42] Traub, J. F., *Iterative Methods for the Solution of Equations*, Prentice Hall, New York, 1964.

[43] Weilbeer, M., *Efficient Numerical Methods for Fractional Differential Equations and their Analytical Background*, PhD Thesis (2005), Fakultät für Mathematik und Informatik der Technischen Universität Braunschweig.