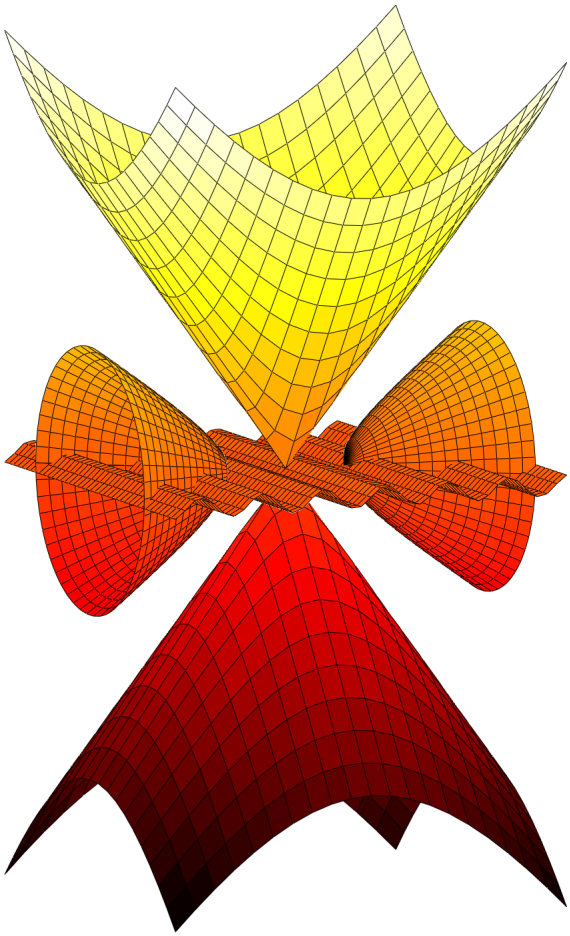
Universidad de El Salvador

Facultad de Ciencias Naturales y Matemática

Escuela de Matemática

Ciclo I - 2021



Solución numérica de sistemas de ecuaciones no lineales

Materia:
Análisis Numérico I

Profesor:
Maestro Porfirio Armando Rodríguez

Estudiante:
Leonel Antonio Prudencio Castro PC11012

Fecha:
1 de julio de 2021

1 Índice

2 Introducción

Resolver una ecuación de tipo $f(x) = 0$ no siempre es fácil, es más, si se tratara de un sistema de ecuaciones de la forma:

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) = 0 \\ f_2(x_1, x_2, \ldots, x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) = 0 \end{cases}$$

, el problema se complica mucho más. Sin embargo hay métodos numéricos que posibilitan encontrar una aproximación aceptable de la solución de dichos sistemas, en este material se busca exponer métodos numéricos que logren conseguir soluciones para *sistemas de ecuaciones no lineales (SENL)*, en algunos casos estos métodos son análogos a los utilizados en el caso lineal como lo son los métodos de *puto fijo* y de *Newton*, además de un método *cuasi-Newton* como lo es el método de *Broyden* que propone una mejora en el método de Newton relajando algunos cálculos del proceso en el algoritmo de Newton, se expone también del método de *descenso más rápido* que aunque no es tan exacto, puede dar resultados que sirvan como aproximación inicial para otros métodos y para finalizar, se introduce el método de *continuación* que utiliza un parámetro auxiliar para alcanzar el valor deseado de la aproximación.

Para cada uno de estos métodos se propone un código correspondiente en lenguaje Octave/Matlab y al final de este material se resuelve un problema de aplicación por medio de la implementación de uno de estos métodos, se omiten las demostraciones de los teoremas enunciados pero se hace énfasis en la motivación del algoritmo asociado a cada método.

3 Preliminares

Un sistema de ecuaciones no lineales con n variables y n ecuaciones tiene esta forma:

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) = 0 \\ f_2(x_1, x_2, \ldots, x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) = 0 \end{cases}$$

; donde cada función f_i, para cada $i = 1, 2, \ldots, n$ está definida de esta manera:

$$\begin{array}{rcl} f_i : \mathbb{R}^n & \longrightarrow & \mathbb{R} \\ (x_1, x_2, \ldots, x_n)^t & \longmapsto & f_i(x_1, x_2, \ldots, x_n) \end{array}$$

; donde alguna función es no lineal respecto de alguna variable x_i, cada función f_i se puede llamar *campo escalar* o *función escalar*.

Luego se puede representar el sistema, definiendo una función $\mathbf{F}$ de esta manera:

$$\begin{array}{rcl} \mathbf{F} : \mathbb{R}^n & \longrightarrow & \mathbb{R}^n \\ (x_1, x_2, \ldots, x_n)^t & \longmapsto & (f_1(x_1, x_2, \ldots, x_n), f_2(x_1, x_2, \ldots, x_n), \ldots, f_n(x_1, x_2, \ldots, x_n))^t \end{array}$$

Bajo ciertas condiciones esta función se llama *campo vectorial*, pero para propósitos del tema se hace énfasis en el uso de negrita para denotar este tipo de funciones, así a menos que se indique lo contrario, estas notaciones por ejemplo: $\mathbf{F}$, $\mathbf{G}$ y $\mathbf{H}$ denotan funciones con las características anteriores.

Para facilitar cálculos se introduce la notación $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$ para denotar el vector cuyas componentes son $x_1, x_2, \ldots, x_n$, notar que se usa negrita, así por ejemplo $\mathbf{y}, \mathbf{s}$ y $\mathbf{p}$ son vectores en $\mathbb{R}^n$.

Para "medir" la distancia entre elementos (vectores) de $\mathbb{R}^n$ se introduce lo siguiente:

• **Definición 1:** La *norma infinito* de $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$ se denota $||\mathbf{x}||_\infty$ y se define así:

$$\|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$$

Luego la distancia entre dos vectores, $\mathbf{x}$ y $\mathbf{y}$ por ejemplo se define como la *norma infinito* de su diferencia así: $||\mathbf{x} - \mathbf{y}||_\infty$, esto será muy útil en el momento de comparar las aproximaciones que se vayan generando.

4 Métodos numéricos para la solución de sistemas de ecuaciones no lineales

4.1 Método de punto fijo

• **Definición 2:** Una función $\mathbf{G} : D \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}^n$ tiene un *punto fijo*: $\mathbf{p} \in D$ si $\mathbf{G}(D) = D$

• **Teorema 3:** Sea $D = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n]$ para conjunto de constantes $a_1, a_2, \ldots, a_n$ y $b_1, b_2, \ldots, b_n$. Sea $\mathbf{G} : D \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}^n$ un función continua con la propiedad de que $\mathbf{G}(\mathbf{x}) \in D$ siempre que $\mathbf{x} \in D$ ($\mathbf{G}$ tiene un punto fijo en D). Supongamos que las funciones componentes de $\mathbf{G}$ tienen todas sus derivadas parciales continuas y que existe una constante $K < 1$ tal que:

$$\left| \frac{\partial g_i(\mathbf{x})}{\partial x_j} \right| \leq \frac{K}{n} \text{ ,siempre que } \mathbf{x} \in D$$

; esto es para toda $j = 1, 2, \ldots, n$ y toda función componente g_i con $i = 1, 2, \ldots, n$.

Entonces la sucesión $\left\{\mathbf{x}^{(k)}\right\}_{k=0}^{\infty}$ definida desde alguna $\mathbf{x}^{(0)} \in D$ seleccionada arbitrariamente y generada por $\mathbf{x}^{(k)} = \mathbf{G}\left(\mathbf{x}^{(k-1)}\right)$, para cada $k \geq 1$ converge al punto fijo: $\mathbf{p} \in D$ de $\mathbf{G}$ y es único, además:

$$\left\|\mathbf{x}^{(k)} - \mathbf{p}\right\|_\infty \leq \frac{K^k}{1-K} \left\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\right\|_\infty$$

4.1.1 Algoritmo

$$\begin{cases} \mathbf{x}^{(0)} \in D \subset \mathbb{R}^n \\ \mathbf{x}^{(k)} = \mathbf{G}\left(\mathbf{x}^{(k-1)}\right), k \geq 1 \end{cases}$$

; donde $\mathbf{G}$ cumple las condiciones del *teorema 3*.

4.1.2 Ejemplo

Demostrar que el sistema :

$$\begin{cases} x_1^2 - 10x_1 + x_2^2 + 8 & = 0 \\ x_1^2 x_2^2 + x_1 - 10x_2 + 8 & = 0 \end{cases}$$

, tiene solución única en $D = \left\{(x_1, x_2)^t \,|0 \leq x_1, x_2 \leq 1.5\right\}$ y se calcular dicha solución con con aproximación inicial $\mathbf{x}^{(0)} = (0, 0)^t$ y tolerancia de 10^{-5}.

◦ *Solución:*

Se plantea el problema como una búsqueda de punto fijo, definiendo una función $\mathbf{G} : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ donde sus funciones componentes $g_1 : \mathbb{R}^2 \longrightarrow \mathbb{R}$ y $g_2 : \mathbb{R}^2 \longrightarrow \mathbb{R}$ se obtienen despejando las ecuaciones del sistema así:

$$\begin{aligned} g_1(x_1, x_2) &= x_1 = \frac{x_1^2 + x_2^2 + 8}{10} \\ g_2(x_1, x_2) &= x_2 = \frac{x_1 x_2^2 + x_1 + 8}{10} \end{aligned}$$

Notar que $\mathbf{G}(D) \subset D$ porque $\frac{8}{10} \leq \frac{x_1^2+x_2^2+8}{10} \leq 1.25$ y también $\frac{8}{10} \leq \frac{x_1x_2^2+x_1+8}{10} \leq 1.2875$
Obteniendo las derivadas parciales de $\mathbf{G}$, se tiene que:

$$\left|\frac{\partial g_1(x)}{\partial x_1}\right| = \left|\frac{\partial g_2(x)}{\partial x_2}\right| = 0$$

$$\left|\frac{\partial g_1(x)}{\partial x_2}\right| = \frac{x_2}{5} \leq \frac{1.5}{5} = 0.3 \quad \text{y} \quad \left|\frac{\partial g_2(x)}{\partial x_1}\right| = \frac{x_1}{5} \leq \frac{1.5}{5} = 0.3$$

$$\Rightarrow \left|\frac{\partial g_i(x)}{\partial x_j}\right| \leq 0.3 \quad ; \text{para toda} \quad i,j = 1,2,\ldots,n \text{ y toda } x \in D$$

Haciendo $\frac{K}{2} = 0.3$, se tiene que $K = 0.6 < 1$, se nota además que cada derivada parcial es continua en D y aplicando el teorema se tiene que $\mathbf{G}$ posee un único punto fijo en D, luego aplicando el algoritmo para $\mathbf{x}^{(0)} = (0,0)^t$, se tiene para $k \geq 1$: $\mathbf{x}^{(k)} = \left(x_1^{(k)}, x_2^{(k)}\right) = \mathbf{G}(\mathbf{x}^{(k-1)})$, donde los valores de $x_1^{(k)}$ y $x_2^{(k)}$ vienen dados así:

$$\begin{aligned} x_1^{(k)} &= \frac{\left(x_1^{(k-1)}\right)^2 + \left(x_2^{(k-1)}\right)^2 + 8}{10} \\ x_2^{(k)} &= \frac{x_1^{(k-1)}\left(x_2^{(k-1)}\right)^2 + x_1^{(k-1)} + 8}{10} \end{aligned}$$

▷ *Primera iteración:*

$\mathbf{x}^{(0)} = (0,0)^t \Rightarrow \mathbf{x}^{(1)} = \mathbf{G}\left(\mathbf{x}^{(0)}\right) = (0.8, 0.8)^t$

▷ *Segunda iteración:*

$\mathbf{x}^{(1)} = (0.8,0.8)^t \Rightarrow \mathbf{x}^{(2)} = \mathbf{G}\left(\mathbf{x}^{(1)}\right) = (0.928, 0.9312)^t$

▷ *Iteraciones siguientes:*

Tabla 1: Iteraciones de método de *punto fijo*

k	$\mathbf{x}^{(k-1)}$	$\mathbf{x}^{(k)}$	$\left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right\|_\infty$
1	$\begin{pmatrix} 0.0000000 \\ 0.0000000 \end{pmatrix}$	$\begin{pmatrix} 0.80000000 \\ 0.80000000 \end{pmatrix}$	0.8000000
2	$\begin{pmatrix} 0.8000000 \\ 0.8000000 \end{pmatrix}$	$\begin{pmatrix} 0.9280000 \\ 0.9312000 \end{pmatrix}$	0.1312000
3	$\begin{pmatrix} 0.9280000 \\ 0.9312000 \end{pmatrix}$	$\begin{pmatrix} 0.9728317 \\ 0.9732700 \end{pmatrix}$	0.0448317
4	$\begin{pmatrix} 0.9728317 \\ 0.9732700 \end{pmatrix}$	$\begin{pmatrix} 0.9893656 \\ 0.9894351 \end{pmatrix}$	0.0165339
5	$\begin{pmatrix} 0.9893656 \\ 0.9894351 \end{pmatrix}$	$\begin{pmatrix} 0.9957826 \\ 0.9957937 \end{pmatrix}$	0.0064170
6	$\begin{pmatrix} 0.9957826 \\ 0.9957937 \end{pmatrix}$	$\begin{pmatrix} 0.9983188 \\ 0.9983206 \end{pmatrix}$	0.0025362

7	$\begin{pmatrix} 0.9983188 \\ 0.9983206 \end{pmatrix}$	$\begin{pmatrix} 0.9993284 \\ 0.9993287 \end{pmatrix}$	0.0010096
8	$\begin{pmatrix} 0.9993284 \\ 0.9993287 \end{pmatrix}$	$\begin{pmatrix} 0.9997315 \\ 0.9997316 \end{pmatrix}$	0.0004031
9	$\begin{pmatrix} 0.9997315 \\ 0.9997316 \end{pmatrix}$	$\begin{pmatrix} 0.9998926 \\ 0.9998926 \end{pmatrix}$	0.0001611
10	$\begin{pmatrix} 0.9998926 \\ 0.9998926 \end{pmatrix}$	$\begin{pmatrix} 0.9999571 \\ 0.9999571 \end{pmatrix}$	0.0000644
11	$\begin{pmatrix} 0.9999571 \\ 0.9999571 \end{pmatrix}$	$\begin{pmatrix} 0.9999828 \\ 0.9999828 \end{pmatrix}$	0.0000258
12	$\begin{pmatrix} 0.9999828 \\ 0.9999828 \end{pmatrix}$	$\begin{pmatrix} 0.9999931 \\ 0.9999931 \end{pmatrix}$	0.0000103
13	$\begin{pmatrix} 0.9999931 \\ 0.9999931 \end{pmatrix}$	$\begin{pmatrix} 0.9999973 \\ 0.9999973 \end{pmatrix}$	0.0000041

Se considera la aproximación: $\mathbf{x}^{(13)} = (0.9999973, 0.9999973)^t$ con tolerancia de 10^{-5}, el valor real del punto fijo es $\mathbf{x} = (1, 1)^t$.

4.1.3 Código

Código 1: Implementación del método de *punto fijo*

```
function [x i]= PuntoFijo_SENL(Fsis, x0, TOL, n=35)
%INPUT:
%Fsis: función de R^n a R^n asociada al sistema
%x0: aproximación inicial
%TOL: tolerancia permitida
%n: numero máximo de iteraciones (35 por defecto)
%OUTPUT:
%x: aproximación final
%i: indice de la ultima iteración
fprintf('\t\t>> Método de Punto Fijo');
fprintf('k\tx^(k-1)\t\tx^(k)\t\t||x^(k)-x^(k-1)||oo\n');
fprintf('_____________________________________________________________\n');
for i = 1:n
  x = Fsis(x0);

  err = max(abs(x-x0));%para comparar con la tolerancia (es la norma infinito)

  fprintf('%d\t%10.7f\t%10.7f\t%10.7f\n',i,x0(1),x(1),err);
  for j = 2:length(x)
    fprintf('\t%10.7f\t%10.7f\t\n',x0(j),x(j));
  endfor
  fprintf('_____________________________________________________________\n');

  if err < TOL %se comprueba si ya se satisface la tolerancia
     return
  else
```

```
      x0 = x; % si no se satisface, se realiza el mismo proceso
                %refrescando el valor de x0
  endif
endfor
%se llega al máximo de iteraciones
disp("Numero máximo de iteraciones alcanzado")
return
endfunction
```

4.2 Método de Newton

El algoritmo del método de Newton para una sola ecuación no lineal viene dado por:

$$x^{(k)} = x^{(k-1)} - \frac{f\left(x^{(k-1)}\right)}{f'\left(x^{(k-1)}\right)}$$

Para asociar este resultado a funciones de $\mathbb{R}^n$ se se introduce la siguiente definición:

• **Definición 4:** Sea $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ una función tal que sus derivadas parciales de primer orden existen en $\mathbb{R}^n$. La *matriz Jacobiana* de $\mathbf{F}$ denotada por $J_{\mathbf{F}}$ está definida como una matriz de tamaño $n \times n$ que tiene esta forma:

$$J_{\mathbf{F}}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \frac{\partial f_1}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}) \\ \frac{\partial f_2}{\partial x_1}(\mathbf{x}) & \frac{\partial f_2}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial f_2}{\partial x_n}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\mathbf{x}) & \frac{\partial f_n}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial f_n}{\partial x_n}(\mathbf{x}) \end{bmatrix}$$

Ahora en $\mathbb{R}^n$ se puede introducir la siguiente fórmula para el cálculo de las aproximaciones:

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - \left[J_{\mathbf{F}}\left(\mathbf{x}^{(k-1)}\right)\right]^{-1} \mathbf{F}\left(\mathbf{x}^{(k-1)}\right)$$

, donde la ecuación matricial anterior tendrá solución siempre que el $det\left(J_{\mathbf{F}}\left(\mathbf{x}^{(k-1)}\right)\right)$ sea no nulo para cada $k \geq 0$, notar que para cada iteración es necesario calcular $\left[J_{\mathbf{F}}\left(\mathbf{x}^{(k-1)}\right)\right]^{-1}$.

4.2.1 Algoritmo

$$\begin{cases} \mathbf{x}^{(0)} \in \mathbb{R}^n \\ \mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - \left[J_{\mathbf{F}}\left(\mathbf{x}^{(k-1)}\right)\right]^{-1} \mathbf{F}\left(\mathbf{x}^{(k-1)}\right), k \geq 1 \end{cases}$$

; donde $J_{\mathbf{F}}$ es la *matriz Jacobiana* de $\mathbf{F}$ y la ecuación tendrá solución siempre que $det\left(J_{\mathbf{F}}\left(\mathbf{x}^{(k-1)}\right)\right)$ sea no nulo.

4.2.2 Ejemplo

Resolver el siguiente sistema con aproximación inicial $\mathbf{x}^{(0)} = (0, 0, 0)^t$ y tolerancia de 10^{-6}:

$$\begin{cases} 6x_1 - 2cos\left(x_2 x_3\right) - 1 & = 0 \\ 9x_2 + \sqrt{x_1^2 + sen\left(x_3\right) + 1.06} + 0.9 & = 0 \\ 60x_3 + 3e^{-x_1 x_2} + 10\pi - 3 & = 0 \end{cases}$$

◦ *Solución:*

$$\mathbf{F}(\mathbf{x}) = \Big(6x_1 - 2cos\,(x_2x_3) - 1,9x_2 + \sqrt{x_1^2 + sen\,(x_3) + 1.06} + 0.9,$$
$$60x_3 + 3e^{-x_1x_2} + 10\pi - 3\Big)$$

$$J_{\mathbf{F}}(\mathbf{x}) = \begin{bmatrix} 6 & 2x_3sen\,(x_2x_3) & 2x_2sen\,(x_2x_3) \\ \frac{x_1}{\sqrt{x_1^2+sen(x_3)+1.06}} & 9 & \frac{0.5cos(x_3)}{\sqrt{x_1^2+sen(x_3)+1.06}} \\ -3e^{-x_1x_2}x_2 & -3e^{-x_1x_2}x_1 & 60 \end{bmatrix}$$

▷ *Primera iteración:*

$$\mathbf{F}\left(\mathbf{x}^{(0)}\right) = (-3.0000, 1.9296, 31.4159)^t$$

$$J_{\mathbf{F}}\left(\mathbf{x}^{(0)}\right) = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 9 & 0.4856 \\ 0 & 0 & 60 \end{bmatrix} \Rightarrow \left[J_{\mathbf{F}}\left(\mathbf{x}^{(0)}\right)\right]^{-1} = \begin{bmatrix} 0.1667 & 0 & 0 \\ 0 & 0.1111 & -0.0009 \\ 0 & 0 & 0.0167 \end{bmatrix}$$

$$\Rightarrow \mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \left[J_{\mathbf{F}}\left(\mathbf{x}^{(0)}\right)\right]^{-1}\mathbf{F}\left(\mathbf{x}^{(0)}\right) = (0.5000000, -0.1861424, -0.5235988)^t$$

▷ *Segunda iteración:*

$$\mathbf{F}\left(\mathbf{x}^{(1)}\right) = (0.0094917, 0.12472, 0.29262)^t$$

$$J_{\mathbf{F}}\left(\mathbf{x}^{(1)}\right) = \begin{bmatrix} 6 & -0.1019 & 0.036227 \\ 0.55556 & 9 & 0.48113 \\ 0.6129 & -1.6463 & 60 \end{bmatrix}$$

$$\Rightarrow \left[J_{\mathbf{F}}\left(\mathbf{x}^{(1)}\right)\right]^{-1} = \begin{bmatrix} 0.16648 & 0.0019006 & 0.000085278 \\ -0.010171 & 0.11083 & -0.00089488 \\ -0.0019797 & 0.0030217 & 0.016641 \end{bmatrix}$$

$$\Rightarrow \mathbf{x}^{(2)} = \mathbf{x}^{(1)} - \left[J_{\mathbf{F}}\left(\mathbf{x}^{(1)}\right)\right]^{-1}\mathbf{F}\left(\mathbf{x}^{(1)}\right) = (0.3306729, -0.2434028, 0.5194079)^t$$

▷ *Iteraciones siguientes:*

Tabla 2: Iteraciones de método de *Newton*

k	$\mathbf{x}^{(k-1)}$	$\mathbf{x}^{(k)}$	$\left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right\|_\infty$
1	$\begin{pmatrix} 0.0000000 \\ 0.0000000 \\ 0.0000000 \end{pmatrix}$	$\begin{pmatrix} 0.5000000 \\ -0.1861424 \\ -0.5235988 \end{pmatrix}$	0.5235988
2	$\begin{pmatrix} 0.3333333 \\ -0.2426495 \\ 0.5235988 \end{pmatrix}$	$\begin{pmatrix} 0.4981578 \\ -0.1996068 \\ -0.5288264 \end{pmatrix}$	0.0134645
3	$\begin{pmatrix} 0.3306729 \\ -0.2434028 \\ 0.5194079 \end{pmatrix}$	$\begin{pmatrix} 0.4981447 \\ -0.1996059 \\ -0.5288260 \end{pmatrix}$	0.0000131
4	$\begin{pmatrix} 0.4981447 \\ -0.1996059 \\ -0.5288260 \end{pmatrix}$	$\begin{pmatrix} 0.4981447 \\ -0.1996059 \\ -0.5288260 \end{pmatrix}$	0.0000000

Se considera la aproximación: $\mathbf{x}^{(4)} = (0.4981447, -0.1996059, -0.5288260)^t$ con tolerancia de 10^{-6}.

4.2.3 Código

Código 2: Implementación del método de *Newton*

```
function [x i]= Newton_SENL(Fsis, x0, TOL, n=35)
%INPUT:
%Fsis: función de ℝ^n a ℝ^n asociada al sistema
%x0: aproximación inicial
%TOL: tolerancia permitida
%n: numero máximo de iteraciones (35 por defecto)
%OUTPUT:
%x: aproximación final
%i: índice de la ultima iteración
fprintf('\t\t>> Método de Newton\n');
fprintf('k\tx^(k-1)\t\tx^(k)\t\t||x^(k)-x^(k-1)||oo\n');
fprintf('_____________________________________________________________\n');
for i = 1:n
    J0 = jacob(Fsis, x0, TOL);%se calcula la matriz jacobiana aplicada a x0
    if det(J0) == 0 %se comprueba que sea invertible
        disp("Utilice otro método")
        return
    endif
    JI = inv(J0);%se calcula la inversa
    Fx = Fsis(x0);%se calcula el valor de F sobre x0
    x = x0 - JI*Fx;%se aplica la fórmula de la aproximación de Newton

    err = max(abs(x-x0));%para comparar con la tolerancia (es la norma infinito)

    %se refresca la tabla
    fprintf('%d\t%10.7f\t%10.7f\t%10.7f\n',i,x0(1),x(1),err);
    for j = 2:length(x)
      fprintf('\t%10.7f\t%10.7f\t\n',x0(j),x(j));
    endfor
    fprintf('_____________________________________________________________\n');

    if err < TOL %se comprueba si ya se satisface la tolerancia
        return
    else
        x0 = x; % si no se satisface, se realiza el mismo proceso
                %refrescando el valor de x0
    endif
endfor
%se llega al máximo de iteraciones
disp("Número máximo de iteraciones alcanzado")
return
```

```
endfunction
```

Código 3: Aproximación de $J_{\mathbf{F}}$ en $\mathbf{x}$

```
function J = jacob(Fsis,x0,h)
%INPUT:
%Fsis: función de ℝ^n a ℝ^n asociada al sistema
%x0: aproximación inicial
%h: tamaño de paso
%OUTPUT:
%J: matriz jacobiana e F en el punto x0
n = length(x0);%dimensión
J = zeros(n,n);%se inicializa la matriz a llenar
y = x0;%para las aproximaciones de las derivadas
Fx = Fsis(x0);%el valor de F en x0
for i = 1:n%recorriendo filas
  fx = Fx(i);%se toma el valor requerido
  for j = 1:n%recorriendo columnas
    y(j) = y(j)+h;%se aumenta el correspondiente
    Fy = Fsis(y);%se calcula el valor de F en x0_j+h
    fy = Fy(i);%se toma el valor requerido
    J(i,j) = (fy-fx)/h;%aproximación de la derivadas
    y = x0;%para comenzar otra vez
  endfor
endfor
endfunction
```

4.3 Método de Broyden

Del método de Newton se tiene: $\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - \left[J_{\mathbf{F}}\left(\mathbf{x}^{(k-1)}\right)\right]^{-1}\mathbf{F}\left(\mathbf{x}^{(k-1)}\right)$.

El método de Broyden aplica el mismo algoritmo lo que lo convierte en un método *cuasi-Newton* que se diferencia del método de Newton al proponer una forma de aproximación para las matrices $\left[J_{\mathbf{F}}\left(\mathbf{x}^{(k-1)}\right)\right]^{-1}$ para cada iteración.

Considerando $f(x) = 0$, un sistema de ecuaciones lineales entonces es de la forma $Ax - b = 0$. Por el método de la secante se sabe que:

$$f\left(x^{(k)}\right) - f\left(x^{(k-1)}\right) = A\left(x^{(k)} - x^{(k-1)}\right)$$

Pero en el caso no lineal la igualdad anterior no se cumple aunque se puede lograr, eligiendo adecuadamente A_k, tal que:

$$\mathbf{F}\left(\mathbf{x}^{(k)}\right) - \mathbf{F}\left(\mathbf{x}^{(k-1)}\right) = A_k\left(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right)$$

Se añade además la condición $A_k(\mathbf{z}) = A_{k-1}(\mathbf{z})$ siempre que $\left(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right)^t\mathbf{z} = 0$, es decir que para los vectores $\mathbf{z}$ ortogonales a $\left(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right)^t$ se tiene que $A_k(\mathbf{z}) = A_{k-1}(\mathbf{z})$ y con esto

cada A_k queda bien definida y se deduce de las condiciones anteriores:

$$\begin{aligned} A_k &= \frac{\mathbf{F}\left(\mathbf{x}^{(k)}\right) - \mathbf{F}\left(\mathbf{x}^{(k-1)}\right)}{\left(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right)} \\ &= A_{k-1} + \frac{\mathbf{F}\left(\mathbf{x}^{(k)}\right) - \mathbf{F}\left(\mathbf{x}^{(k-1)}\right)}{\left(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right)} - A_{k-1} \\ &= A_{k-1} + \frac{\mathbf{F}\left(\mathbf{x}^{(k)}\right) - \mathbf{F}\left(\mathbf{x}^{(k-1)}\right) - A_{k-1}\left(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right)}{\left(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right)} \\ &= A_{k-1} + \frac{\left(\mathbf{F}\left(\mathbf{x}^{(k)}\right) - \mathbf{F}\left(\mathbf{x}^{(k-1)}\right) - A_{k-1}\left(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right)\right)}{\left(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right)\left(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right)^t}\left(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right)^t \end{aligned}$$

Es decir:

$$A_k = A_{k-1} + \frac{(\mathbf{y}_k - A_{k-1}\mathbf{s}_k)}{\|\mathbf{s}_k\|_2^2}\mathbf{s}_k^t$$

; donde $\mathbf{y}_k = \mathbf{F}\left(\mathbf{x}^{(k)}\right) - \mathbf{F}\left(\mathbf{x}^{(k-1)}\right)$ y $\mathbf{s}_k = \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}$

Luego se aplica el método de Newton deduciendo en cada iteración la matriz A_k en sustitución de $J_{\mathbf{F}}$ y calculando su inversa. Para mejorar el método se plantea una forma de generar las matrices inversas directamente.

• **Teorema 5 (Fórmula de Sherman-Morrison):** Si A es una matriz no singular y si $\mathbf{x}$ y $\mathbf{y}$ son vectores tales que $\mathbf{y}^t A\mathbf{x} \neq -1$. Entonces $A + \mathbf{x}\mathbf{y}^t$ es no singular y además:

$$\left(A + \mathbf{x}\mathbf{y}^t\right)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{x}\mathbf{y}^t A^{-1}}{1 + \mathbf{y}^t A^{-1} x}$$

Aplicando la fórmula de Sherman-Morrison para $A = A_{k-1}$, $\mathbf{x} = \frac{(\mathbf{y}_k - A_{k-1}\mathbf{s}_k)}{\|\mathbf{s}_k\|_2^2}$ y $\mathbf{y} = \mathbf{s}_k$, se tiene:

$$A_k^{-1} = A_{k-1}^{-1} + \frac{\left(\mathbf{s}_k - A_{k-1}^{-1}\mathbf{y}_k\right)\mathbf{s}_k^t A_{k-1}^{-1}}{\mathbf{s}_k^t A_{k-1}^{-1}\mathbf{y}_k}$$

4.3.1 Algoritmo

$$\begin{cases} \mathbf{x}^{(0)} \in \mathbb{R}^n \\ \mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - A_k^{-1}\mathbf{F}\left(\mathbf{x}^{(k-1)}\right) \ , \ k \geq 1 \end{cases}$$

; donde $A_0^{-1} = \left[J_{\mathbf{F}}\left(\mathbf{x}^{(0)}\right)\right]^{-1}$, $A_k^{-1} = A_{k-1}^{-1} + \frac{\left(\mathbf{s}_k - A_{k-1}^{-1}\mathbf{y}_k\right)\mathbf{s}_k^t A_{k-1}^{-1}}{\mathbf{s}_k^t A_{k-1}^{-1}\mathbf{y}_k}$

$$, \mathbf{y}_k = \mathbf{F}\left(\mathbf{x}^{(k)}\right) - \mathbf{F}\left(\mathbf{x}^{(k-1)}\right) \text{ y } \mathbf{s}_k = \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}$$

4.3.2 Ejemplo

Resolver el siguiente sistema con aproximación inicial $\mathbf{x}^{(0)} = (2, 2)^t$

$$\begin{cases} ln\left(x_1^2 + x_2^2\right) - sen\left(x_1 x_2\right) & = ln(2) + ln(\pi) \\ e^{x_1 - x_2} + cos\left(x_1 x_2\right) & = 0 \end{cases}$$

◦ *Solución:*

$$\mathbf{F}(\mathbf{x}) = \begin{array}{l}\left(ln\left(x_1^2+x_2^2\right) - sen\left(x_1x_2\right) - ln(2) - ln(\pi),\right. \\ \left. e^{x_1-x_2} + cos\left(x_1x_2\right)\right)\end{array}$$

$$J_{\mathbf{F}}(\mathbf{x}) = \begin{bmatrix} \frac{2x_1}{x_1^2+x_2^2} - cos(x_1x_2)x_2 & \frac{2x_2}{x_1^2+x_2^2} - cos(x_1x_2)x_1 \\ e^{x_1-x_2} - sen\left(x_1x_2\right)x_2 & -e^{x_1-x_2} - sen\left(x_1x_2\right)x_1 \end{bmatrix}$$

▷ *Primera iteración:*

$$\mathbf{F}\left(\mathbf{x}^{(0)}\right) = (0.9984, 0.3464)^t$$

$$A_0 = J_{\mathbf{F}}\left(\mathbf{x}^{(0)}\right) = \begin{bmatrix} 1.8073 & 1.8073 \\ 2.5136 & 0.5136 \end{bmatrix} \Rightarrow A_0^{-1} = \begin{bmatrix} -0.1421 & 0.5 \\ 0.6954 & -0.5 \end{bmatrix}$$

$$\mathbf{s}_0 = A_0^{-1}\mathbf{F}\left(\mathbf{x}^{(0)}\right) = (-0.031211, -0.521105)^t$$

$$\Rightarrow \mathbf{x}^{(1)} = \mathbf{x}^{(0)} - A_0^{-1}\mathbf{F}\left(\mathbf{x}^{(0)}\right) = (1.9686887, 1.4788947)^t$$

▷ *Segunda iteración:*

$$\mathbf{F}\left(\mathbf{x}^{(1)}\right) = (-0.2638, 0.6583)^t$$

$$\mathbf{y}_1 = \mathbf{F}\left(\mathbf{x}^{(1)}\right) - \mathbf{F}\left(\mathbf{x}^{(0)}\right) = (-1.2621, -0.3120)^t$$

$$\mathbf{s}_1 = \mathbf{x}^{(1)} - \mathbf{x}^{(0)} = (-0.031211, -0.521105)^t$$

$$A_1^{-1} = A_0^{-1} + \left(1/\mathbf{s}_1^t A_0^{-1}\mathbf{y}_1\right)\left[\left(\mathbf{s}_1 - A_0^{-1}\mathbf{y}_1\right)\mathbf{s}_1^t A_0^{-1}\right] = \begin{bmatrix} 0.1064 & 0.3300 \\ 0.3480 & -0.2623 \end{bmatrix}$$

$$\Rightarrow \mathbf{x}^{(2)} = \mathbf{x}^{(1)} - A_1^{-1}\mathbf{F}\left(\mathbf{x}^{(1)}\right) = (1.7795004, 1.7433937)^t$$

▷ *Iteraciones siguientes:*

Tabla 3: Iteraciones de método de *Broyden*

k	$\mathbf{x}^{(k-1)}$	$\mathbf{x}^{(k)}$	$\left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right\|_\infty$
1	$\begin{pmatrix} 2.0000000 \\ 2.0000000 \end{pmatrix}$	$\begin{pmatrix} 1.9686887 \\ 1.4788947 \end{pmatrix}$	0.5211053
2	$\begin{pmatrix} 1.9686887 \\ 1.4788947 \end{pmatrix}$	$\begin{pmatrix} 1.7795004 \\ 1.7433937 \end{pmatrix}$	0.2644989
3	$\begin{pmatrix} 1.7795004 \\ 1.7433937 \end{pmatrix}$	$\begin{pmatrix} 1.7719841 \\ 1.7736647 \end{pmatrix}$	0.0302710
4	$\begin{pmatrix} 1.7719841 \\ 1.7736647 \end{pmatrix}$	$\begin{pmatrix} 1.7723637 \\ 1.7725654 \end{pmatrix}$	0.0010994
5	$\begin{pmatrix} 1.7723637 \\ 1.7725654 \end{pmatrix}$	$\begin{pmatrix} 1.7724313 \\ 1.7724832 \end{pmatrix}$	0.0000821
6	$\begin{pmatrix} 1.7724313 \\ 1.7724832 \end{pmatrix}$	$\begin{pmatrix} 1.7724546 \\ 1.7724529 \end{pmatrix}$	0.0000303
7	$\begin{pmatrix} 1.7724546 \\ 1.7724529 \end{pmatrix}$	$\begin{pmatrix} 1.7724539 \\ 1.7724539 \end{pmatrix}$	0.0000010

Se considera la aproximación: $\mathbf{x}^{(7)} = (1.7724539, 1.7724539)^t$ con tolerancia de 10^{-5}.

4.3.3 Código

Código 4: Implementación del método de *Broyden*

```
function [x i]= Broyden_SENL(Fsis, x0, TOL, n=35)
%INPUT:
%Fsis: función de ℝ^n a ℝ^n asociada al sistema
%x0: aproximación inicial
%TOL: tolerancia permitida
%n: numero maximo de iteraciones (35 por defecto)
%OUTPUT:
%x: aproximación final
%i: índice de la ultima iteración
fprintf('\t\t>> Método de Broyden\n');
fprintf('k\tx^(k-1)\t\tx^(k)\t\t||x^(k)-x^(k-1)||oo\n');
fprintf('_____________________________________________________________\n');

J0 = jacob(Fsis, x0, TOL);%se calcula la matriz jacobiana aplicada a x0

v = Fsis(x0);
if det(J0) == 0 %se comprueba que sea invertible
  disp("Utilice otro método")
  return
endif

A = inv(J0);%se calcula la inversa
s = (A*v).*(-1);
x = x0 + s;

err = max(abs(x-x0));
if err < TOL %se comprueba si ya se satisface la tolerancia
  return
else
  i = 1;
  fprintf('%d\t%10.7f\t%10.7f\t%10.7f\n',i,x0(1),x(1),err);
  for j = 2:length(x)
    fprintf('\t%10.7f\t%10.7f\t\n',x0(j),x(j));
  endfor
  fprintf('_____________________________________________________________\n')
  x0 = x;% si no se satisface, se continúa el mismo proceso
endif

for i = 2:n
    w = v;
    v = Fsis(x0);
    y = v-w;

```

```
    z = (A*y).*(-1);
    p = ((s')*z).*(-1);
    u = ((s')*A);

    A = A + ((s+z)*u).*(1/p);

    s = (A*v).*(-1);
    x  = x0 + s;
    err = max(abs(x-x0));%para comparar con la tolerancia (es la norma infinito)

    %se refresca la tabla
    fprintf('%d\t%10.7f\t%10.7f\t%10.7f\n',i,x0(1),x(1),err);
    for j = 2:length(x)
      fprintf('\t%10.7f\t%10.7f\t\n',x0(j),x(j));
    endfor
    fprintf('_____________________________________________________________\n')

    if err < TOL %se comprueba si ya se satisface la tolerancia
        return
    else
        x0 = x;% si no se satisface, se realiza el mismo proceso
    endif
endfor
%se llega al máximo de iteraciones
disp("Número máximo de iteraciones alcanzado")
return
endfunction
```

4.4 Método del descenso más rápido

Este método determina un mínimo local para una función de varias variables de la forma $g : \mathbb{R}^n \longrightarrow \mathbb{R}$. La relación entre un sistema de ecuaciones no lineales se debe a que un sistema lineal de la forma:

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) = 0 \\ f_2(x_1, x_2, \ldots, x_n) = 0 \\ \qquad\qquad \vdots \\ f_n(x_1, x_2, \ldots, x_n) = 0 \end{cases}$$

, tiene solución $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$ justo cuando la función g definida como $g(x_1, x_2, \ldots, x_n) = (f_1(x_1, x_2, \ldots, x_n))^2 + (f_2(x_1, x_2, \ldots, x_n))^2 + \cdots + (f_n(x_1, x_2, \ldots, x_n))^2$ tiene el valor mínimo.

El algoritmo se describe de forma sencilla así:

1. Evaluar g en una aproximación inicial $\mathbf{x}^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}\right)^t$.

2. Determinar una dirección d desde $\mathbf{x}^{(0)}$ que origine una disminución del valor de g.

3. Desplazar una cantidad adecuada en la dirección d y llamar al nuevo vector $\mathbf{x}^{(1)}$.

4. Repetir los pasos del 1 al 3 con el vector $\mathbf{x}^{(1)}$.

Para determinar la dirección de cada $\mathbf{x}^{(k)}$ desde $\mathbf{x}^{(k-1)}$ de manera que se garantice una disminución del valor de g, se introduce la siguiente definición:

• **Definición 6:** Si $g : \mathbb{R}^n \longrightarrow \mathbb{R}$, el *gradiente de* g en $\mathbf{x}$ con $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$ se denota como $\nabla g(\mathbf{x})$ y se define:

$$\nabla g(\mathbf{x}) = \left(\frac{\partial g}{\partial x_1}(\mathbf{x}), \frac{\partial g}{\partial x_2}(\mathbf{x}), \ldots, \frac{\partial g}{\partial x_n}(\mathbf{x}) \right)^t$$

Así la dirección viene dada por un vector $\mathbf{s}^{(k)} = -\nabla g\left(\mathbf{x}^{(k)}\right)$, notar que se considera el gradiente negativo dado que se necesita un descenso en los valores de g.

Luego para determinar el tamaño del desplazamiento, se considera una variable α_k y una función h real-evaluada que cumpla:

$$\begin{cases} h\left(\alpha_k\right) = g\left(\mathbf{x}^{(k)} + \alpha_k \mathbf{s}^{(k)}\right) \\ h'\left(\alpha_k\right) = 0 \end{cases}$$

, donde se toma el valor de α_k que produzca el menor valor en h.

Para obtener el valor de α_k, primero se suponen $\alpha_k^{(1)} = 0$ y $\alpha_k^{(3)} = 1$, luego se calculan:

$$\begin{aligned} g_1 &= g\left(\mathbf{x}^{(k)} - \alpha_k^{(1)} \nabla g\left(\mathbf{x}^{(k)}\right)\right) \\ g_3 &= g\left(\mathbf{x}^{(k)} - \alpha_k^{(3)} \nabla g\left(\mathbf{x}^{(k)}\right)\right) \end{aligned}$$

Si $g_3 < g_1$, se acepta $\alpha_k^{(3)}$ y se define $\alpha_k^{(2)} = \alpha_k^{(3)}/2$, caso contrario se toma $\alpha_k^{(3)} = \alpha_k^{(3)}/2$ y se vuelve a realizar el mismo proceso para determinar un $\alpha_k^{(2)}$. Al determinar los tres: $\alpha_k^{(1)}$, $\alpha_k^{(2)}$ y $\alpha_k^{(3)}$ y sus respectivos g_1 , $g_2 = g\left(\mathbf{x}^{(k)} - \alpha_k^{(2)} \nabla g\left(\mathbf{x}^{(k)}\right)\right)$ y g_3, se considera el polinomio interpolante mediante diferencias divididas de Newton sobre estos tres puntos:

$$\begin{aligned} P(\alpha) &= g_1 + h_1 \alpha + h_3 \alpha\left(\alpha - \alpha_2\right) \\ \text{; donde:} \qquad h_1 &= \frac{g_2 - g_1}{\alpha_2 - \alpha_1} \\ h_2 &= \frac{g_3 - g_2}{\alpha_3 - \alpha_2} \\ h_3 &= \frac{h_2 - h_1}{\alpha_3 - \alpha_1} \end{aligned}$$

Notar que la solución de $P'(\alpha) = 0$ viene dada por:

$$\alpha = \frac{1}{2}\left(\alpha_2 - \frac{h_1}{h_3}\right)$$

Por último se comparan $g_0 = g\left(\mathbf{x}^{(k)} - \alpha \nabla g\left(\mathbf{x}^{(k)}\right)\right)$ con g_3, y de nuevo se toma el valor entre α y α_3 que genere el menor valor en g y se continúa el proceso.

4.4.1 Algoritmo

$$\begin{cases} \mathbf{x}^{(0)} \in \mathbb{R}^n \\ \mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + \alpha_{k-1}\mathbf{s}^{(k-1)}, k \geq 1 \end{cases}$$

; donde $\mathbf{s}^{(k-1)} = -\nabla g\left(\mathbf{x}^{(k-1)}\right)$ y α_{k-1} cumple:

$$\begin{cases} h\left(\alpha_{k-1}\right) = g\left(\mathbf{x}^{(k-1)} + \alpha_{k-1}\mathbf{s}^{(k-1)}\right) \\ h'\left(\alpha_{k-1}\right) = 0 \end{cases}$$

, y además se toma el α_{k-1} que produzca el menor valor en h.

4.4.2 Ejemplo

Aproximar la solución del siguiente sistema con tolerancia de 0.05 y aproximación inicial $\mathbf{x}^{(0)} = (0,0,0)^t$

$$\begin{cases} x_1 + cos\left(x_1x_2x_3\right) - 1 & = 0 \\ \left(1-x_1\right)^{\frac{1}{4}} + x_2 + 0.05x_3^2 - 0.15x_3 - 1 & = 0 \\ -x_1^2 - 0.1x_2^2 + 0.01x_2 + x_3 - 1 & = 0 \end{cases}$$

∘ *Solución:*

$$\begin{aligned} f_1(\mathbf{x}) &= x_1 + cos\left(x_1x_2x_3\right) - 1 \\ f_2(\mathbf{x}) &= \left(1-x_1\right)^{\frac{1}{4}} + x_2 + 0.05x_3^2 - 0.15x_3 - 1 \\ f_3(\mathbf{x}) &= -x_1^2 - 0.1x_2^2 + 0.01x_2 + x_3 - 1 \end{aligned}$$

$$\begin{aligned} g(\mathbf{x}) &= \left(f_1(\mathbf{x})\right)^2 + \left(f_2(\mathbf{x})\right)^2 + \left(f_3(\mathbf{x})\right)^2 \\ &= \left(x_1 + cos\left(x_1x_2x_3\right) - 1\right)^2 + \left(\left(1-x_1\right)^{\frac{1}{4}} + x_2 + 0.05x_3^2 - 0.15x_3 - 1\right)^2 \\ &\quad + \left(-x_1^2 - 0.1x_2^2 + 0.01x_2 + x_3 - 1\right)^2 \end{aligned}$$

$$\begin{aligned} \nabla g(\mathbf{x}) = \Bigg(& 2f_1(\mathbf{x})\frac{\partial f_1}{\partial x_1}(\mathbf{x}) + 2f_2(\mathbf{x})\frac{\partial f_1}{\partial x_1}(\mathbf{x}) + 2f_3(\mathbf{x})\frac{\partial f_1}{\partial x_1}(\mathbf{x}), \\ & 2f_1(\mathbf{x})\frac{\partial f_1}{\partial x_2}(\mathbf{x}) + 2f_2(\mathbf{x})\frac{\partial f_2}{\partial x_2}(\mathbf{x}) + 2f_3(\mathbf{x})\frac{\partial f_3}{\partial x_2}(\mathbf{x}), \\ & 2f_1(\mathbf{x})\frac{\partial f_1}{\partial x_3}(\mathbf{x}) + 2f_2(\mathbf{x})\frac{\partial f_2}{\partial x_3}(\mathbf{x}) + 2f_3(\mathbf{x})\frac{\partial f_3}{\partial x_3}(\mathbf{x}), \Bigg) \end{aligned}$$

▷ *Primera iteración:*

$$g\left(\mathbf{x}^{(0)}\right) = 1 \qquad \nabla g\left(\mathbf{x}^{(0)}\right) = (0.1, -0.01, -2)^t \Rightarrow \mathbf{s}_0 = (-0.1, 0.01, 2)^t$$

$\alpha_0 = 0.5007 \Rightarrow \mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \alpha_0 s_0 = (-0.050068074, 0.005006807, 1.001361489)^t$

▷ *Segunda iteración:*

$$\begin{aligned} g\left(\mathbf{x}^{(1)}\right) = 0.0093593 \qquad & \nabla g\left(\mathbf{x}^{(0)}\right) = (-0.059616, -0.16555, 0.0056453)^t \\ & \Rightarrow \mathbf{s}_1 = (0.059616, 0.16555, -0.0056453)^t \end{aligned}$$

$\alpha_1 = 0.5862 \Rightarrow \mathbf{x}^{(2)} = \mathbf{x}^{(1)} - \alpha_1 s_1 = (-0.050068074, 0.005006807, 1.001361489)^t$

▷ *Iteraciones siguientes:*

Tabla 4: Iteraciones de método de *descenso más rápido*

k	$\mathbf{x}^{(k-1)}$	$\mathbf{x}^{(k)}$	$\left\|\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)}\right\|_\infty$
1	$\begin{pmatrix} 0.0000000 \\ 0.0000000 \\ 0.0000000 \end{pmatrix}$	$\begin{pmatrix} -0.050068074 \\ 0.005006807 \\ 1.001361489 \end{pmatrix}$	1.001361489
2	$\begin{pmatrix} -0.050068074 \\ 0.005006807 \\ 1.001361489 \end{pmatrix}$	$\begin{pmatrix} -0.015119550 \\ 0.102054203 \\ 0.998052052 \end{pmatrix}$	0.097047396
3	$\begin{pmatrix} -0.015119550 \\ 0.102054203 \\ 0.998052052 \end{pmatrix}$	$\begin{pmatrix} -0.001526812 \\ 0.097176021 \\ 1.000086597 \end{pmatrix}$	0.013592738

Se considera la aproximación: $\mathbf{x}^{(3)} = (-0.001526812, 0.097176021, 1.000086597)^t$ con tolerancia de 0.05.

4.4.3 Código

Código 5: Implementación del método de *descenso más rápido*

```
function [x i]= DescensoMR_SENL(Fsis, x0, TOL, n=35)
%INPUT:
%Fsis: función de R^n a R^n asociada al sistema
%x0: aproximación inicial
%TOL: tolerancia permitida
%n: numero maximo de iteraciones (35 por defecto)
%OUTPUT:
%x: aproximación final
%i: índice de la ultima iteración
fprintf('\t\t>> Método de Descenso más rápido\n');
fprintf('k\tx^(k-1)\t\tx^(k)\t\t||x^(k)-x^(k-1)||oo\n');
fprintf('_____________________________________________________________\n');

for i = 1:n
    g1 = gval(Fsis, x0);%valor de x0 en la función escalar g
    z = ggrad(Fsis, x0, TOL);%valor del gradiende de g en x0

    a1 = 0.0;%primeros valores de α
    a3 = 1.0;
    v = x0 - a3.*z;%se calcula la prime aproximación
    g3 = gval(Fsis, v);%se calcula el valor de g en esa aproximación

    while g3 >= g1%mientras g3 supere a g1, se buscan nuevos a3
        a3 = a3/2;
        v = x0 - a3.*z;
        g3 = gval(Fsis, v);
    endwhile

    a2 = a3/2;%se tiene a1 y a3, ahora se define a2
```

```
    v = x0 - a2.*z;
    g2 = gval(Fsis, v);

    %esto es por las diferencias divididas de Newton
    h1 = (g2-g1)/a2;
    h2 = (g3-g2)/(a3-a2);
    h3 = (h2-h1)/a3;

    %ahora se estima el α
    a0 = 0.5*(a2-(h1/h3));
    v = x0 - a0.*z;
    g0 = gval(Fsis, v);

    if g0 > g3
        a = a3;
    else
        a = a0;
    endif

    x = x0 - a.*z;%se calcula la aproximación

    err = max(abs(x-x0));%para comparar con la tolerancia (es la norma infinito)

    fprintf('%d\t%10.9f\t%10.9f\t%10.9f\n',i,x0(1),x(1),err);
    for j = 2:length(x)
      fprintf('\t%10.9f\t%10.9f\t\n',x0(j),x(j));
    endfor
    fprintf('_____________________________________________________________\n');

    if err < TOL %se comprueba si ya se satisface la tolerancia
        return
    else
        x0 = x; % si no se satisface, se realiza el mismo proceso
                %refrescando el valor de x0
    endif
endfor
%se llega al maximo de iteraciones
disp("Numero maximo de iteraciones alcanzado")
return
endfunction
```

Código 6: Aproximación de g en $\mathbf{x}$

```
function gx = gval(Fsis,x)
%INPUT:
%Fsis: función de ℝ^n a ℝ^n asociada al sistema
%x: vector
%OUTPUT:
```

```
%gx: valor de g evaluado en x; es la suma de los cuadrados
% de las componentes de F al evaluarse en x.
total = Fsis(x);
total = total.^2;
gx = sum(total);
endfunction
```

Código 7: Aproximación del gradiente de g en $\mathbf{x}$

```
function ggx = ggrad(Fsis,x,h)
%INPUT:
%Fsis: función de ℝ^n a ℝ^n asociada al sistema
%x: vector
%h: tamaño de paso
%OUTPUT:
%ggx: vector del gradiente de g en x
ggx = 1:length(x);
y = x;
tem1 = [];
tem2 = [];
tem3 = [];
tem4 = [];
for i = 1:length(x)
    y(i) = y(i)+h;
    tem1 = Fsis(y);
    tem2 = Fsis(x);
    tem3 = tem1.-tem2;
    tem4 = tem3./h;
    ggx(i) = 2*sum(tem4.*tem2);
    y = x;
endfor
ggx = ggx';
endfunction
```

4.5 Método de continuación

También se llama *método de homotopía* e introduce el problema de aproximación dentro de una familia de problemas. Específicamente para un problema de tipo $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ con solución desconocida $\mathbf{x}^*$, se considera una familia de problemas descritos mediante un parámetro $\lambda \in [0, 1]$, donde un problema con solución conocida $\mathbf{x}(0)$ corresponde a $\lambda = 0$ y la solución desconocida $\mathbf{x}(1) = \mathbf{x}^*$ corresponde a $\lambda = 1$.

Bajo estas condiciones se puede definir una función $\mathbf{G}$ de esta forma por ejemplo:

$$\begin{aligned} \mathbf{G} : [0,1] \times \mathbb{R}^n &\longrightarrow \mathbb{R}^n \\ (\lambda, \mathbf{x}) &\longmapsto \mathbf{G}(\lambda, \mathbf{x}) = \mathbf{F}(\mathbf{x}) + (\lambda - 1)\mathbf{F}(\mathbf{x}(0)) \end{aligned}$$

La función $\mathbf{G}$ con el parámetro λ proporciona una familia de funciones que pueden conducir del valor conocido $\mathbf{x}(0)$ a la solución $\mathbf{x}(1) = \mathbf{x}^*$. La función $\mathbf{G}$ se llama *homotopía* entre la

función $\mathbf{G}(0,\mathbf{x}) = \mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{x}(0))$ y la función $\mathbf{G}(1,\mathbf{x}) = \mathbf{F}(\mathbf{x})$. El problema de continuación consiste en: Determinar una forma de pasar de la solución conocida $\mathbf{x}(0)$ de $\mathbf{G}(0,\mathbf{x})$ a la solución desconocida $\mathbf{x}(1) = \mathbf{x}^*$ de $\mathbf{G}(1,\mathbf{x}) = \mathbf{F}(\mathbf{x})$.

El conjunto $\{\mathbf{x}(\lambda) \,|0 \leq \lambda \leq 1\}$ puede verse como una curva parametrizada por λ que va desde $\mathbf{x}(0) = \mathbf{x}^{(0)}$ a $\mathbf{x}(1) = \mathbf{x}^*$. Si las funciones $\mathbf{G}(\lambda,\mathbf{x})$ y $\mathbf{x}(\lambda)$ son diferenciables, al derivar $\mathbf{G}(\lambda,\mathbf{x}) = \mathbf{0}$ respecto a λ se tiene que:

$$\frac{\partial \mathbf{G}(\lambda,\mathbf{x}(\lambda))}{\partial \lambda} + \frac{\partial \mathbf{G}(\lambda,\mathbf{x}(\lambda))}{\partial \mathbf{x}} \mathbf{x}'(\lambda) = \mathbf{0}$$

Despejando $\mathbf{x}'(\lambda)$:

$$\mathbf{x}'(\lambda) = -\left[\frac{\partial \mathbf{G}(\lambda,\mathbf{x}(\lambda))}{\partial \mathbf{x}}\right]^{-1} \frac{\partial \mathbf{G}(\lambda,\mathbf{x}(\lambda))}{\partial \lambda}$$

Como $\mathbf{G}(\lambda,\mathbf{x}) = \mathbf{F}(\mathbf{x}) + (\lambda - 1)\mathbf{F}(\mathbf{x}(0))$, se tiene que:

$$\frac{\partial \mathbf{G}(\lambda,\mathbf{x}(\lambda))}{\partial \mathbf{x}} = J(\mathbf{x}(\lambda)) \quad \text{y} \quad \frac{\partial \mathbf{G}(\lambda,\mathbf{x}(\lambda))}{\partial \lambda} = \mathbf{F}(\mathbf{x}(0)) \text{ , por lo que:}$$

$$\mathbf{x}'(\lambda) = -\left[J_{\mathbf{F}}(\mathbf{x}(\lambda))\right]^{-1} \mathbf{F}(\mathbf{x}(0)) \text{ , para } 0 \leq \lambda \leq 1 \text{ y } \mathbf{x}(0) \text{ conocido.}$$

El siguiente teorema da condiciones bajo las cuales es factible el método de continuación:

• **Teorema 7:** Sea $\mathbf{F}(\mathbf{x})$ una función continuamente diferenciable en $\mathbb{R}^n$ para $\mathbf{x} \in \mathbb{R}^n$. Suponga que la *matriz Jacobiana* $J_{\mathbf{F}}(\mathbf{x})$ es no singular para cada $\mathbf{x} \in \mathbb{R}^n$ y que existe una constante M tal que $\|J_{\mathbf{F}}(\mathbf{x})^{-1}\| \leq M$, para cada $\mathbf{x} \in \mathbb{R}^n$. Entonces, para cualquier $\mathbf{x}(0)$ en $\mathbb{R}^n$, existe una única solución $\mathbf{x}(\lambda)$, tal que:

$$\mathbf{G}(\lambda,\mathbf{x}(\lambda)) = \mathbf{0}$$

, para cada $\lambda \in [0,1]$. Además, $\mathbf{x}(\lambda)$ es continuamente diferenciable y

$$\mathbf{x}'(\lambda) = -\left[J_{\mathbf{F}}(\mathbf{x}(\lambda))\right]^{-1} \mathbf{F}(\mathbf{x}(0)) \text{ , para cada } \lambda \in [0,1]$$

El sistema de ecuaciones que se debe resolver tiene esta forma:

$$\begin{aligned}
\frac{dx_1}{d\lambda} &= \phi_1(\lambda, x_1, x_2, \ldots, x_n) \\
\frac{dx_2}{d\lambda} &= \phi_2(\lambda, x_1, x_2, \ldots, x_n) \\
&\vdots \\
\frac{dx_n}{d\lambda} &= \phi_n(\lambda, x_1, x_2, \ldots, x_n)
\end{aligned}$$

; donde

$$\begin{bmatrix} \phi_1(\lambda, x_1, x_2, \ldots, x_n) \\ \phi_2(\lambda, x_1, x_2, \ldots, x_n) \\ \vdots \\ \phi_n(\lambda, x_1, x_2, \ldots, x_n) \end{bmatrix} = -J_{\mathbf{F}}(x_1, x_2, \ldots, x_n)^{-1} \begin{bmatrix} f_1(\mathbf{x}(0)) \\ f_2(\mathbf{x}(0)) \\ \vdots \\ f_n(\mathbf{x}(0)) \end{bmatrix}$$

Para resolverlo se utiliza el método de Runge-Kutta de cuarto orden, primero se elige un entero de $N > 0$ y $h = \frac{(1-0)}{N} = \frac{1}{N}$. Se divide el intervalo $[0, 1]$ en N subintervalos con los puntos de red: $\lambda_j = jh$, para cada $j = 0, 1, 2, \ldots, N$. Se usa la notación w_{ij} para cada $i = 0, 1, 2, \ldots, N$ y $j = 0, 1, 2, \ldots, n$, para denotar una aproximación a $x_i(\lambda_j)$. Para la condición inicial, se toma: $w_{1,0} = x_1(0), w_{2,0} = x_2(0), \ldots, w_{n,0} = x_n(0)$.

Suponiendo que se han calculado $w_{1,j}, w_{2,j}, \ldots, w_{n,j}$, se obtiene $w_{1,j+1}, w_{2,j+1}, \ldots, w_{n,j+1}$ usando las ecuaciones:

$$\begin{aligned}
k_{1,i} &= h\phi_i\left(\lambda_j, w_{1,j}, w_{2,j}, \ldots, w_{n,j}\right) \text{ , para cada } i = 1, 2, \ldots, n \\
k_{2,i} &= h\phi_i\left(\lambda_j + \frac{h}{2}, w_{1,j} + \frac{k_{1,1}}{2}, w_{2,j} + \frac{k_{1,2}}{2}, \ldots, w_{n,j} + \frac{k_{1,n}}{2}\right) \text{ , para cada } i = 1, 2, \ldots, n \\
k_{3,i} &= h\phi_i\left(\lambda_j + \frac{h}{2}, w_{1,j} + \frac{k_{2,1}}{2}, w_{2,j} + \frac{k_{2,2}}{2}, \ldots, w_{n,j} + \frac{k_{2,n}}{2}\right) \text{ , para cada } i = 1, 2, \ldots, n \\
k_{4,i} &= h\phi_i\left(\lambda_j + h, w_{1,j} + k_{3,1}, w_{2,j} + k_{3,2}, \ldots, w_{n,j} + k_{3,n}\right) \text{ , para cada } i = 1, 2, \ldots, n
\end{aligned}$$

y finalmente

$$w_{i,j+1} = w_{i,j} + \frac{1}{6}\left(k_{1,i} + 2k_{2,i} + 2k_{3,i} + k_{4,i}\right) \text{ , para cada } i = 1, 2, \ldots, n$$

Se usa la siguiente notación para simplificar la presentación:

$$\mathbf{k}_1 = \begin{bmatrix} k_{1,1} \\ k_{1,2} \\ \vdots \\ k_{1,n} \end{bmatrix}, \quad \mathbf{k}_2 = \begin{bmatrix} k_{2,1} \\ k_{2,2} \\ \vdots \\ k_{2,n} \end{bmatrix}, \quad \mathbf{k}_3 = \begin{bmatrix} k_{3,1} \\ k_{3,2} \\ \vdots \\ k_{3,n} \end{bmatrix}, \quad \mathbf{k}_4 = \begin{bmatrix} k_{4,1} \\ k_{4,2} \\ \vdots \\ k_{4,n} \end{bmatrix} \quad \text{, y} \quad \mathbf{w}_j = \begin{bmatrix} w_{1,j} \\ w_{2,j} \\ \vdots \\ w_{n,j} \end{bmatrix}$$

Notar que $\mathbf{x}(0) = \mathbf{x}(\lambda_0) = \mathbf{w}_0$, y para cada $j = 0, 1, 2, \ldots, n$ se tiene:

$$\begin{aligned}
\mathbf{k}_1 &= h \begin{bmatrix} \phi_1(\lambda_j, w_{1,j}, w_{2,j}, \ldots, w_{n,j}) \\ \phi_2(\lambda_j, w_{1,j}, w_{2,j}, \ldots, w_{n,j}) \\ \vdots \\ \phi_n(\lambda_j, w_{1,j}, w_{2,j}, \ldots, w_{n,j}) \end{bmatrix} = h\left[-J_{\mathbf{F}}\left(w_{1,j}, w_{2,j}, \ldots, w_{n,j}\right)\right]^{-1}\mathbf{F}(\mathbf{x}(0)) \\
&= h\left[-J_{\mathbf{F}}\left(\mathbf{w}_j\right)\right]^{-1}\mathbf{F}(\mathbf{x}(0)) \\
\mathbf{k}_2 &= h\left[-J_{\mathbf{F}}\left(\mathbf{w}_j + \frac{\mathbf{k}_1}{2}\right)\right]^{-1}\mathbf{F}(\mathbf{x}(0)) \\
\mathbf{k}_3 &= h\left[-J_{\mathbf{F}}\left(\mathbf{w}_j + \frac{\mathbf{k}_2}{2}\right)\right]^{-1}\mathbf{F}(\mathbf{x}(0)) \\
\mathbf{k}_4 &= h\left[-J_{\mathbf{F}}\left(\mathbf{w}_j + \mathbf{k}_3\right)\right]^{-1}\mathbf{F}(\mathbf{x}(0))
\end{aligned}$$

y

$$\mathbf{x}\left(\lambda_{j+1}\right) = \mathbf{x}\left(\lambda_j\right) + \frac{1}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right) = \mathbf{w}_j + \frac{1}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right)$$

Finalmente $\mathbf{x}\left(\lambda_n\right) = \mathbf{x}(1)$ es la aproximación de $\mathbf{x}^*$.

4.5.1 Algoritmo

$$\begin{cases} \mathbf{x}(0) \in \mathbb{R}^n \\ \mathbf{x}\left(\lambda_{j+1}\right) = \mathbf{x}\left(\lambda_j\right) + \frac{1}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right) \end{cases}$$

; donde dado $N > 0$, se hace $h = 1/N$ y luego para cada λ_j se tiene $0 \leq \lambda_j \leq 1$ y $\lambda_j = jh$ para cada $j = 0, 1, 2, \ldots, N$, y además:

$$\begin{aligned} \mathbf{k}_1 &= \left[J_{\mathbf{F}}\left(\mathbf{x}\left(\lambda_j\right)\right)\right]^{-1}\mathbf{b} \\ \mathbf{k}_2 &= \left[J_{\mathbf{F}}\left(\mathbf{x}\left(\lambda_j\right) + \frac{\mathbf{k}_1}{2}\right)\right]^{-1}\mathbf{b} \\ \mathbf{k}_3 &= \left[J_{\mathbf{F}}\left(\mathbf{x}\left(\lambda_j\right) + \frac{\mathbf{k}_2}{2}\right)\right]^{-1}\mathbf{b} \\ \mathbf{k}_4 &= \left[J_{\mathbf{F}}\left(\mathbf{x}\left(\lambda_j\right) + \mathbf{k}_3\right)\right]^{-1}\mathbf{b} \end{aligned}$$

; con $\mathbf{b} = -h\mathbf{F}(\mathbf{x}(0))$.

4.5.2 Ejemplo

Encontrar la solución del siguiente sistema de ecuaciones con $N = 2$:

$$\begin{cases} 4x_1^2 - 20x_1 + \frac{1}{4}x_2^2 + 8 &= 0 \\ \frac{1}{2}x_1x_2^2 + 2x_1 - 5x_2 + 8 &= 0 \end{cases}$$

∘ *Solución:*

$$N = 2 \Rightarrow h = 1/2 = 0.5 \Rightarrow \lambda_0 = 0, \lambda_1 = 0.5, \lambda_2 = 1$$

$$\text{Sea } \mathbf{x}(0) = (0,0)^t \Rightarrow \mathbf{b} = -h\mathbf{F}(\mathbf{x}(0)) = -0.5(8,8)^t = (-4,-4)^t$$

▷ *Primera iteración:*

$$J_{\mathbf{F}}\left(\mathbf{x}\left(0\right)\right) = \begin{bmatrix} -20 & 0 \\ 2 & -5 \end{bmatrix} \Rightarrow \left[J_{\mathbf{F}}\left(\mathbf{x}\left(0\right)\right)\right]^{-1} = \begin{bmatrix} -0.05 & 0 \\ -0.02 & -0.2 \end{bmatrix}$$

$$\mathbf{k}_1 = \begin{pmatrix} 0.2 \\ 0.88 \end{pmatrix}, \quad \mathbf{k}_2 = \begin{pmatrix} 0.2186 \\ 0.8996 \end{pmatrix}, \quad \mathbf{k}_3 = \begin{pmatrix} 0.2197 \\ 0.9012 \end{pmatrix}, \quad \mathbf{k}_4 = \begin{pmatrix} 0.2429 \\ 0.9547 \end{pmatrix}$$

$$\Rightarrow \mathbf{x}\left(0.5\right) = \mathbf{x}\left(0\right) + \frac{1}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right) = (0.2199375, 0.9060504)^t$$

▷ *Segunda iteración:*

$$J_{\mathbf{F}}\left(\mathbf{x}\left(0.5\right)\right) = \begin{bmatrix} -18.2405 & 0.4530 \\ 2.4105 & -4.8007 \end{bmatrix} \Rightarrow \left[J_{\mathbf{F}}\left(\mathbf{x}\left(0.5\right)\right)\right]^{-1} = \begin{bmatrix} -0.055515 & -0.0052388 \\ -0.027875 & -0.21093 \end{bmatrix}$$

$$\mathbf{k}_1 = \begin{pmatrix} 0.2430 \\ 0.9552 \end{pmatrix}, \quad \mathbf{k}_2 = \begin{pmatrix} 0.2742 \\ 1.0626 \end{pmatrix}, \quad \mathbf{k}_3 = \begin{pmatrix} 0.2786 \\ 1.0798 \end{pmatrix}, \quad \mathbf{k}_4 = \begin{pmatrix} 0.3321 \\ 1.3264 \end{pmatrix}$$

$$\Rightarrow \mathbf{x}\left(1\right) = \mathbf{x}\left(0.5\right) + \frac{1}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right) = (0.5000510, 2.0004611)^t$$

▷ *Iteraciones:*

Tabla 5: Iteraciones de método de *continuación*

j	$\mathbf{x}(\lambda_j)$	$\mathbf{x}(\lambda_{j+1})$	$\|\mathbf{x}(\lambda_{j+1}) - \mathbf{x}(\lambda_j)\|_\infty$
0	$\begin{pmatrix} 0.0000000 \\ 0.0000000 \end{pmatrix}$	$\begin{pmatrix} 0.2199375 \\ 0.9060504 \end{pmatrix}$	0.9060504
1	$\begin{pmatrix} 0.2199375 \\ 0.9060504 \end{pmatrix}$	$\begin{pmatrix} 0.5000510 \\ 2.0004611 \end{pmatrix}$	1.0944107

4.5.3 Código

Código 8: Implementación de método de *continuación*

```
function [x i]= Continuacion_SENL(Fsis, x0, N)
%INPUT:
%Fsis: función de R^n a R^n asociada al sistema
%x0: aproximación inicial
%N: número de iteraciones
%OUTPUT:
%x: aproximación final
%i: índice de la ultima iteración
fprintf('\t\t>> Método de Continuación\n');
fprintf('j\tx(j)\t\t x(j+1)\t\t||x(j+1)-x(j)||oo\n');
fprintf('_____________________________________________________________\n');
h = 1/N;
b = (Fsis(x0)).*(-h);
for i = 1:N

    A = jacob(Fsis, x0, 10^(-10));%se calcula la matriz jacobiana aplicada a x0

    AI = inv(A);%se calcula la inversa

    k1 = AI*b;

    A = jacob(Fsis, x0+k1.*(1/2), 10^(-10));
    AI = inv(A);%se calcula la inversa
    k2 = AI*b;

    A = jacob(Fsis, x0+k2.*(1/2), 10^(-10));
    AI = inv(A);%se calcula la inversa
    k3 = AI*b;

    A = jacob(Fsis, x0+k3, 10^(-10));
    AI = inv(A);%se calcula la inversa
    k4 = AI*b;

    x = x0 + (k1 + k2.*2 + k3.*2 + k4).*(1/6);

    err = max(abs(x-x0));%para comparar con la tolerancia (es la norma infinito)

```

```
    %se refresca la tabla
    fprintf('%d\t%10.7f\t%10.7f\t%10.7f\n',i,x0(1),x(1),err);
    for j = 2:length(x)
      fprintf('\t%10.7f\t%10.7f\t\n',x0(j),x(j));
    endfor
    fprintf('_____________________________________________________________\n');

    if i == N
      return
    else
      x0 = x;
    endif
endfor
endfunction
```

5 Problema de aplicación

5.1 Descripción

- Una empresa de teléfonos desea introducir un nuevo producto para competir con otros.
- La empresa planea invertir un millón de dólares para desarrollar este producto.
- El éxito del producto dependerá de la inversión en campaña de marketing y el precio final del producto.
- Esta es la fórmula utilizada por el departamento de marketing para estimar las ventas del nuevo producto durante el próximo año:

$$20000 + 5\sqrt{a} - 60p$$

; donde a: cantidad para invertir en la campaña de marketing
p: precio del teléfono

- El coste de producción del teléfono es de 100 dólares por unidad
- ¿Cómo podría la empresa maximizar sus ganancias para el próximo año?

5.2 Planteamiento

- Beneficios de las ventas:

$$\left(20000 + 5\sqrt{a} - 60p\right) p$$

- Costos totales de producción:

$$\left(20000 + 5\sqrt{a} - 60p\right) 100$$

- Costos de desarrollo:

$$1000000$$

- Costos de marketing:

$$a$$

- Beneficio total:

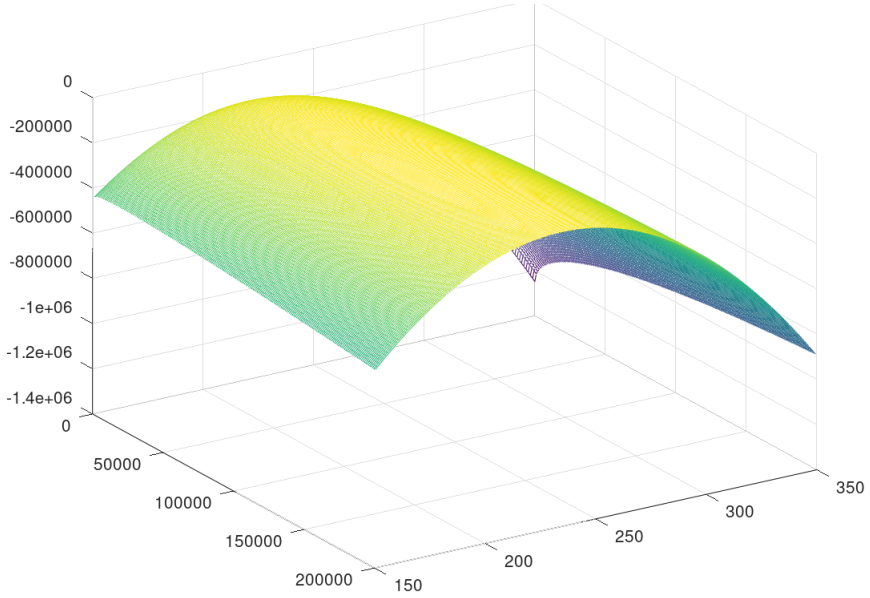
$$\mathbf{S}(a,p) = \left(20000 + 5\sqrt{a} - 60p\right)(p - 100) - 1000000 - a$$

5.3 Solución

- Se considera el gráfico de la superficie generada por $\mathbf{S}(a,p)$ en la *figura 1.*
- Se observa que es posible calcular un máximo local.
- Luego se considera el gradiente de $\mathbf{S}$:

$$\nabla\mathbf{S}(a,p) = \begin{pmatrix} 5(p-100)\frac{1}{2\sqrt{a}} - 1 \\ 20000 + 5\sqrt{a} - 120p + 6000 \end{pmatrix}$$

Figura 1:
$\mathbf{S}(a,p) = (20000 + 5\sqrt{a} - 60p)\,(p - 100) - 1000000 - a$



- Se alcanza el máximo local cuando $\nabla \mathbf{S}(a,p) = 0$
- Se tiene el siguiente sistema:

$$\begin{cases} 5(p-100)\frac{1}{2\sqrt{a}} - 1 & = 0 \\ 20000 + 5\sqrt{a} - 120p + 6000 & = 0 \end{cases}$$

- Para determinar una aproximación a su solución se crea el siguiente código en Octave/Matlab, donde se utiliza la implementación del método de *Newton* antes presentado (ver *código 2* y *código 3*)
- Observando el gráfico, se toma una aproximación inicial: $\mathbf{x}^{(0)} = (100000, 100)^t$ y tolerancia de 10^{-5}, además el código genera el gráfico de la superficie (*figura 1*) y se determina también el valor de la ganancia máxima.

Código 9: Problema

```
#Problema

#I) Para el gráfico de la superficie S:

%Para la malla
x = [1:1000:200000];
y = [151:1:350];

[X, Y] = meshgrid(x,y);

%Para la superficie
for i = 1:200
  for j = 1:200
```

```
    a = X(i,j);
    p = Y(i,j);
    %Se calculan los valores de Z para cada punto en la malla
    Z(i,j) = (20000+5*sqrt(a)-60*p)*(p-100)-1000000-a;
  endfor
endfor
%Se genera el gráfico de la superficie
mesh(X,Y,Z)

#II) Para resolver el sistema

%Primera función componente del gradiente
s1 = @(a,p)(5*(p-100)*(1/(2*sqrt(a)))-1);
%Segunda función componente del gradiente
s2 = @(a,p)(20000+5*(sqrt(a))-120*p+6000);
%Gradiente de S
Sgrad = @(x)[s1(x(1),x(2));s2(x(1),x(2))];

%Aproximación inicial
x0 = [100000;100];

%Aplicando implementación
[x i] = Newton_SENL(Sgrad,x0,10^-5,100)

#III) Para la ganancia máxima

S = @(a,p)((20000+5*sqrt(a)-60*p)*(p-100)-1000000-a);
S(x(1),x(2))
```

- Los resultados de la implementación se muestran en la *tabla 6*.

Tabla 6: Iteraciones

k	$\mathbf{x}^{(k-1)}$	$\mathbf{x}^{(k)}$	$\left\|\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)}\right\|_\infty$
1	$\begin{pmatrix}100000\\100\end{pmatrix}$	$\begin{pmatrix}49124.4947221\\226.4911063\end{pmatrix}$	50875.5052779
2	$\begin{pmatrix}49124.4947221\\226.4911063\end{pmatrix}$	$\begin{pmatrix}80332.5598434\\228.8351279\end{pmatrix}$	31208.0651213
3	$\begin{pmatrix}80332.5598434\\228.8351279\end{pmatrix}$	$\begin{pmatrix}101069.2416232\\230.0005069\end{pmatrix}$	20736.6817798
4	$\begin{pmatrix}101069.2416232\\230.0005069\end{pmatrix}$	$\begin{pmatrix}105826.0755469\\230.2248042\end{pmatrix}$	4756.8339238
5	$\begin{pmatrix}105826.0755469\\230.2248042\end{pmatrix}$	$\begin{pmatrix}106003.0165429\\230.2325481724832\end{pmatrix}$	176.9409959
6	$\begin{pmatrix}106003.0165429\\230.2325481724832\end{pmatrix}$	$\begin{pmatrix}106003.2449961\\230.2325581\end{pmatrix}$	0.2284533
7	$\begin{pmatrix}106003.2449961\\230.2325581\end{pmatrix}$	$\begin{pmatrix}106003.2449973\\230.2325581\end{pmatrix}$	0.0000011

- La implementación da como resultado una aproximación $\mathbf{x}^{(7)} = (106003.2449973, 230.2325581)^t$ con tolerancia de 10^{-5} un valor de $\$ - 88372$ como ganancia máxima.

5.4 Análisis de resultados

- Para maximizar las ganancias se deben invertir \$106003.24 en marketing y establecer un precio de venta de \$230.23 por cada teléfono.

- Al sustituir las variables obtenidas en la ecuación **S**, se estima la ganancia máxima en $\$ - 88372$.

- Se concluye que la venta del teléfono no dejará ganancias, sino lo contrario, dejará pérdidas y se recomienda no producir el teléfono con esas condiciones.

6 Conclusiones

6.1 Sobre los métodos de solución

1. El método de *punto fijo* requiere de varias condiciones que garanticen la convergencia de la solución como se ve en el *teorema 3*, además de que es necesario adaptar el sistema para cambiar el problema de búsqueda de solución a un problema de punto fijo como se manifestó en el ejemplo correspondiente, sin embargo hay que destacar que no requiere de muchos cálculos por cada iteración, el algoritmo correspondiente es bastante simple. Pero también hay que mencionar que su velocidad de convergencia es lenta.

2. El método de *Newton* es bastante sólido en velocidad de convergencia, pero requiere de condiciones fuertes para garantizar la convergencia, como se vio en el algoritmo correspondiente a este método es necesario que la *matriz Jacobiana* de la función asociada al sistema sea invertible al aplicarse a cada aproximación, además se realizan bastantes operaciones por cada iteración.

3. El método de *Broyden* relaja la condición de la *matriz Jacobiana* del método de *Newton* al proponer una aproximación para esas matrices, lo que reduce los cálculos pero pierde algo de la velocidad de convergencia que tenía el método de *Newton*.

4. Lo métodos de *Newton* y *Broyden* requieren de una buena aproximación inicial dado que se basan en un análisis de la *matriz Jacobiana* que involucra derivadas, se podría decir que son "métodos locales", esto es válido también para los métodos *cuasi-Newton* en general.

5. El método de *descenso más rápido* puede complementar a los métodos *cuasi-Newton* al converger a un punto que sería cercano a la solución verdadera, ese punto de convergencia podría tomarse como aproximación inicial para los métodos *cuasi-Newton*.

6. El método de *descenso más rápido* se basa un proceso generalizado para búsqueda de puntos críticos , haciendo unas modificaciones en el algoritmo correspondiente como por ejemplo tomar el *gradiente* de g con signo positivo y buscar puntos que maximicen a la función g, y así el proceso antes expuesto podría usarse para calcular máximos de una superficie.

7. En el método de *continuación* se realizan 4 inversiones por iteración correspondientes al cálculo de los vectores $\mathbf{k}_1$, $\mathbf{k}_2$, $\mathbf{k}_3$ y $\mathbf{k}_4$, en comparación el método de *Newton* realiza una inversión por iteración, una alternativa a esto sería utilizar el método de Runge-Kutta de orden 2.

8. La implementación de los métodos en el lenguaje de Octave/Matlab fue relativamente fácil, notar que para los cálculos que involucran derivadas se ha utilizado un aproximación sencilla y no se han utilizado paquetes externos.

6.2 Sobre el problema de aplicación

1. Se utilizó la observación del gráfico correspondiente a la superficie de la ecuación **S** (*figura 1*) para calcular la aproximación inicial, de no tener esta información se pudo optar por utilizar otro método auxiliar para encontrar la aproximación inicial.

2. Las observaciones del gráfico fueron importantes, de no tener esa ventaja como en sistemas de más de dos variables, al aplicar algún método de solución sobre el *gradiente* de $\mathbf{S}$, no se tendría claridad sobre si el un máximo o un mínimo, lo que implica otro tipo de análisis que involucra la segunda derivada, en concreto se puede tomar en cuenta la *matriz Hessiana* de $\mathbf{S}$, concepto que no se ha definido en este material pero que determinaría dado un punto crítico si este es máximo o mínimo.

3. El problema de maximización se transformó en problema de búsqueda de solución de un *sistema de ecuaciones no lineales*, esto también puede usarse para problemas de minimización o usar directamente el proceso expuesto el método de *descenso más rápido*.

7 Bibliografía y referencias

7.1 Libros

- Burden R. y Faires D., Numerical Analysis, ninth edition, 2011. Brooks Cole.
- Infante del Río, Juan Antonio, Métodos Numéricos, tercera edición, Ediciones Pirámide, 2002, 2007
- Ruano Lima, Julio Daniel, Análisis Numérico, Escuela Superior Politécnica del Litoral, Guayaquil, Ecuador, 2013

7.2 Enlaces

- Método de Broyden:
  - https://ocw.ehu.eus/file.php/81/capitulo-23.pdf
  - https://sites.google.com/site/cm4312014/sis/metodo-de-broyden
- Método de descenso más rápido:
  - https://sites.google.com/site/cm4312014/sis/metodo-dese
- Método de continuación:
  - http://personales.upv.es/dginesta/docencia/posgrado/prac_no_lineal.pdf
  - https://www.dropbox.com/scl/fi/aukvccnixw1d3tlg2wvp6/Soluci%C3%B3n-num%C3%A9rica-de-sistemas-de-ecuaciones-no-lineales.paper?dl=0&rlkey=4m4kkctov8dkl4
- Problema de aplicación:
  - https://rua.ua.es/dspace/bitstream/10045/16373/3/Microsoft%20Word%20-%203.SISTEMAS%20ECUACIONES%20NO%20LINEALES.pdf