ICT-UNPA-36-2012
ISSN: 1852- 4516
Aprobado por Resolución Nro. 0236/12-R-UNPA

USO DE EVOLUCIÓN DIFERENCIAL PARA LA BÚSQUEDA DE SEMILLAS EN EL MÉTODO DE NEWTON

Sebastián Hernández [a]

Guillermo Leguizamón [b]

Departamento de Ciencias Exactas y Naturales, UARG, Universidad Nacional de la Patagonia Austral

bDepartamento de Informática, Universidad Nacional de San Luis

Resumen

Una adecuada performance del método de Newton requiere una eficiente elección de la semilla, fundamentalmente en los casos en que la convergencia para la obtención de raéces sea altamente sensible a este punto inicial. En este trabajo se propone el uso de un método de optimización, denominado evolución diferencial, para generar los puntos iniciales necesarios que aseguren un eficiente uso del método de Newton. Este método es una metaheurística que tiene origen en las ideas de algoritmos evolutivos y el método del gradiente. Aquí se presentan la implementación de la propuesta -codificada en MATLAB-, ejemplos de aplicación y comparación de resultados obtenidos, con y sin la inclusión de esta mejora.

Palabras Clave: método de Newton, evolución diferencial, sistemas no lineales, metaheurísticas.

1 INTRODUCCIÓN

Uno de los algoritmos tradicionales usados para el cálculo de raíces de ecuaciones -o sistemas de ecuaciones- no lineales de una variable es el método de Newton y sus múltiples variantes posteriores (Tjalling1995,Oezban2004). Éste es considerado un método abierto ya que su convergencia global no está garantizada. En la mayoría de los casos, la única manera de alcanzar la convergencia es seleccionar un valor inicial lo suficientemente cercano a la raíz buscada. El método de Newton surge cuando se considera el problema de hallar una raíz real de una función $f : I \subseteq \mathbb{R} \to \mathbb{R}$. Este cero puede ser determinado como un punto fijo de una función de iteración a través del método del punto fijo:

$$x_{k+1} = h\left(x_k\right), \tag{1}$$

donde x_0 es el valor inicial. El ejemplo más conocido y ampliamente usado de este tipo de iteraciones es el clásico método de Newton, dado por:

$$x_{k+1} = x_k - \frac{f\left(x_k\right)}{f\prime\left(x_k\right)}. \tag{2}$$

La principal complicación radica en que la función derivada no puede anularse durante el proceso. Para el cálculo de sistemas del tipo

$$F\left(\mathbf{x}\right) = \begin{cases} f_1\left(x_1, x_2, ..., x_n\right) = 0 \\ f_2\left(x_1, x_2, ..., x_n\right) = 0 \\ \vdots \\ f_n\left(x_1, x_2, ..., x_n\right) = 0 \end{cases} \tag{3}$$

donde cada $f_i : \mathbb{R}^n \to \mathbb{R}, i = 1, 2, ..., n$ puede ser una función no lineal con respecto a cualquiera de sus variables, se utiliza la versión extendida a varias variables del método de Newton. Su forma general de iteración es

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - J_F\left(\mathbf{x}^{(n)}\right)^{-1} F\left(\mathbf{x}^{(n)}\right), \tag{4}$$

donde $J_F\left(\mathbf{x}^{(k)}\right)^{-1}$ es la inversa de la matriz jacobiana del sistema. Sumada a las complicaciones clásicas del cálculo numérico (mal condicionamiento de la matriz jacobiana; errores de redondeo y de aritmética en la obtención de $J_F\left(x\right)^{-1}$ entre otras), surge un problema mayor: que la matriz jacobiana tenga determinante nulo. En este caso este método no puede aplicarse.

Si el sistema a resolver no tiene los inconvenientes citados anteriormente, la principal dificultad es usar un valor inicial cercano a la solución como para asegurar la convergencia del método de Newton. A fines prácticos, suelen emplearse métodos que propongan un adecuado valor inicial o semilla para la variable de iteración (Burden2000,Oezban2004).

En este trabajo se propone utilizar un método de optimización llamado Evolución Diferencial (ED) (Storn1995,Storn1996), con el fin de obtener puntos cercanos a la solución para ser usados como semillas en el método de Newton. Aquí se emplean transformaciones lineales por lo que siempre puede ejecutarse, aún resultando en muchos casos más lento que los métodos basados en iteraciones de punto fijo. La solución obtenida no es de la misma calidad que la obtenida por métodos basados en derivadas, pero siempre resulta mejor que escoger el punto inicial al azar y por tanto puede ser usado como semilla inicial de buena

calidad para los métodos tipo Newton.

El método de evolución diferencial es una metaheurística basada en ideas de algoritmos evolutivos y método del gradiente. Se define el espacio de búsqueda y se crean individuos al azar en ese espacio de búsqueda. Para cada uno de ellos (padre) se seleccionan otros dos de la misma población. Se calcula el vector diferencia entre estos dos individuos. Ese vector se suma al original (padre) y este nuevo vector resulta ser el hijo para la próxima población. Ahora se mide la función de adaptabilidad de la población de padres y de hijos, y se selecciona el que mejor ajusta en cada caso (mutación) sea éste padre o hijo. Por otra parte, para aportar diversidad genética se realiza el cruzamiento o *crossover*, que consiste en seleccionar algunas posiciones de la población y tomar una parte del individuo padre y otra del individuo hijo (de la misma posición) para generar un nuevo individuo que será comparado con el mejor entre padre e hijo. Es decir, se elegirá el mejor entre tres empleando la función de ajuste. Este proceso se repite a través de una cantidad previamente establecida de generaciones (o tiempo de vida de la población). Luego, el mejor individuo de la población final es la solución del problema.

Una de las principales ventajas de la ED es que el proceso principal por el que evoluciona la población numérica es una transformación lineal. Esto evita que se deban calcular jacobianos cuya principal desventaja radica en los tiempos de cómputos necesarios, errores de punto flotante y posibles existencias de matrices mal condicionadas. También es importante recalcar que al ser una técnica metaheurística, no asegura una convergencia a una solución globalmente óptima, aunque puede encontrar soluciones sub-óptimas de muy buena calidad que sirvan como semilla inicial de un método tipo Newton.

La organización del trabajo será de acuerdo a las siguientes secciones: (2) Nociones preliminares: Método de Newton; (3) Método acelerador: Evolución Diferencial; (4) Codificación en Matlab de ED; (5) Ejemplos de aplicación, (6) Evaluación comparativa y (7) Conclusiones.

2 NOCIONES PRELIMINARES: MÉTODO DE NEWTON

Se considera el sistema no lineal de ecuaciones con n incógnitas $F(\mathbf{x})$ Sea $\mathbf{x}^{(0)}$ una aproximación inicial a la solución $\overline{\mathbf{x}}$ del sistema. Luego se puede aplicar el siguiente resultado, que no es otra cosa que la generalización del teorema fundamental del cálculo integral para funciones de varias variables.

Lema 1 *Sea* $F : D \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}^n$ *continuamente diferenciable en un conjunto convexo . Entonces, para cualesquiera , se verifica*

$$F(\mathbf{y}) - F(\mathbf{x}) = \int_0^1 J_F(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))(\mathbf{y} - \mathbf{x})\, dt \tag{5}$$

Si se conoce la iteración k-ésima, es decir $\mathbf{x}^{(k)}$, se puede determinar la iteración $k+1$. A partir de la expresión del lema anterior, se tiene que

$$F(\mathbf{y}) = F(\mathbf{x}^{(k)}) + \int_0^1 J_F(\mathbf{x}^{(k)} + t(\mathbf{y} - \mathbf{x}^{(k)}))(\mathbf{y} - \mathbf{x}^{(k)})\, dt \tag{6}$$

y si se aproxima la función integrando por el valor que toma en el extremo inferior del intervalo de integración, es decir, por su valor en $t = 0$ y se reemplaza por $\overline{\mathbf{x}}$. resulta

$$0 \approx F\left(\mathbf{x}^{(k)}\right) + J_F\left(\mathbf{x}^{(k)}\right)\left(\overline{\mathbf{x}} - \mathbf{x}^{(k)}\right) \tag{7}$$

de donde se puede obtener una nueva aproximación a $\overline{\mathbf{x}}$

$$\overline{\mathbf{x}}^{(k+1)} = \overline{\mathbf{x}}^{(k)} - J_F\left(\mathbf{x}^{(k)}\right)^{-1} F\left(\mathbf{x}^{(k)}\right) \tag{8}$$

lo que es la expresión iterativa para el método de Newton aplicado a funciones de varias variables.

3 MÉTODO ACELERADOR: EVOLUCIÓN DIFERENCIAL

Como la mayoría de las metaheurísticas basadas en algoritmos evolutivos (Goldberg, 1989; Brindle, 1981), la Evolución Diferencial es un optimizador que procesa una población de soluciones potenciales del problema a resolver. La utilidad principal es atacar un problema de punto inicial (semilla) por medio de la aplicación de una función de ajuste (*fitness*) a una serie de vectores numéricos escogidos al azar (Fogel, 2000; DeJong, 2006). Se siguen claramente cuatro etapas: Inicialización; Mutación; Cruzamiento y Selección.

- **Inicialización**. Antes de que la población de vectores sea inicializada, se debe definir un límite superior e inferior para cada una de los variables a utilizar, b_U y b_L. respectivamente. Una vez determinados, un generador de números aleatorios crea, para cada variable del vector, un valor numérico entre b_U y b_L. Por ejemplo, el valor inicial, g=0, de la j-ésima variable de i-ésimo vector es

$$x_{j,i,0} = \text{rand}_j\,[0,1) \cdot (b_{j,U} - b_{j,L}) + b_{j,L}. \tag{9}$$

  El generador de números aleatorios se ejecuta una vez para cada valor de variable, lo que asegura una población completamente aleatoria. Sin importar que una variable sea discreta o continua, debe ser inicializada con un valor continuo, puesto que ED trabaja con números decimales de punto flotante. Cada uno de esos vectores generados es identificado con un número desde 0 hasta . N_{p-1}

- **Mutación**. Como otros métodos basados en poblaciones, ED genera nuevos puntos que no son otra cosa que perturbaciones de puntos existentes. ED perturba, uno a uno, todos los puntos de la población sumándole la diferencia escalada de dos puntos diferentes ($\mathbf{x}_{r1}$ y $\mathbf{x}_{r2}$) de la misma población escogidos al azar. Para el vector se genera el vector por medio de la siguiente definición

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot \left(\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}\right), \tag{10}$$

  donde es el valor de escala utilizado, generalmente un valor aleatorio entre y . Una vez calculado éste se evalúa por medio de la función de ajuste, y si es más apto reemplazará al vector en la siguiente población. Lo mismo ocurre con todos los individuos de la población. Éste es el proceso denominado *mutación diferencial*.

- **Cruzamiento**. A fin de agregarle diversidad genética a la población, se utiliza otro proceso evolutivo denominado *cruzamiento uniforme*. También identificado como *recombinación discreta*, el cruzamiento genera vectores de prueba utilizando información que copia de dos diferentes vectores. En particular, ED cruza cada vector con otro vector mutante dependiendo de la probabilidad de cruzamiento de la población

. $(C_r).\ C_r \in (0,1)$ es un valor definido por el usuario que controla qué valores de variables serán copiados al mutante desde el vector sin mutar:

$$\mathbf{u}_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g} \text{ si } \text{rand}_j\,(0,1) \leq C_r \\ x_{j,i,g} \text{ en otro caso.} \end{cases} \tag{11}$$

- **Selección**. Si el vector de prueba generado luego del cruzamiento, , tiene un valor de ajuste mejor que el vector original, lo reemplazará en la población. De otra forma, permanecerá en la población por, al menos, una generación más. En nuestro caso, como queremos obtener un cero (o un mínimo en valor absoluto), la función de selección es

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} \text{ si } f\,(\mathbf{u}_{i,g}) < f\,(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} \text{ en otro caso.} \end{cases} \tag{12}$$

4 CODIFICACIÓN EN MATLAB DE ED

Para codificar el algoritmo de ED, se utilizó Matlab 5.3. Si bien esta versión tiene algún tiempo en el mercado, es una de las últimas que conserva el comando `flops`, encargado de estimar la cantidad de operaciones de punto flotante que se efectúan a lo largo de una serie de cálculos. En las versiones 6 y posterior, se eliminó el comando `flops`, porque el *kernel* de Matlab incorpora LAPack (*Linear Algebra Package*) con lo que reduce la cantidad de operaciones de punto flotante en la inversión de matrices, además, es muy complicado estimar la cantidad de flops en matrices *sparse* sin agregar operaciones al kernel. La sintaxis del ED es `ED(f, pop, datos, t, Cr, Es)`, donde f representa la función a minimizar; `pop` es el tamaño de la población, es decir la cantidad de vectores a utilizar; `datos` es una matriz que representa los valores límite para cada una de las variables involucradas; `Cr` es la probabilidad de cruzamiento, se sugiere en la bibliografía básica elegir `Cr=1/pop`; es el parámetro de escala a utilizar en la mutación. En el Apéndice se encuentra la codificación completa para MATLAB.

Lo que se debe remarcar es que para obtener raíces de sistemas de ecuaciones no lineales, se debe rescribir el sistema de la siguiente forma:

$$F\,(\mathbf{x}) = \begin{cases} f_1\,(x_1, x_2, \ldots, x_n) = 0 \\ f_2\,(x_1, x_2, \ldots, x_n) = 0 \\ \qquad\vdots \\ f_2\,(x_1, x_2, \ldots, x_n) = 0 \end{cases} \Rightarrow f\,(\mathbf{x}) = \sum_{i=1}^{n} \left| f_i\,(x_1, x_2, \ldots, x_n) \right|, \tag{13}$$

con lo que se minimiza $\mathbf{f}\,(\mathrm{x})$ si $\mathrm{F}\,(\mathrm{x}) \rightarrow 0$.

5 EJEMPLOS DE APLICACIÓN

Para evaluar la estabilidad y convergencia de ED en sistemas no lineales, se proponen cuatro ejemplos cuya complejidad es diversa. Se utilizarán diferentes parémetros, de acuerdo a cada problema. Sin embargo, para todos los ejemplos siguientes, Es, es decir el parámetro de escala utilizado en la mutación tomará siempre el valor 0,75 .

5.1 EJEMPLO 1

(Quarteroni et al.,2000), plantea el sistema de ecuaciones no lineales:

$$F(\mathbf{x}) = \begin{cases} e^{x_1^2+x_2^2} - 1 = 0 \\ e^{x_1^2-x_2^2} - 1 = 0 \end{cases}, \qquad (14)$$

cuya solución exacta es x=(0;0). Resolviendo por medio de ED a lo largo de 30 ejecuciones, con una población de 25 individuos por vez; tomando individuos en el espacio inicial ; con un tiempo de vida de 25 generaciones y una probabilidad de cruzamiento de 1/25, se obtiene como mejor solución el vector $x_s = (1{,}495 \times 10^{-3}; -1{,}928 \times 10^{-3})$ con un valor de fitness de $\mathbf{f}(x_s) = 7{,}438 \times 10^{-6}$ y un valor real de

$$F(\mathbf{x}_s) = \begin{cases} 5{,}952 \times 10^{-6} \approx 0 \\ -1{,}482 \times 10^{-6} \approx 0 \end{cases} \qquad (15)$$

Además, se obtuvo la siguiente salida luego de las 30 ejecuciones, lo que demuestra la robustez del algoritmo de ED sobre este sistema de ecuaciones no lineales:

Mediana: $4{,}506 \times 10^{-5}$

Promedio: $8{,}975 \times 10^{-5}$

Desviación Estándar: $8{,}477 \times 10^{-5}$

Máximo fitness: $2{,}271 \times 10^{-4}$

Mínimo fitness: $7{,}438 \times 10^{-6}$

5.2 EJEMPLO 2

Según Burden (Burden y Faires, 2000), el sistema de ecuaciones no lineales

$$F(\mathbf{x}) = \begin{cases} \frac{4\pi-1}{4\pi}(e^{2x_1} - e) + 4ex_2^2 - 2ex_1 = 0 \\ \sin(4\pi x_1 x_2) - 2x_2 - x_1 = 0 \end{cases}, \qquad (16)$$

es complicado de resolver y tiene varias soluciones, dentro de las cuales podemos identificar x1 = (1,0331; −0,2800) y x2 = (−0,3737; 0,0563). Ejecutando el algoritmo de ED 30 veces, con una población de 50 individuos; tomando individuos al azar en el espacio [−5, 5; −5, 5]; con un tiempo de vida de 40 generaciones y una probabilidad de cruzamiento de 1/50, se obtiene el vector solución xs = (−3,736 × 10−1 ; 5,671 × 10−2) , con un valor de fitness de f (xs) = 3,136 × 10−3 y un valor real de

$$F(\mathbf{x}_s) = \begin{cases} 9{,}698 \times 10^{-5} \approx 0 \\ -2{,}927 \times 10^{-3} \approx 0 \end{cases} \qquad (17)$$

Los siguientes estadísticos fueron calculados durante las 30 ejecuciones del ED:

Mediana: $3{,}009 \times 10^{-2}$

Promedio: $3{,}674 \times 10^{-2}$

Desviación Estándar: $2{,}585 \times 10^{-2}$

Máximo fitness: $9{,}520 \times 10^{-2}$

Mínimo fitness: $3{,}136 \times 10^{-3}$

5.3 EJEMPLO 3

También del libro de Burden (Burden y Faires, 2000), el sistema de ecuaciones no lineales

$$F(\mathbf{x}) = \begin{cases} \ln(x_1^2 + x_2^2) - \sin(x_1 x_2) = \ln(2) + \ln(\pi) \\ e^{x_1 - x_2} + \cos(x_1 x_2) = 0 \end{cases}, \tag{18}$$

tiene una gran cantidad de soluciones. Ejecutando ED 30 veces, con una población de 80 indi- viduos por ejecución; generando la población inicial en el espacio [−10, 10; −10, 10]; con un tiempo de vida de 50 generaciones y una probabilidad de cruzamiento de 1/80, se obtiene el vector solución xs = (−3,965; 1,186) , con un valor de fitness de f (xs) = 7,455 × 10−3 y un

valor real de

$$F(\mathbf{x}_s) = \begin{cases} 2{,}876 \times 10^{-3} \approx 0 \\ -4{,}105 \times 10^{-3} \approx 0 \end{cases} \tag{19}$$

Las características de salida del ED son las siguientes:

Mediana: 2,885 × 10−2
Promedio: 3,189 × 10−2
Desviación Estándar: 2,286 × 10−2
Máximo fitness: 8,847 × 10−2
*Mínimo fitnes*s: 7,455 × 10−3

5.4 EJEMPLO 4

Según Yang (Yang et al., 2004), el sistema de ecuaciones no lineales

$$F(\mathbf{x}) = \begin{cases} xe^y - x^5 + y = 3 \\ x + y + \tan(x) - \sin(y) = 0 \end{cases}, \tag{20}$$

posee varias soluciones. Resolviendo el sistema con ED a lo largo de 30 ejecuciones, donde la población utilizada es de 350 individuos; el espacio de búsqueda es [−5, 5; −5, 5]; el tiempo de vida de la población es 150 generaciones y la probabilidad de cruzamiento es de 1/50, el problema tiene como solución al vector xs = (1,837; 2,458), con un valor de fitness de f (xs) = 6,403 × 10−3 y un valor real de

$$F(\mathbf{x}_s) = \begin{cases} -2{,}455 \times 10^{-3} \approx 0 \\ -3{,}948 \times 10^{-3} \approx 0 \end{cases} \tag{21}$$

Las 30 ejecuciones se resumen en los siguientes estadísticos:

Mediana: 1,174
Promedio: $7{,}045 \times 10^{-1}$

Desviación Estándar: $6{,}064 \times 10^{-1}$
Máximo fitness: 1,174
Mínimo fitness: $6{,}403 \times 10^{-3}$

6 EVALUACIÓN COMPARATIVA

Los cuatro sistemas de ecuaciones antes mencionados, se resolvieron primero por el método de Newton, con semillas aleatorias en los espacios indicados; luego por medio de ED, con semillas aleatorias y parámetros específicos para cada sistema y finalmente se utilizó la solución obtenida por ED como semilla para resolverlo por el método de Newton. Se muestran cuatro ejecuciones por cada método resolutivo. Se analiza la cantidad de *flops* que requiere cada una de las estrategias utilizadas y se mide la calidad de la solución obtenida. Los resultados se muestran en tablas.

6.1 EJEMPLO 1

Resolviendo el sistema (14) a través del método de Newton; se tomaron cuatro vectores aleatorios de forma tal que x_1, x_2 pertenece [−4, 4]; con un máximo de 50 ciclos y se iteró hasta que el error relativo entre x_{n-1} y x_n sea menor a 0,001. Los resultados se resumen en la tabla 1.

Semilla	Solución	Iter.	*Flops*	F (x)
(−0,4662; 3,4011)	Matriz singular	-	-	-
(−1,1425; −3,0249)	Matriz singular	-	-	-
(3,9345; 1,7527)	$(0{,}75;\ 0{,}69) \times 10^{-8}$	47	6670	(0,00; 0,00)
(−3,8935; 2,0607)	$(-0{,}55;\ 0{,}55) \times 10^{-8}$	48	6814	(0,00; 0,00)

Tabla 1: Soluciones obtenidas por el método de Newton

El mismo problema fue resuelto a través de ED a lo largo de 30 ejecuciones; con una población de 10 individuos; escogidos al azar de forma tal que x_1 , x_2 pertenece [−4, 4]; con un tiempo de vida de 5 generaciones y una probabilidad de cruzamiento de 0,1. Algunas de las soluciones obtenidas se expresan en la tabla 2.

Solución	*Fitness*	*Flops*	F (x)
$(-4{,}429 \times 10^{-3};\ 6{,}579 \times 10^{-2})$	$8{,}657 \times 10^{-3}$	2455	$(4{,}357 \times 10^{-3};\ -4{,}299 \times 10^{-3})$

$(-3{,}036 \times 10^{-1}; 3{,}291 \times 10^{-1})$	$2{,}379 \times 10^{-1}$	2433	$(2{,}219 \times 10^{-1}; -1{,}600 \times 10^{-2})$
$(2{,}519 \times 10^{-1}; 2{,}243 \times 10^{-1})$	$1{,}337 \times 10^{-1}$	2441	$(1{,}205 \times 10^{-1}; 1{,}323 \times 10^{-2})$
$(-2{,}044 \times 10^{-1}; 5{,}219 \times 10^{-1})$	$5{,}750 \times 10^{-1}$	2445	$(3{,}691 \times 10^{-1}; -2{,}059 \times 10^{-1})$

Tabla 2: Soluciones obtenidas por medio de ED

Los estadísticos obtenidos en las ejecuciones del algoritmo de ED son:

Mediana: $2{,}379 \times 10^{-1}$
Promedio: $4{,}125 \times 10^{-1}$
Desviación Estándar: $4{,}844 \times 10^{-1}$
Máximo fitness: 2,505
Mínimo fitness: $8{,}657 \times 10^{-3}$
Flops *promedio:* 2443

Ahora, si se utilizan puntos obtenidos como salida del ED, mostrados en la tabla 2 como semillas para el método de Newton, con los mismos parámetros anteriores, se obtienen 4 soluciones refinadas. Dichas soluciones se muestran en la tabla 3.

Solución refinada	*Flops*	Iter.	Σ *Flops*	F (x)
$(-5{,}366 \times 10^{-9}; 5{,}882 \times 10^{-9})$	3618	25	6073	(0,000; 0,000)
$(-3{,}887 \times 10^{-9}; 6{,}638 \times 10^{-9})$	3908	27	6341	(0,000; 0,000)
$(2{,}150 \times 10^{-9}; 2{,}049 \times 10^{-9})$	3908	27	6349	(0,000; 0,000)
$(7{,}389 \times 10^{-10}; 4{,}110 \times 10^{-9})$	4050	28	6495	(0,000; 0,000)

Tabla 3: Soluciones obtenidas por la combinación de ED y Newton

6.2 EJEMPLO 2

Características aplicadas para el método de Newton en el sistema (16): cuatro vectores alea- torios de forma tal que x_1 , x_2 pertenece [−5, 5]; un máximo de 50 ciclos e iterando hasta que el error relativo entre x_{n-1} y x_n sea menor a 0,001. Los resultados se resumen en la tabla 4.

Semilla	Solución	Iter.	*Flops*	F (x)
(−0,7003; 2,4511)	(1,0331; −0,2800)	23	3438	$(4{,}91; 5{,}73) \times 10^{-4}$

(1,8813; 4,0493)	(1,0331; −0,2800)	41	6098	$(4{,}91; 5{,}73) \times 10^{-4}$
(4,0375; −4,5450)	(−0,3737; 0,0563)	27	4024	$(0{,}49; -2{,}18) \times 10^{-4}$
(−3,0450; −4,1493)	Matriz singular	-	-	-

Tabla 4: Soluciones obtenidas a través del método de Newton

A fin de buscar soluciones de (16) para utilizar como semillas del método de Newton, se ejecuta ED 30 veces; con una población de 10 individuos escogidos al azar de forma tal que x_1 , x_2 pertenece [−5, 5]; con un tiempo de vida de 5 generaciones y una probabilidad de cruzamiento de 0,1. Algunas de esas soluciones se muestran en la tabla 5.

Solución	*Fitness*	*Flops*	F (x)
$(-3{,}827 \times 10^{-1}; 8{,}709 \times 10^{-2})$	$2{,}875 \times 10^{-1}$	3744	$(8{,}921 \times 10^{-2}; -1{,}982 \times 10^{-1})$
$(1{,}010; -3{,}319 \times 10^{-1})$	$6{,}778 \times 10^{-1}$	3762	$(1{,}433 \times 10^{-1}; 5{,}314 \times 10^{-1})$
$(3{,}596 \times 10^{-1}; -2{,}600 \times 10^{-1})$	2,594	3761	$(-1{,}883; -7{,}622 \times 10^{-1})$
$(1{,}007; -2{,}107 \times 10^{-1})$	1,638	3773	$(-5{,}969 \times 10^{-1}; -1{,}043)$

Tabla 5: Soluciones obtenidas a través de ED

Los estadísticos obtenidos en las ejecuciones del algoritmo de ED son:

Mediana: 1,638
Promedio: 2,521
Desviación Estándar: 2,758
Máximo fitness: 10,87
Mínimo fitness: $2{,}875 \times 10^{-1}$
Flops *promedio:* 3754

Ahora, si se utilizan los puntos mostrados en la tabla 5 como semillas para el método de Newton, con los mismos parámetros anteriores, se obtienen las soluciones refinadas que se muestran en la tabla 6.

Solución refinada	*Flops*	Iter.	Σ *Flops*	F (x)
$(-3{,}736 \times 10^{-1}; 5{,}626 \times 10^{-2})$	464	3	4208	$(-4{,}563 \times 10^{-4}; 1{,}121 \times 10^{-5})$
$(1{,}033; -2{,}799 \times 10^{-1})$	612	4	4374	$(-1{,}026 \times 10^{-3}; -9{,}811 \times 10^{-4})$
$(1{,}478 \times 10^{-1}; -4{,}362 \times 10^{-1})$	760	5	4521	$(3{,}285 \times 10^{-4}; 2{,}036 \times 10^{-4})$

$(1{,}033; -2{,}799 \times 10^{-1})$	464	3	4237	$(-1{,}026 \times 10^{-3}; -9{,}811 \times 10^{-4})$

Tabla 6: Soluciones obtenidas combinando ED y Newton

6.3 EJEMPLO 3

Resolviendo (18) a través del método de Newton; se tomaron cuatro vectores aleatorios de forma tal que x_1 , x2 pertenece [−10, 10]; con un máximo de 50 ciclos y se iteró hasta que el error relativo entre x_{n-1} y x_n sea menor a 0,001. Los resultados se resumen en la tabla 7

Semilla	Solución	Iter.	*Flops*	F (x)
(5,1021; 3,1630)	(28,0999; 26,9591)	40	5838	(5,89; 2,21)
(−1,5011; 1,6265)	Matriz singular	-	-	-
(0,6994; 2,5456)	(0,3877; 4,1139)	7	1060	$(0{,}05; -76{,}1) \times 10^{-6}$
(1,1803; −0,5607)	Matriz singular	-	-	-

Tabla 7: Soluciones obtenidas por el método de Newton

El mismo problema fue resuelto a través de 30 ejecuciones de ED; con una población de 10 individuos; escogidos al azar de forma tal que x_1, $x_2 \in [-10, 10]$; con un tiempo de vida de 5 generaciones y una probabilidad de cruzamiento de 0,1. Algunas de las soluciones obtenidas se muestran en la tabla 8.

Solución	*Fitness*	*Flops*	F (x)
(−1,170; 4,014)	$3{,}267 \times 10^{-2}$	2862	$(2{,}337 \times 10^{-2}; -1{,}040 \times 10^{-2})$
$(-4{,}594; 9{,}965 \times 10^{-1})$	$3{,}973 \times 10^{-1}$	2892	$(2{,}666 \times 10^{-1}; -1{,}303 \times 10^{-1})$
(−2,999; −2,730)	$4{,}528 \times 10^{-1}$	2877	$(1{,}729 \times 10^{-2}; 4{,}370 \times 10^{-1})$
(−2,328; −1,318)	$6{,}899 \times 10^{-1}$	2865	$(5{,}695 \times 10^{-2}; -6{,}331 \times 10^{-1})$

Tabla 8: Soluciones obtenidas según ED

Los estadísticos obtenidos en las ejecuciones del algoritmo de ED son:

Mediana: $6{,}943 \times 10^{-1}$

Promedio: $7{,}682 \times 10^{-1}$

Desviación Estándar: $4{,}159 \times 10^{-1}$

Máximo fitness: 1,502

Mínimo fitness: $3{,}267 \times 10^{-2}$

Flops *promedio:* 2887

Ahora, si se utilizan las soluciones mostradas en la tabla 8 como semillas para el método de Newton, con los mismos parámetros anteriores, se obtienen soluciones refinadas, mostradas en la tabla 9.

Solución refinada	*Flops*	Iter.	Σ *Flops*	F (x)
(−1,189; 3,958)	316	2	3178	$(2{,}098 \times 10^{-5}; -5{,}101 \times 10^{-4})$
(−3,958; 1,189)	614	4	3506	$(2{,}098 \times 10^{-5}; -5{,}101 \times 10^{-4})$
Matriz singular	-	-	-	-
(−1,772; −1,772)	746	5	3611	$(-2{,}121 \times 10^{-3}; 1{,}294 \times 10^{-6}) \times 10^{-3}$

Tabla 9: Soluciones obtenidas combinando ED y Newton

6.4 EJEMPLO 4

Resolviendo el sistema (20) a través del método de Newton; se tomaron cuatro vectores aleatorios de forma tal que x_1, x_2 pertenece [−5, 5]; con un máximo de 50 ciclos y se iteró hasta que el error relativo entre x_{n-1} y x_n sea menor a 0,001. Los resultados se expresan en la tabla 10.

Semilla	Solución	Iter.	*Flops*	F (x)
(−1,1240; 3,5948)	Matriz singular	-	-	-
(0,9866; −2,8456)	(−6,3987; 7,4274)	38	4538	$(-28{,}0; 2{,}2 \times 10^{-3})$
(−4,7594; −1,4265)	Matriz singular	-	-	-
(0,4405; −0,9405)	(−17,4716; 11,4487)	33	3936	$(-10{,}4; -2{,}6 \times 10^{-3})$

Tabla 10: Soluciones obtenidas por el método de Newton

El mismo problema fue resuelto a través de 30 ejecuciones de ED; con una población de 10 individuos; escogidos al azar de forma tal que x_1 , x_2 pertenece [−4, 4]; con un tiempo de vida de 10 generaciones y una probabilidad de cruzamiento de 0,1. En la tabla 11 se muestran algunas soluciones.

Solución	*Fitness*	*Flops*	F (x)

$(2{,}498 \times 10^{-1}; 1{,}666)$	1,188	4186	$(-1{,}329 \times 10^{-2}; 1{,}175)$
$(2{,}573 \times 10^{-1}; 1{,}449)$	1,432	4161	$(-4{,}563 \times 10^{-1}; 9{,}768 \times 10^{-1})$
$(4{,}681 \times 10^{-1}; 1{,}226)$	1,459	4198	$(-2{,}014 \times 10^{-1}; 1{,}258)$
$(2{,}686 \times 10^{-1}; 1{,}354)$	1,528	4203	$(-6{,}071 \times 10^{-1}; 9{,}213 \times 10^{-1})$

Tabla 11: Soluciones obtenidas con ED

Los estadísticos obtenidos en las ejecuciones del algoritmo de ED son:

Mediana: 1,637
Promedio: 1,727
Desviación Estándar: $4{,}352 \times 10^{-1}$
Máximo fitness: 2,754
Mínimo fitness: 1,188
Flops *promedio:* 4205

Ahora, si se utilizan las soluciones mostradas en la tabla 11 como semillas para el método de Newton, con los mismos parámetros anteriores, se obtienen soluciones refinadas, que son mostradas en la tabla 12.

Solución refinada	*Flops*	Iter.	Σ *Flops*	F (x)
(1,837; 2,459)	1213	10	5011	$(2{,}001 \times 10^{-2}; -2{,}172 \times 10^{-3})$
(−6,399; 7,425)	850	7	5011	$(-2{,}226 \times 10^{-1}; 2{,}749 \times 10^{-4})$
Matriz singular	-	-	-	-
(1,837; 2,459)	1802	15	6005	$(2{,}001 \times 10^{-2}; -2{,}172 \times 10^{-3})$

Tabla 12: Soluciones obtenidas por ED + Newton

7 CONCLUSIONES

Se presentó la técnica metaheurística Evolución Diferencial, y se utilizé como complemento del Método de Newton para sistemas de ecuaciones no lineales. Si bien las soluciones obtenidas por ED, resolviendo con un gran espacio de búsqueda y con varias generaciones de vida, no son de tan buena calidad como las obtenidas por Newton, se remarca que la combinación ED + Newton obtiene soluciones de la misma calidad que Newton e incluso evita los problemas de matrices mal condicionadas. Además, ED logra identificar semillas de buena calidad para iniciar el método de Newton. La cantidad de operaciones flotantes involucradas en cada ejemplo muestran que la combinación ED + Newton es factible de ser usada cuando se desconozca una semilla buena para iniciar el método de Newton, sin afectar el proceso global de convergencia.

8 REFERENCIAS

Brindle A. 1981. *Genetic Algorithms for Function Optimization*. , Departament of Computer Science, University of Alberta.

Burden R.L. Faires J.D. 2000 *Numerical Analysis, (7th Ed)*. Thomson Learning.

DeJong K.A. 2006. *Evolutionary Computation - A Unified Approach*. MIT Press.

Fogel D.B. 2000. *Evolutionary Computation 1: Basic Algorithms and Operators*. Taylor & Francis Group,

Goldberg D.E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley.

Quarteroni A., Sacco R., Saleri F. 2000 *Numerical Mathematics*. Springer.

Storn R. 1995. On the usage of differential evolution for function optimization. *NAFIPS*, I:519 – 523,

Storn R. Price K. 1995.Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. , ICSI.

Tjalling J.Y. 1995. Historical development of the newton-raphson method. *SIAM Review*, 37:531–551.

Yang W.Y., Cao W., Chung T., Morris J. 2004.*Applied Numerical Methods Using MatLab*. Wiley-Interscience.

Цzban A. 2004. Some new variants of newton's method. *Applied Mathematics Letters*, 17:677–682,

A CODIGO EN MATLAB

```
function sol = ED(f, pop, datos, t, Cr, Es)
% f: función a minimizar
% pop: cantidad de individuos a utilizar
% datos: rangos de los parámetros de la función
% t: tiempo de vida de la población
% Cr: probabilidad de crossover
% Es: factor de escala a utilizar en la mutación

G = generacion(pop, datos);
G = fitness(f, G);
for i=1:t
    G = evolucion(f, G, Cr, Es, datos);
end
```

```
[a, b] = size(G);
optimo = min(G(:,b));
for i=1:a
    if optimo == G(i,b)
        break
    end
end
sol = G(i,:);
```

```
function [G] = generacion(n, inicial)
% n: cantidad de individuos a utilizar
% inicial: rangos de los n parámetros de la función

[a, b] = size(inicial);
rand('state',sum(100*clock));
for i=1:n
    for j=1:a
        G(i,j) = inicial(j,1) + rand*(inicial(j,2) - inicial(j,1));
        G(i,j+1) = 0;
    end
end
```

```
function G = fitness(f, G)
% f: función a minimizar
% G: matriz de datos de la función

[a, b] = size(G);
for i=1:a;
    for j=1:b-1;
        x(j) = G(i,j);
    end
    G(i, b) = eval(f);
End
```

```
function G = evolucion(f, G, Cr, Es, datos)
% f: función a minimizar
% G: matriz de datos de la función
% Cr: probabilidad de crossover
% Es: factor de escala a utilizar en la mutación
% datos: rangos de los parámetros de la función

G2 = G;
[a, b] = size(G);
for i=1:a
    % Inicio Mutacion
    v = randperm(a);
    while v(1) == i | v(2) == i;
       v = randperm(a);
    end
    G(i,1:b-1) = G2(v(3),1:b-1) + Es*(G2(v(1),1:b-1) - G2(v(2),1:b-1));
    for j=1:b-1
```

```
        if G(i,j)<datos(j,1) | G(i,j)>datos(j,2)
           G(i,1:b-1) = G2(i,1:b-1);
           break
        end
    end
    % Fin Mutación

    % Inicio Crossover
    if rand < Cr
        v = randperm(b-2);
        G(i, v(1):b-1) = G2(i, v(1):b-1);
    end
    % Fin Crossover

    % Inicio Selección
    for j=1:b-1;
        x(j) = G(i,j);
    end
    test = eval(f);
    if test > G2(i,b)
        G(i,:) = G2(i,:);
    end
    % Fin Selección
end
G = fitness(f, G);
```