

10.1.5 El sistema no lineal 
$$\begin{aligned} x_1^2 - 10x_1 + x_2^2 + 8 &= 0 \\ x_1x_2^2 + x_1 - 10x_2 + 8 &= 0 \end{aligned}$$
 puede transformarse en el problema de punto fijo

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} g_1(x_1, x_2) \\ g_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} \frac{x_1^2 + x_2^2 + 8}{10} \\ \frac{x_1x_2^2 + x_1 + 8}{10} \end{pmatrix}$$

- a) Demuestre que  $G = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}$  que mapea  $D \subset \mathbb{R}^2$  en  $\mathbb{R}^2$  tiene un punto fijo en  $D = \{(x_1, x_2)^t: 0 \leq x_1, x_2 \leq 1.5\}$ .  
b) Aplique la iteración funcional para aproximar la solución.  
c) ¿Acelera la convergencia el método de Gauss-Seidel?

a) Notemos que

$$\begin{aligned} |g_1(x_1, x_2)| &= \left| \frac{x_1^2 + x_2^2 + 8}{10} \right| \leq \left| \frac{2(1.5)^2 + 8}{10} \right| = \frac{12.5}{10} = 1.25 \\ |g_2(x_1, x_2)| &= \left| \frac{x_1x_2^2 + x_1 + 8}{10} \right| \leq \left| \frac{(1.5)^3 + 1.5 + 8}{10} \right| = \frac{12.875}{10} = 1.2875 \end{aligned}$$

Entonces  $g_1, g_2$  son acotadas en  $D$ , por tanto  $G(x_1, x_2) \in D$ . Ahora veamos que las derivadas parciales de  $g_1, g_2$  también son acotadas en  $D$ :

$$\begin{aligned} \left| \frac{\partial g_1}{\partial x_1} \right| &= \left| \frac{2x_1}{10} \right| \leq \left| \frac{2(1.5)}{10} \right| = 0.3 \\ \left| \frac{\partial g_1}{\partial x_2} \right| &= \left| \frac{2x_2}{10} \right| \leq \left| \frac{2(1.5)}{10} \right| = 0.3 \\ \left| \frac{\partial g_2}{\partial x_1} \right| &= \left| \frac{1 + x_2^2}{10} \right| \leq \left| \frac{1 + (1.5)^2}{10} \right| = \frac{3.25}{10} = 0.325 \\ \left| \frac{\partial g_2}{\partial x_2} \right| &= \left| \frac{2x_1x_2}{10} \right| \leq \left| \frac{2(1.5)^2}{10} \right| = \frac{4.5}{10} = 0.45 \end{aligned}$$

Luego  $g_1, g_2$  son continuas en  $D$ , por lo tanto  $G = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}$  es continua en  $D$  y además  $G(x_1, x_2) \in D$  siempre que  $(x_1, x_2) \in D$ . Esto implica que  $G$  tiene un único punto fijo en  $D$ .

- b) Se aproximó la solución aplicando iteración de punto fijo. Esto fue implementado en un script de Matlab, se consideró como punto inicial  $x^{(0)} = (0,0)$ , tolerancia de  $10^{-7}$ ,  $10^{-12}$  y  $10^{-16}$  y la forma:

$$x^{(k)} = \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \end{pmatrix} = \begin{pmatrix} \frac{(x_1^{(k-1)})^2 + (x_2^{(k-1)})^2 + 8}{10} \\ \frac{x_1^{(k-1)} (x_2^{(k-1)})^2 + x_1^{(k-1)} + 8}{10} \end{pmatrix}$$

En tanto se busque cumplir  $|x^k - x^{(k-1)}|_\infty \leq \text{Tol}$ .

- c) Para implementar la iteración de punto fijo con aceleración de Gauss-Seidel también se tomó como punto inicial  $x^{(0)} = (0,0)$ , tolerancia de  $10^{-7}$ ,  $10^{-12}$  y  $10^{-16}$  pero la forma:

$$x^{(k)} = \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \end{pmatrix} = \begin{pmatrix} \frac{(x_1^{(k-1)})^2 + (x_2^{(k-1)})^2 + 8}{10} \\ \frac{x_1^{(k)} (x_2^{(k-1)})^2 + x_1^{(k)} + 8}{10} \end{pmatrix}$$

En tanto se busque cumplir  $|x^k - x^{(k-1)}|_\infty \leq \text{Tol}$ .

Para este caso, el número de iteraciones necesarias para alcanzar la tolerancia sí disminuye con respecto a la iteración normal, por lo que el método de Gauss-Seidel **sí acelera la convergencia**.

```
%Definimos las funciones
g1=@(x1,x2) (x1.^2+x2.^2+8)./10;
g2=@(x1,x2) (x1.*x2.^2+x1+8)./10;
%Tolerancia
tol=input('Tolerancia: ');

%SIN ACELERACIÓN Gauss-Seidel
%Evaluación inicial
xk=[0,0];
x=[g1(xk(1),xk(2)),g2(xk(1),xk(2))];
%Numero de evaluaciones
eval=1;
%Iteración de punto fijo
while(norm(x-xk,'inf')>tol)
    xk=x;
    x=[g1(xk(1),xk(2)),g2(xk(1),xk(2))];
    eval=eval+1;
end
%Mostramos los resultados
disp('Resultado (normal):'); disp(x);
disp('No. de iteraciones:'); disp(eval);

%CON ACELERACIÓN Gauss-Seidel
%Evaluación inicial
xk=[0,0]; x=zeros(1,2);
x(1)=g1(xk(1),xk(2));
x(2)=g2(x(1),xk(2));
%Numero de evaluaciones
eval=1;
%Iteración de punto fijo
while(norm(x-xk,'inf')>tol)
    xk=x;
    x(1)=g1(xk(1),xk(2));
    x(2)=g2(x(1),xk(2));
    eval=eval+1;
end
%Mostramos los resultados
disp('Resultado (Gauss-Seidel):'); disp(x);
disp('No. de iteraciones:'); disp(eval);
```

Command Window

```
>> ejercicio_10_1_5
Tolerancia: 1e-7
Resultado (normal):
      0.999999971858213      0.999999971858214

No. de iteraciones:
      18

Resultado (Gauss-Seidel):
      0.999999985228011      0.99999999175343

No. de iteraciones:
      15
```

Command Window

```
>> ejercicio_10_1_5
Tolerancia: 1e-12
Resultado (normal):
      0.999999999999528      0.999999999999528

No. de iteraciones:
      30

Resultado (Gauss-Seidel):
      0.99999999999959      0.999999999999771

No. de iteraciones:
      24
```

Command Window

```
>> ejercicio_10_1_5
Tolerancia: 1e-16
Resultado (normal):
      1      1

No. de iteraciones:
      41

Resultado (Gauss-Seidel):
      1      1

No. de iteraciones:
      33
```

### 10.1.6 El sistema no lineal

$$\begin{aligned} 5x_1^2 - x_2^2 &= 0 \\ x_2 - 0.25(\sin x_1 + \cos x_2) &= 0 \end{aligned}$$

tiene una solución cercana a  $\left(\frac{1}{4}, \frac{1}{4}\right)^t$ .

- Encuentre una función  $G$  y un conjunto  $D \subset \mathbb{R}^2$  de modo que  $G: D \rightarrow \mathbb{R}^2$  y  $G$  tenga un punto fijo único en  $D$ .
- Aplice la iteración funcional para aproximar la solución con una exactitud de  $10^{-5}$  en la norma infinito.
- ¿Acelera la convergencia el método de Gauss-Seidel?

- Despejemos  $x_1$  y  $x_2$  de las dos ecuaciones del sistema no lineal para formar  $G$  de la forma

$$G \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} g_1(x_1, x_2) \\ g_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} x_2/\sqrt{5} \\ 0.25(\sin x_1 + \cos x_2) \end{pmatrix}$$

Entonces tenemos el problema de punto fijo  $G \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ . Sabemos que la solución se encuentra cerca de  $\left(\frac{1}{4}, \frac{1}{4}\right)$ , definamos el dominio  $D = \{(x_1, x_2)^t: 0 \leq x_1, x_2 \leq \frac{1}{2}\}$ . Notemos que

$$|g_1(x_1, x_2)| = |x_2/\sqrt{5}| \leq \left|\frac{1}{2\sqrt{5}}\right| \approx 0.2236$$

$$|g_2(x_1, x_2)| = |0.25(\sin x_1 + \cos x_2)| \leq \left|0.25\left(\sin \frac{1}{2} + \cos \frac{1}{2}\right)\right| \approx 0.252172$$

Entonces  $g_1, g_2$  son acotadas en  $D$ , por tanto  $G(x_1, x_2) \in D$ . Ahora veamos que las derivadas parciales de  $g_1, g_2$  también son acotadas en  $D$ :

$$\left|\frac{\partial g_1}{\partial x_1}\right| = 0$$

$$\left|\frac{\partial g_1}{\partial x_2}\right| = \left|\frac{1}{\sqrt{5}}\right| \approx 0.4472136$$

$$\left|\frac{\partial g_2}{\partial x_1}\right| = |0.25 \cos x_1| \leq \left|0.25 \cos \frac{1}{2}\right| \approx 0.2499905$$

$$\left|\frac{\partial g_2}{\partial x_2}\right| = |-0.25 \sin x_2| \leq \left|-0.25 \sin \frac{1}{2}\right| \approx 0.0022$$

Luego  $g_1, g_2$  son continuas en  $D$ , por lo tanto  $G = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}$  es continua en  $D$  y además  $G(x_1, x_2) \in D$  siempre que  $(x_1, x_2) \in D$ . Esto implica que  $G$  tiene un único punto fijo en  $D$ .

- Se aproximó la solución aplicando iteración de punto fijo. Se consideró como punto inicial  $x^{(0)} = \left(\frac{1}{4}, \frac{1}{4}\right)$ , tolerancia de  $10^{-5}$  y la forma:

$$x^{(k)} = \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \end{pmatrix} = \begin{pmatrix} x_2^{(k-1)}/\sqrt{5} \\ 0.25(\sin x_1^{(k-1)} + \cos x_2^{(k-1)}) \end{pmatrix}$$

En tanto se busque cumplir  $|x^k - x^{(k-1)}|_{\infty} \leq \text{Tol}$ .

- c) Se aproximó la solución aplicando iteración de punto fijo, pero ahora usando Gauss-Seidel. Se consideró como punto inicial  $x^{(0)} = \left(\frac{1}{4}, \frac{1}{4}\right)$ , tolerancia de  $10^{-5}$  y la forma:

$$x^{(k)} = \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \end{pmatrix} = \begin{pmatrix} x_2^{(k-1)} / \sqrt{5} \\ 0.25 (\sin x_1^{(k)} + \cos x_2^{(k-1)}) \end{pmatrix}$$

En tanto se busque cumplir  $|x^k - x^{(k-1)}|_{\infty} \leq \text{Tol}$ .

De nuevo, el método de Gauss-Seidel converge más rápidamente a la solución (en menos iteraciones).

```
%Definimos las funciones
g1=@(x1,x2) x2./sqrt(5);
g2=@(x1,x2) 0.25.*(sin(x1)+cos(x2));
%Tolerancia
tol=input('Tolerancia: ');

%SIN ACELERACIÓN Gauss-Seidel
%Evaluación inicial
xk=[0.25,0.25];
x=[g1(xk(1),xk(2)),g2(xk(1),xk(2))];
%Numero de evaluaciones
eval=1;
%Iteración de punto fijo
while(norm(x-xk,'inf')>tol)
    xk=x;
    x=[g1(xk(1),xk(2)),g2(xk(1),xk(2))];
    eval=eval+1;
end
%Mostramos los resultados
disp('Resultado (normal):'); disp(x);
disp('No. de iteraciones:'); disp(eval);

%CON ACELERACIÓN Gauss-Seidel
%Evaluación inicial
xk=[0.25,0.25]; x=zeros(1,2);
x(1)=g1(xk(1),xk(2));
x(2)=g2(x(1),xk(2));
%Numero de evaluaciones
eval=1;
%Iteración de punto fijo
while(norm(x-xk,'inf')>tol)
    xk=x;
    x(1)=g1(xk(1),xk(2));
    x(2)=g2(x(1),xk(2));
    eval=eval+1;
end
%Mostramos los resultados
disp('Resultado (Gauss-Seidel):'); disp(x);
disp('No. de iteraciones:'); disp(eval);
```

Command Window

```
>> ejercicio_10_1_6
Tolerancia: 1e-5
Resultado (normal):
           0.12124079608339           0.271106226195433

No. de iteraciones:
           11

Resultado (Gauss-Seidel):
           0.121241873785078           0.271105152080845

No. de iteraciones:
           5
```

**10.1.7 Demuestre que  $G: D \subset \mathbb{R}^3 \rightarrow \mathbb{R}^3$  tiene un punto fijo en  $D$ . Aplique la iteración funcional para aproximar la solución con una exactitud de  $10^{-5}$  empleando  $\|\cdot\|_\infty$ .**

$$\text{a) } G \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \frac{\cos x_2 x_3 + 0.5}{3} \\ \frac{1}{25} \sqrt{x_1^2 + 0.3125} - 0.03 \\ -\frac{1}{20} e^{-x_1 x_2} - \frac{10\pi - 3}{60} \end{pmatrix}$$

$$D = \left\{ \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} : -1 \leq x_i \leq 1, i = 1, 2, 3 \right\}$$

Sean  $G \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \\ g_3 \end{pmatrix}$  y consideremos el problema de punto fijo  $G \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ . Notemos que

$$\left| \frac{\cos x_2 x_3 + 0.5}{3} \right| \leq \left| \frac{\cos 1 + 0.5}{3} \right| \leq \frac{1.5}{3} = \frac{1}{2}$$

$$\left| \frac{1}{25} \sqrt{x_1^2 + 0.3125} - 0.03 \right| \leq \left| \frac{1}{25} \sqrt{1.3125} - 0.03 \right| \approx 0.015826$$

$$\left| -\frac{1}{20} e^{-x_1 x_2} - \frac{10\pi - 3}{60} \right| \leq \left| -\frac{e}{20} - \frac{10\pi - 3}{60} \right| \approx 0.609513$$

Esto implica que  $G(x_1, x_2, x_3) \in D$  siempre que  $(x_1, x_2, x_3) \in D$ . Además, consideremos:

$$\left| \frac{\partial g_1}{\partial x_1} \right| = 0$$

$$\left| \frac{\partial g_1}{\partial x_2} \right| = \left| -\frac{x_3 \sin x_2 x_3}{3} \right| \leq \left| -\frac{\sin 1}{3} \right| \approx 0.2805$$

$$\left| \frac{\partial g_1}{\partial x_3} \right| = \left| -\frac{x_2 \sin x_2 x_3}{3} \right| \leq \left| -\frac{\sin 1}{3} \right| \approx 0.2805$$

$$\left| \frac{\partial g_2}{\partial x_1} \right| = \left| \frac{x_1}{25} (x_1^2 + 0.3125)^{-\frac{1}{2}} \right| \leq \left| \frac{1}{25} (1 + 0.3125)^{-\frac{1}{2}} \right| \approx 0.035$$

$$\left| \frac{\partial g_2}{\partial x_2} \right| = 0$$

$$\left| \frac{\partial g_2}{\partial x_3} \right| = 0$$

$$\left| \frac{\partial g_3}{\partial x_1} \right| = \left| \frac{x_2}{20} e^{-x_1 x_2} \right| \leq \left| \frac{1}{20} e^{-1} \right| \approx 0.0184$$

$$\left| \frac{\partial g_3}{\partial x_2} \right| = \left| \frac{x_1}{20} e^{-x_1 x_2} \right| \leq \left| \frac{1}{20} e^{-1} \right| \approx 0.0184$$

$$\left| \frac{\partial g_3}{\partial x_3} \right| = 0$$

Todas las derivadas parciales están acotadas para puntos  $(x_1, x_2, x_3) \in D$ , por lo tanto  $G$  es continua en  $D$  y como  $G(x_1, x_2, x_3) \in D$  entonces  $G$  tiene un único punto fijo en  $D$ . Se utilizó la misma rutina anterior para encontrar el punto fijo con punto inicial  $(0,0,0)$ .

```
Command Window
>> ejercicio_10_1_7a
Tolerancia: 1e-5
Resultado (normal):
                0.5                0                -0.523598777376017

No. de iteraciones:
                4

Resultado (Gauss-Seidel):
                0.5                0                -0.523598775598299

No. de iteraciones:
                2
```

```
%Definimos las funciones
g1=@(x2,x3) (cos(x2*x3)+0.5)/3;
g2=@(x1) (sqrt(x1^2+0.3125)/25)-0.03;
g3=@(x1,x2) -(exp(-x1*x2)/20)-((10*pi-3)/60);
%Tolerancia
tol=input('Tolerancia: ');

%SIN ACELERACIÓN Gauss-Seidel
xk=[0,0,0];
x=[g1(xk(2),xk(3)),g2(xk(1)),g3(xk(1),xk(2))];
eval=1;
while(norm(x-xk,'inf')>tol)
    xk=x;
    x=[g1(xk(2),xk(3)),g2(xk(1)),g3(xk(1),xk(2))];
    eval=eval+1;
end
disp('Resultado (normal):'); disp(x);
disp('No. de iteraciones:'); disp(eval);

%CON ACELERACIÓN Gauss-Seidel
xk=[0,0,0]; x=zeros(1,3);
x(1)=g1(xk(2),xk(3));
x(2)=g2(x(1));
x(3)=g3(x(1),x(2));
eval=1;
while(norm(x-xk,'inf')>tol)
    xk=x;
    x(1)=g1(xk(2),xk(3));
    x(2)=g2(x(1));
    x(3)=g3(x(1),x(2));
    eval=eval+1;
end
disp('Resultado (Gauss-Seidel):'); disp(x);
disp('No. de iteraciones:'); disp(eval);
```



$$\text{b) } G \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 - \cos x_1 x_2 x_3 \\ 1 - (1 - x_1)^{\frac{1}{4}} - 0.05x_3^2 + 0.15x_3 \\ x_1^2 + 0.1x_2^2 - 0.01x_2 + 1 \end{pmatrix},$$

$$D = \left\{ \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} : -0.1 \leq x_1 \leq 0.1, -0.1 \leq x_2 \leq 0.3, 0.5 \leq x_3 \leq 1.1 \right\}$$

Sean  $G \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \\ g_3 \end{pmatrix}$  y consideremos el problema de punto fijo  $G \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ . Notemos que

$$|1 - \cos x_1 x_2 x_3| \leq |1 - \cos(0.1 \times 0.3 \times 1.1)| \approx 0.0005445$$

$$0 \leq 1 - (1 - x_1)^{\frac{1}{4}} - 0.05x_3^2 + 0.15x_3 \leq 1 - (1 - 0.1)^{\frac{1}{4}} - 0.05(1.1)^2 + 0.15(1.1) \approx 0.136$$

$$0.5 \leq x_1^2 + 0.1x_2^2 - 0.01x_2 + 1 \leq (0.1)^2 + 0.1(0.3)^2 - 0.01(0.3) + 1 \approx 1.016$$

Esto implica que  $G(x_1, x_2, x_3) \in D$  siempre que  $(x_1, x_2, x_3) \in D$ . Además, consideremos:

$$\left| \frac{\partial g_1}{\partial x_1} \right| = |x_2 x_3 \sin x_1 x_2 x_3| \leq (0.3 \times 1.1)(1) = 0.33$$

$$\left| \frac{\partial g_1}{\partial x_2} \right| = |x_1 x_3 \sin x_1 x_2 x_3| \leq (0.1 \times 1.1)(1) = 0.11$$

$$\left| \frac{\partial g_1}{\partial x_3} \right| = |x_1 x_2 \sin x_1 x_2 x_3| \leq (0.1 \times 0.3)(1) = 0.03$$

$$\left| \frac{\partial g_2}{\partial x_1} \right| = \left| -\frac{1}{4}(1 - x_1)^{-\frac{3}{4}} \right| \leq \left| -\frac{1}{4}(1 - 0.1)^{-\frac{3}{4}} \right| \approx 0.27$$

$$\left| \frac{\partial g_2}{\partial x_2} \right| = 0$$

$$\left| \frac{\partial g_2}{\partial x_3} \right| = |-0.1x_3 + 0.15| \leq |-0.1(1.1) + 0.15| = 0.04$$

$$\left| \frac{\partial g_2}{\partial x_1} \right| = |2x_1| \leq |2(0.1)| = 0.2$$

$$\left| \frac{\partial g_2}{\partial x_2} \right| = |2(0.1)x_2 - 0.01| \leq |0.2(0.3) - 0.01| = 0.59$$

$$\left| \frac{\partial g_2}{\partial x_3} \right| = 0$$

Todas las derivadas parciales están acotadas para puntos  $(x_1, x_2, x_3) \in D$ , por lo tanto  $G$  es continua en  $D$  y como  $G(x_1, x_2, x_3) \in D$  entonces  $G$  tiene un único punto fijo en  $D$ . Se utilizó la misma rutina anterior para encontrar el punto fijo con punto inicial  $(0, 0.1, 0.8)$ .

```
Command Window
>> ejercicio_10_1_7c
Tolerancia: 1e-5
Resultado (normal):
               0               0.1       0.999999947197213

No. de iteraciones:
               4

Resultado (Gauss-Seidel):
               0       0.0999999973598605       0.99999999973599

No. de iteraciones:
               3
```

%Definimos las funciones

```
g1=@(x1,x2,x3) 1-cos(x1*x2*x3);
g2=@(x1,x3) 1-(1-x1)^(1/4)-0.05*x3^2+0.15*x3;
g3=@(x1,x2) x1^2+0.1*x2^2-0.01*x2+1;
```

%Tolerancia

```
tol=input('Tolerancia: ');
```

%SIN ACELERACIÓN Gauss-Seidel

```
xk=[0,0.1,0.8];
x=[g1(xk(1),xk(2),xk(3)),g2(xk(1),xk(3)),g3(xk(1),xk(2))];
eval=1;
while(norm(x-xk,'inf')>tol)
    xk=x;
    x=[g1(xk(1),xk(2),xk(3)),g2(xk(1),xk(3)),g3(xk(1),xk(2))];
    eval=eval+1;
end
%Mostramos los resultados
disp('Resultado (normal):'); disp(x);
disp('No. de iteraciones:'); disp(eval);
```

%CON ACELERACIÓN Gauss-Seidel

```
xk=[0,0.1,0.8]; x=zeros(1,3);
x(1)=g1(xk(1),xk(2),xk(3));
x(2)=g2(x(1),xk(3));
x(3)=g3(x(1),x(2));
eval=1;
while(norm(x-xk,'inf')>tol)
    xk=x;
    x(1)=g1(xk(1),xk(2),xk(3));
    x(2)=g2(x(1),xk(3));
    x(3)=g3(x(1),x(2));
    eval=eval+1;
end
%Mostramos los resultados
disp('Resultado (Gauss-Seidel):'); disp(x);
disp('No. de iteraciones:'); disp(eval);
```

### 10.2.1 Mediante el método de Newton con $x^{(0)} = 0$ calcule $x^{(2)}$ para los siguientes sistemas no lineales

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

c) 
$$\begin{aligned} 3x_1 - \cos x_2x_3 - \frac{1}{2} &= 0 \\ 4x_1^2 - 625x_2^2 + 2x_2 - 1 &= 0 \\ e^{-x_1x_2} + 20x_3 + \frac{10\pi-3}{3} &= 0 \end{aligned}$$

El método de Newton para sistemas de ecuaciones no lineales multivariantes aproxima la solución de la forma  $F(x) = 0$  dada una aproximación inicial  $x^{(0)}$ . Está dado por el siguiente algoritmo:

```
% ENTRADA:      n ecuaciones e incógnitas
%               aproximación inicial x0
%               tolerancia
%               número máximo de iteraciones
% SALIDA:       solución aproximada x
%               o bien, mensaje de error
%
% 1) Inicialmente no hemos hecho iteraciones, k=0 y xk=x0
% 2) Mientras no se exceda el máximo de iteraciones, k<=MaxItera
%   3) Calcular el valor de F(xk)
%   4) Calcular el valor de J(xk) con J matriz jacobiana del sistema
%   5) Resolver el sistema lineal J(xk) y=-F(xk) para y
%   6) Hacer xk=xk+y
%   7) Si norm(y,'inf')<tolerancia
%   8) break, finalización correcta
%   9) Hacer k=k+1, hemos iterado una vez más
% 10) Mostrar el último resultado, nuestra mejor aproximación
% 11) Mostrar si se excedieron las iteraciones o se llegó al resultado
```

El mayor problema para implementar este algoritmo es la evaluación de la matriz Jacobiana del sistema en cada iteración. Para solventar esta dificultad se implementó un script en Matlab utilizando la Symbolic Math Toolbox, esta poderosa herramienta cuenta con las funciones:

- `jacobian`. Devuelve la matriz Jacobiana simbólica de un conjunto de ecuaciones en función de variables simbólicas dadas.
- `syms`. Declaración de variables simbólicas.
- `symvar`. Devuelve el listado de variables simbólicas de otra expresión simbólica.
- `subs`. Sustituye escalares u variables simbólicas en las variables simbólicas de una expresión.
- `double`. Proporciona el valor numérico de una expresión simbólica en doble precisión.

En resumidas cuentas, el algoritmo implementado permite ejecutar el algoritmo del método de Newton satisfactoriamente. Este script será modificado para ejercicios posteriores siguiendo la misma idea.

Volviendo al problema requerido, para obtener  $x^{(2)}$  se hizo que el número máximo de iteraciones fuera precisamente 2, de esta manera el algoritmo se detiene en la aproximación requerida y la devuelve.

```
Command Window

>> ejercicio_10_2_1a
Tolerancia:1e-15
Máximo num de iteraciones:2
Número máximo de iteraciones alcanzado
    2

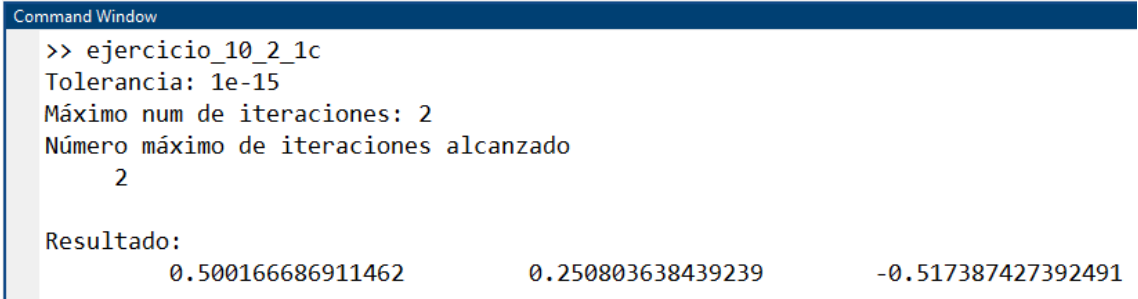
Resultado:
           0.495893610552932           1.98342347419233
```

```
%Método de Newton multivariable
%PARTE SIMBOLICA
%Definir variables simbolicas para MATLAB (según se requiera)
syms sx1 sx2 sf1 sf2
%Definimos las funciones SIMBOLICAS sf1,sf2,...,sfn
sf1=4*sx1^2-20*sx1+(sx2^2)/4+8;
sf2=(sx1*sx2^2)/2+2*sx1-5*sx2+8;
%Definimos nuestra función simbólica vectorial
sF=[sf1;sf2];
%Jacobian(F,X) devuelve la matriz jacobiana simbólica
sJ=jacobian(sF,symvar(sF));

%ALGORITMO DE NEWTON
%Punto inicial (introducir valores en orden alfabetico simbolico)
x=[0,0];
%Tolerancia
tol=input('Tolerancia:');
%Variables de iteración
max_iteraciones=input('Máximo num de iteraciones:');
iteraciones=0;

%Algoritmo
while iteraciones<max_iteraciones
    %Aumentamos número de iteraciones
    iteraciones=iteraciones+1;
    %Evaluamos la función -F en el x actual, obteniendo -F(x)
    F=-double(subs(sF,symvar(sF),x));
    %Evaluamos el jacobiano J en el x actual, obteniendo J(x)
    J=double(subs(sJ,symvar(sJ),x));
    %Resolvemos el SistLin J(x) y = -F(x) para y
    y=mldivide(J,F);
    %Hacemos x=x+y, es decir, actualizamos nuestra aprox
    x=x+y';
    %Vemos si hemos cumplido la tolerancia de error
    if (norm(y,'inf')<=tol)
        %Terminamos, imprimimos resultado y salimos
        break;
```

```
    end
    %Sino, seguimos calculado
end
%Mensajes de resultado
if(iteraciones>=max_iteraciones)
    disp('Número máximo de iteraciones alcanzado');
    disp(iteraciones);
end
disp('Resultado:'); disp(x);
```



```
Command Window
>> ejercicio_10_2_1c
Tolerancia: 1e-15
Máximo num de iteraciones: 2
Número máximo de iteraciones alcanzado
    2

Resultado:
    0.500166686911462    0.250803638439239   -0.517387427392491
```

```
%Método de Newton multivariable
%PARTE SIMBOLICA
syms sx1 sx2 sx3 sf1 sf2 sf3
sf1=3*sx1-cos(sx2*sx3)-1/2;
sf2=4*sx1^2-625*sx2^2+2*sx2-1;
sf3=exp(-sx1*sx2)+20*sx3+((10*pi-3)/3);
sF=[sf1;sf2;sf3];
sJ=jacobian(sF,symvar(sF));

%ALGORITMO DE NEWTON
x=[0,0,0];
tol=input('Tolerancia: ');
max_iteraciones=input('Máximo num de iteraciones: ');
iteraciones=0;
while iteraciones<max_iteraciones
    iteraciones=iteraciones+1;
    F=-double(subs(sF,symvar(sF),x));
    J=double(subs(sJ,symvar(sJ),x));
    y=mldivide(J,F);
    x=x+y';
    if (norm(y,'inf')<=tol)
        break;
    end
end
%Mensajes de resultado
if(iteraciones>=max_iteraciones)
    disp('Número máximo de iteraciones alcanzado');
    disp(iteraciones);
end
disp('Resultado:'); disp(x);
```

**10.2.3 Use el método de Newton para encontrar una solución a los siguientes sistemas no lineales con la aproximación inicial dada. Itere hasta que  $\|x^{(k)} - x^{(k-1)}\|_{\infty} < 10^{-6}$ .**

$$\text{a) } \begin{cases} 3x_1^2 - x_2^2 = 0 \\ 3x_1x_2^2 - x_1^3 - 1 = 0 \end{cases}, x^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\text{c) } \begin{cases} x_1^3 + x_1^2x_2 - x_1x_3 + 6 = 0 \\ e^{x_1} + e^{x_2} - x_3 = 0 \\ x_2^2 - 2x_1x_3 = 4 \end{cases}, x^{(0)} = \begin{pmatrix} -1 \\ -2 \\ 1 \end{pmatrix}$$

Para este problema se modificó el script anterior. Para cada caso se aplicó aproximación inicial requerida, la tolerancia de  $10^{-6}$  y un número máximo de iteraciones grande para que se pudiera obtener la respuesta adecuadamente.

Para resolver a) sucede algo interesante. La aproximación inicial hace que la matriz Jacobiana del sistema sea singular, por lo que desde la segunda aproximación  $x^{(1)}$  el resultado es prácticamente incorrecto. Una manera de solucionar esto es iniciando con otra aproximación inicial, por ejemplo  $x^{(0)} = \begin{pmatrix} 0.001 \\ 0.001 \end{pmatrix}$ , esto hace que la convergencia a la solución sea lenta pero no haya indeterminaciones en la ejecución.

Para resolver c) no se encontraron problemas.

```
Command Window
>> ejercicio_10_2_3a
Aproximación inicial:
                        0.001                0.001

Tolerancia: 1e-6
Máximo num de iteraciones: 200
Iteraciones:
    35

Resultado:
        0.5                0.866025403784439
```

%PARTE SIMBOLICA

```
syms sx1 sx2 sf1 sf2
sf1=3*sx1^2-sx2^2;
sf2=3*sx1*sx2^2-sx1^3-1;
sF=[sf1;sf2];
sJ=jacobian(sF,symvar(sF));
```

%ALGORITMO DE NEWTON

%Punto inicial (introducir valores en orden alfabetico simbolico)

```
x=[0.001,0.001];
disp('Aproximación inicial:'); disp(x);
tol=input('Tolerancia: ');
max_iteraciones=input('Máximo num de iteraciones: ');
iteraciones=0;
```

```
while iteraciones<max_iteraciones
    iteraciones=iteraciones+1;
    F=-double(subs(sF,symvar(sF),x));
    J=double(subs(sJ,symvar(sJ),x));
    y=mldivide(J,F);
    x=x+y';
    if (norm(y,'inf')<=tol)
        break;
    end
end
%Mensajes de resultado
if(iteraciones>=max_iteraciones)
    disp('Número máximo de iteraciones alcanzado');
end
disp('Iteraciones: '); disp(iteraciones);
disp('Resultado:'); disp(x);
```

```
Command Window
>> ejercicio_10_2_3c
Aproximación inicial:
    -1    -2     1

Tolerancia: 1e-6
Máximo num de iteraciones: 100
Iteraciones:
     5

Resultado:
    -1.45604279595534    -1.66423046608154     0.422493404446532
```

#### %PARTE SIMBOLICA

```
syms sx1 sx2 sx3 sf1 sf2 sf3
sf1=sx1^3+sx1^2*sx2+-sx1*sx3+6;
sf2=exp(sx1)+exp(sx2)-sx3;
sf3=sx2^2-2*sx1*sx3-4;
sF=[sf1;sf2;sf3];
sJ=jacobian(sF,symvar(sF));
```

#### %ALGORITMO DE NEWTON

```
x=[-1,-2,1]; disp('Aproximación inicial: '); disp(x);
tol=input('Tolerancia: ');
max_iteraciones=input('Máximo num de iteraciones: ');
iteraciones=0;
```

#### %Algoritmo

```
while iteraciones<max_iteraciones
    iteraciones=iteraciones+1;
    F=-double(subs(sF,symvar(sF),x));
    J=double(subs(sJ,symvar(sJ),x));
    y=mldivide(J,F);
    x=x+y';
    if (norm(y,'inf')<=tol)
        break;
    end
end
%Mensajes de resultado
if(iteraciones>=max_iteraciones)
    disp('Número máximo de iteraciones alcanzado');
end
disp('Iteraciones: '); disp(iteraciones);
disp('Resultado:'); disp(x);
```



### 10.2.5 El sistema no lineal

$$\begin{aligned}3x_1 - \cos x_2 x_3 - \frac{1}{2} &= 0 \\x_1^2 - 625x_2^2 - \frac{1}{4} &= 0 \\e^{-x_1 x_2} + 20x_3 + \frac{10\pi - 3}{3} &= 0\end{aligned}$$

**Tiene una matriz jacobiana singular en la solución. Aplique el método de Newton con  $x^{(0)} = (1, 1, -1)^t$ . Observe que la convergencia puede ser lenta o no ocurrir después de realizar una cantidad razonable de iteraciones.**

En este problema se seleccionó una tolerancia máxima en términos de la precisión del programa,  $10^{-16}$  y se hizo variar el número máximo de iteraciones en tres valores concretos: 10, 50 y 100. Efectivamente, la convergencia hacia la solución resulta ser muy lenta, especialmente en el cálculo de la segunda entrada de la solución.

En la iteración 54 se logra satisfacer la tolerancia impuesta, se puede notar que la solución tiende al punto  $\left(\frac{1}{2}, 0, -\frac{\pi}{6}\right)$  (la validez de esta afirmación se ha comprobado sustituyendo dichos valores en el sistema, encontrando que son solución) donde se ha alcanzado una precisión de prácticamente 15 dígitos significativos, aunque no quita el hecho de que la convergencia sigue siendo increíblemente lenta.

Command Window

```
>> ejercicio_10_2_5
Aproximación inicial:
    1    1   -1

Tolerancia: 1e-16
Máximo num de iteraciones: 10
Número máximo de iteraciones alcanzado
Iteraciones:
    10

Resultado:
    0.499999999983738    0.000976459487446986    -0.523574364114985
```

Command Window

```
>> ejercicio_10_2_5
Aproximación inicial:
    1    1   -1

Tolerancia: 1e-16
Máximo num de iteraciones: 50
Número máximo de iteraciones alcanzado
Iteraciones:
    50

Resultado:
    0.499999999999996    8.88084724273641e-16    -0.523598775598293
```

Command Window

```
>> ejercicio_10_2_5
Aproximación inicial:
    1    1   -1

Tolerancia: 1e-16
Máximo num de iteraciones: 100
Iteraciones:
    54

Resultado:
    0.499999999999996    5.55052952671025e-17    -0.523598775598293
```

%Método de Newton multivariable

%PARTE SIMBOLICA

%Definir variables simbolicas para MATLAB (según se requiera)

syms *sx1 sx2 sx3 sf1 sf2 sf3*

%Definimos las funciones SIMBOLICAS *sf1,sf2,...,sfn*

*sf1*=3\**sx1*-cos(*sx2*\**sx3*)-1/2;

*sf2*=*sx1*^2-625\**sx2*^2-1/4;

*sf3*=exp(-*sx1*\**sx2*)+20\**sx3*+((10\*pi-3)/3);

%Definimos nuestra función simbolica vectorial

*sF*=[*sf1*;*sf2*;*sf3*];

%Jacobian(*F,X*) devuelve la matriz jacobiana simbolica

*sJ*=jacobian(*sF*,symvar(*sF*));

%ALGORITMO DE NEWTON

%Punto inicial (introducir valores en orden alfabetico simbolico)

*x*=[1,1,-1]; disp('Aproximación inicial: '); disp(*x*);

%Tolerancia

*tol*=input('Tolerancia: ');

%Variables de iteración

*max\_iteraciones*=input('Máximo num de iteraciones: ');

*iteraciones*=0;

%Algoritmo

while *iteraciones*<*max\_iteraciones*

    %Aumentamos número de iteraciones

*iteraciones*=*iteraciones*+1;

    %Evalúamos la función -F en el x actual, obteniendo -F(x)

*F*=-double(subs(*sF*,symvar(*sF*),*x*));

    %Evalúamos el jacobiano J en el x actual, obteniendo J(x)

*J*=double(subs(*sJ*,symvar(*sJ*),*x*));

    %Resolvemos el SistLin J(x) y = -F(x) para y

*y*=mldivide(*J*,*F*);

    %Hacemos *x*=*x*+*y*, es decir, actualizamos nuestra aprox

*x*=*x*+*y*;

    %Vemos si hemos cumplido la tolerancia de error

    if (norm(*y*, 'inf')<=*tol*)

        %Terminamos, imprimimos resultado y salimos

        break;

    end

    %Sino, seguimos calculado

end

%Mensajes de resultado

if(*iteraciones*>=*max\_iteraciones*)

    disp('Número máximo de iteraciones alcanzado');

end

disp('Iteraciones: '); disp(*iteraciones*);

disp('Resultado:'); disp(*x*);

10.2.9 C. Chiarella, W. Charlton y A. W. Roberts [Optimum Chute Profiles in Gravity Flow of Granular Materials: A Discrete Segment Solution Method], al calcular la forma de un sumidero de descarga de flujo por gravedad que reducirá al máximo el tiempo de tránsito de las partículas granulares descargadas, resuelven la siguiente ecuación por medio del método de Newton:

$$f_n(\theta_1, \theta_2, \dots, \theta_N) = \frac{\sin \theta_{n+1}}{v_{n+1}} (1 - \mu w_{n+1}) - \frac{\sin \theta_n}{v_n} (1 - \mu w_n) = 0,$$

$$n = 1, 2, \dots, N - 1$$

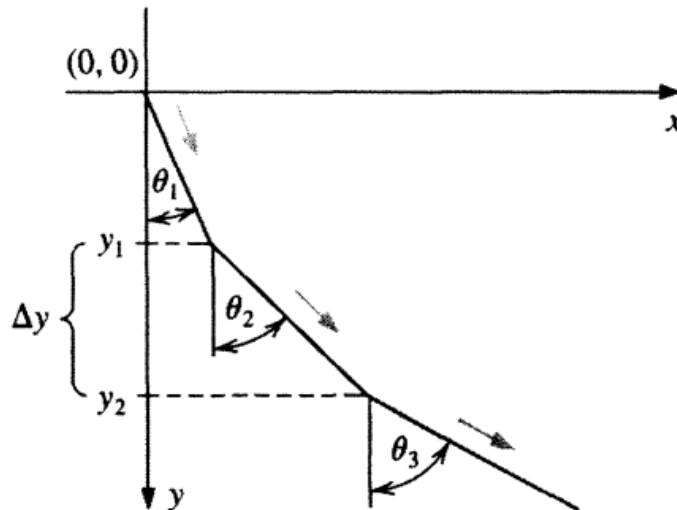
$$f_N(\theta_1, \theta_2, \dots, \theta_N) = \Delta y \sum_{i=1}^N \tan \theta_i - X = 0$$

Donde

$$v_n^2 = v_0^2 + 2gn\Delta y - 2\mu\Delta y \sum_{j=1}^n \frac{1}{\cos \theta_j}, \quad n = 1, 2, \dots, N$$

$$w_n = -\Delta y v_n \sum_{i=1}^N \frac{1}{v_i \cos \theta_i}, \quad n = 1, 2, \dots, N$$

La constante  $v_0$  es la velocidad inicial del material granular,  $X$  es la coordenada  $x$  del extremo final del sumidero,  $\mu$  es la fuerza de fricción,  $N$  es el número de segmentos del sumidero y  $g$  es la constante gravitacional  $32.17 \frac{ft}{s^2}$ . La variable  $\theta_i$  es el ángulo del  $i$ -ésimo segmento del sumidero respecto a la vertical (como se aprecia en la figura siguiente) y  $v_i$  es la velocidad de la partícula en el  $i$ -ésimo segmento del sumidero. Resuelva  $f_n$ ,  $n = 1, 2, \dots, N$  para  $\theta = (\theta_1, \theta_2, \dots, \theta_N)^t$  con  $\mu = 0$ ,  $X = 2$ ,  $\Delta y = 0.2$ ,  $N = 20$  y  $v_0 = 0$ , donde los valores de  $v_n$  y  $w_n$  pueden obtenerse de las expresiones dadas. Itere hasta que  $\|\theta^{(k)} - \theta^{(k-1)}\|_\infty < 10^{-2}$ .



Este interesante problema pone a prueba el programa desarrollado y la habilidad de introducir una cantidad deseada ( $N$ ) de variables simbólicas.

Primero veamos que, dado  $\mu = 0$ ,  $\Delta y = 0.2$ ,  $g = 32.17 \frac{ft}{s^2}$  y  $v_0 = 0$  las expresiones de las velocidades de las partículas en el  $n$ -ésimo sumidero se reducen a

$$v_n^2 = v_0^2 + 2gn\Delta y - 2\mu\Delta y \sum_{j=1}^n \frac{1}{\cos \theta_j} = 2gn\Delta y$$

$$\therefore v_n^2 = 2gn\Delta y = 12.868 \times n \left[ \frac{ft}{s^2} \right], \quad n = 1, 2, \dots, N$$

Mientras que  $w_n = -\Delta y v_n \sum_{i=1}^N \frac{1}{v_i \cos \theta_i} = -0.2v_n \sum_{i=1}^{20} \frac{1}{v_i \cos \theta_i}$ ,  $n = 1, 2, \dots, N$ .

Tenemos entonces que las expresiones para las  $f_n$ ,  $n = 1, 2, \dots, N$  se reducen, tomando en cuenta todas las consideraciones dadas en el problema, a:

$$f_n(\theta_1, \theta_2, \dots, \theta_N) = \frac{\sin \theta_{n+1}}{v_{n+1}} (1 - \mu w_{n+1}) - \frac{\sin \theta_n}{v_n} (1 - \mu w_n) = \frac{\sin \theta_{n+1}}{v_{n+1}} - \frac{\sin \theta_n}{v_n}$$

$$= \frac{\sin \theta_{n+1}}{\sqrt{12.868(n+1)}} - \frac{\sin \theta_n}{\sqrt{12.868(n)}} = 0, \quad n = 1, 2, \dots, N-1$$

$$f_N(\theta_1, \theta_2, \dots, \theta_N) = \Delta y \sum_{i=1}^N \tan \theta_i - X = 0.2 \sum_{i=1}^{20} \tan \theta_i - 2 = 0$$

Estas son las ecuaciones que debemos resolver para  $\theta_1, \theta_2, \dots, \theta_N$ . De esta manera se modificó la entrada de las variables simbólicas en el script para admitir  $N$  ecuaciones, a pesar de elegir solamente  $N = 20$ .

Se seleccionó la tolerancia de  $10^{-2}$  y como aproximación inicial a  $(\theta_1, \theta_2, \dots, \theta_N)^t = (1, 1, \dots, 1)^t$ . Se obtuvo la solución deseada en 5 iteraciones y se muestra en la Tabla 1. Como dato adicional, para una tolerancia de  $10^{-16}$  se requieren solo 8 iteraciones (recordemos que estamos bajo ciertas condiciones). Los resultados obtenidos son los ángulos a los cuales debemos colocar los  $N$  sumideros para que la descarga de material granular sea máxima por acción de la gravedad (bajo estas condiciones).

Tabla 1. Resultados obtenidos para  $\theta_n$ ,  $n = 1, 2, \dots, 20$  con el método de Newton multivariable con tolerancia de  $10^{-2}$  y con aproximación inicial  $(1, 1, \dots, 1)^t$ .

$n$	1	2	3	4	5
$\theta_n$	0.14062	0.19954	0.24522	0.28413	0.31878
$n$	6	7	8	9	10
$\theta_n$	0.35045	0.3799	0.40763	0.43398	0.4592
$n$	11	12	13	14	15
$\theta_n$	0.48348	0.50697	0.5298	0.55205	0.57382
$n$	16	17	18	19	20
$\theta_n$	0.59516	0.61615	0.63683	0.65726	0.67746

```

%Método de Newton multivariable
%PARTE SIMBOLICA
%Número de sumideros
N=20;
%Definir N variables simbolicas para MATLAB
sT=sym('st',[1,N]);
sF=sym('sf',[1,N]);
%Definimos las funciones SIMBOLICAS sf1,sf2,...,sf(N-1)
%en función de st1,st2,...,stn
for i=1:N-1
    sF(i)=(sin(sT(i+1)))/sqrt(12.868*(i+1))...
        -(sin(sT(i)))/sqrt(12.868*(i));
end
%Definimos la función SIMBOLICA sfN
sF(N)=-2;
for i=1:N
    sF(N)=sF(N)+0.2*tan(sT(i));
end
%Jacobian(F,X) devuelve la matriz jacobiana simbolica
sJ=jacobian(sF,sT);

%ALGORITMO DE NEWTON
%Punto inicial (se pide que sea [1,1,...,1])
x=ones(1,N);
%Tolerancia
tol=input('Tolerancia: ');
%Variables de iteración
max_iteraciones=input('Máximo num de iteraciones: ');
iteraciones=0;
%Algoritmo
while iteraciones<max_iteraciones
    %Aumentamos número de iteraciones
    iteraciones=iteraciones+1;
    %Evaluamos la función -F en el x actual, obteniendo -F(x)
    F=-double(subs(sF,sT,x));
    %Evaluamos el jacobiano J en el x actual, obteniendo J(x)
    J=double(subs(sJ,sT,x));
    %Resolvemos el SistLin J(x) y = -F(x) para y
    y=mldivide(J,F');
    %Hacemos x=x+y, es decir, actualizamos nuestra aprox
    x=x+y';
    %Vemos si hemos cumplido la tolerancia de error
    if (norm(y,'inf')<=tol)
        %Terminamos, imprimimos resultado y salimos
        break;
    end
    %Sino, seguimos calculado
end
%Mensajes de resultado
if(iteraciones>=max_iteraciones)
    disp('Número máximo de iteraciones alcanzado');
end
disp('Iteraciones:'); disp(iteraciones);
disp('Resultado:'); disp(x');

```

```
Command Window
>> ejercicio_10_2_9
Tolerancia: 1e-2
Máximo num de iteraciones: 100
Iteraciones:
    5

Resultado:
    0.14062
    0.19954
    0.24522
    0.28413
    0.31878
    0.35045
    0.3799
    0.40763
    0.43398
    0.4592
    0.48348
    0.50697
    0.5298
    0.55205
    0.57382
    0.59516
    0.61615
    0.63683
    0.65726
    0.67746
```

10.2.11 Un interesante experimento biológico (véase [Thermal tolerance and acclimation of two species of Hydra]) es la determinación de la temperatura máxima del agua,  $X_M$ , en la que varias especies de hidra pueden sobrevivir sin que su esperanza de vida disminuya. Una forma de resolver este problema consiste en aplicar un ajuste ponderado de mínimos cuadrados de la forma  $f(x) = \frac{a}{(x-b)^c}$  a un conjunto de datos experimentales. Los valores  $x$  de los datos se refieren a la temperatura del agua, la constante  $b$  es la asíntota de la gráfica de  $f$  y, por tanto, es una aproximación a  $X_M$ .

Resuelva el siguiente sistema no lineal para  $a, b, c$

$$\begin{aligned} a &= \sum_{i=1}^n \frac{w_i y_i}{(x_i - b)^c} / \sum_{i=1}^n \frac{1}{(x_i - b)^{2c}} \\ 0 &= \sum_{i=1}^n \frac{w_i y_i}{(x_i - b)^c} \cdot \sum_{i=1}^n \frac{1}{(x_i - b)^{2c+1}} - \sum_{i=1}^n \frac{w_i y_i}{(x_i - b)^{c+1}} \cdot \sum_{i=1}^n \frac{1}{(x_i - b)^{2c}} \\ 0 &= \sum_{i=1}^n \frac{w_i y_i}{(x_i - b)^c} \cdot \sum_{i=1}^n \frac{\ln(x_i - b)}{(x_i - b)^{2c}} - \sum_{i=1}^n \frac{w_i y_i \ln(x_i - b)}{(x_i - b)^c} \cdot \sum_{i=1}^n \frac{1}{(x_i - b)^{2c}} \end{aligned}$$

Con los siguientes datos y utilizando los pesos  $w_i = \ln y_i$

$i$	1	2	3	4
$y_i$	2.40	3.80	4.75	21.60
$x_i$	31.8	31.5	31.2	30.2

Las soluciones hacen que  $\sum_{i=1}^n \left( w_i y_i - \frac{a}{(x_i - b)^c} \right)^2$  disminuya al mínimo.

Al igual que el anterior, este problema representa un reto para resolver el sistema lineal utilizando nuestro programa simbólico.

Como tenemos 4 datos,  $n = 4$  y al considerar  $w_i = \ln y_i$  el sistema a resolver se reduce a

$$\begin{aligned} a &= \sum_{i=1}^4 \frac{y_i \ln y_i}{(x_i - b)^c} / \sum_{i=1}^4 \frac{1}{(x_i - b)^{2c}} \\ 0 &= \sum_{i=1}^n \frac{y_i \ln y_i}{(x_i - b)^c} \cdot \sum_{i=1}^n \frac{1}{(x_i - b)^{2c+1}} - \sum_{i=1}^n \frac{y_i \ln y_i}{(x_i - b)^{c+1}} \cdot \sum_{i=1}^n \frac{1}{(x_i - b)^{2c}} \\ 0 &= \sum_{i=1}^n \frac{y_i \ln y_i}{(x_i - b)^c} \cdot \sum_{i=1}^n \frac{\ln(x_i - b)}{(x_i - b)^{2c}} - \sum_{i=1}^n \frac{y_i \ln y_i \ln(x_i - b)}{(x_i - b)^c} \cdot \sum_{i=1}^n \frac{1}{(x_i - b)^{2c}} \end{aligned}$$

Observemos que la primera ecuación nos proporciona explícitamente el valor de  $a$ , mientras que las otras dos forman un sistema no lineal para  $b$  y  $c$ . La estrategia para seguir es implementar las tres ecuaciones simbólicamente pero solo resolver con el método de Newton multivariable a las últimas dos para obtener  $b$  y  $c$ . Una vez hecho esto obtendremos  $a$ .



Se eligió como aproximación inicial para resolver  $b$  y  $c$  el punto  $(26.8, 8.3)^t$  con una tolerancia de  $10^{-12}$  y un máximo de iteraciones de 50. El resultado se alcanza en 9 iteraciones, para obtener finalmente los valores de  $a, b, c$  dados por

$$\begin{aligned}a &= 2,217,952.23196394 \\b &= 26.7702106083026 \\c &= 8.45183116443284\end{aligned}$$

```
Command Window

>> ejercicio_10_2_11|
Tolerancia: 1e-12
Máximo num de iteraciones: 50
Iteraciones:
    9

Resultado [b,c]:
    26.7702106083026
    8.45183116443284

Resultado [a]:
    2217952.23196394
```

%Método de Newton multivariable

%PARTE SIMBOLICA

%Construcción de los valores experimentales como simbolicos

N=4;

sY=sym('sy',[1,N]);

sX=sym('sx',[1,N]);

sY=subs(sY,symvar(sY),[2.4,3.8,4.75,21.6]);

sX=subs(sX,symvar(sX),[31.8,31.5,31.2,30.2]);

%Construcción de las funciones simbólicas

syms a b c aux1 aux2

sF=sym('sf',[1,3]);

%Generamos la primer función

aux1=0; aux2=0;

for i=1:N

aux1=aux1+((sY(i)\*log(sY(i)))/((sX(i)-b)^c));

end

for i=1:N

aux2=aux2+(1/((sX(i)-b)^(2\*c)));

end

sF(1)=aux1/aux2;

%Generamos la segunda función

aux1=0; aux2=0;

for i=1:N

aux1=aux1+((sY(i)\*log(sY(i)))/((sX(i)-b)^c));

end

for i=1:N

```
        aux2=aux2+(1/((sX(i)-b)^(2*c+1)));
end
sF(2)=aux1*aux2;
aux1=0; aux2=0;
for i=1:N
    aux1=aux1+((sY(i)*log(sY(i)))/((sX(i)-b)^(c+1)));
end
for i=1:N
    aux2=aux2+(1/((sX(i)-b)^(2*c)));
end
sF(2)=sF(2)-(aux1*aux2);
%Generamos la tercer función
aux1=0; aux2=0;
for i=1:N
    aux1=aux1+((sY(i)*log(sY(i)))/((sX(i)-b)^c));
end
for i=1:N
    aux2=aux2+((log(sX(i)-b))/((sX(i)-b)^(2*c)));
end
sF(3)=aux1*aux2;
aux1=0; aux2=0;
for i=1:N
    aux1=aux1+((sY(i)*log(sY(i))*log(sX(i)-b))/((sX(i)-b)^c));
end
for i=1:N
    aux2=aux2+(1/((sX(i)-b)^(2*c)));
end
sF(3)=sF(3)-(aux1*aux2);
%Jacobian(F,X) devuelve la matriz jacobiana simbolica
sJ=jacobian([sF(2),sF(3)],[b,c]);

%ALGORITMO DE NEWTON
%Punto inicial
x=[26.8,8.3];
%Tolerancia
tol=input('Tolerancia: ');
%Variables de iteración
max_iteraciones=input('Máximo num de iteraciones: ');
iteraciones=0;

%Algoritmo
while iteraciones<max_iteraciones
    %Aumentamos número de iteraciones
    iteraciones=iteraciones+1;
    %Evaluamos la función -F en el x actual, obteniendo -F(x)
    F=-double(subs([sF(2),sF(3)],[b,c],x));
    %Evaluamos el jacobiano J en el x actual, obteniendo J(x)
    J=double(subs(sJ,[b,c],x));
    %Resolvemos el SistLin J(x) y = -F(x) para y
    y=mldivide(J,F');
    %Hacemos x=x+y, es decir, actualizamos nuestra aprox
    x=x+y';
    %Vemos si hemos cumplido la tolerancia de error
    if (norm(y,'inf')<=tol)
        %Terminamos, imprimimos resultado y salimos
```

```
        break;
    end
    %Sino, seguimos calculado
end
%Mensajes de resultado
if(iteraciones>=max_iteraciones)
    disp('Número máximo de iteraciones alcanzado');
end
disp('Iteraciones:'); disp(iteraciones);
%Mostramos los resultados obtenidos para b y c
disp('Resultado [b,c]:'); disp(x');
%Sustituimos b,c en la expresión de a y mostramos el cálculo
disp('Resultado [a]:'); disp(double(subs(sF(1),[b,c],x)));
```