SOLUCIÓN DE SISTEMAS DE ECUACIONES NO LINEALES

Prof. Jenny Pérez

SISTEMAS DE ECUACIONES NO LINEALES

Este tipo de sistemas los escribiremos en la forma:

$$\left\{\begin{array}{c} f_1(x_1, x_2, ..., x_n) = 0 \\ f_2(x_1, x_2, ..., x_n) = 0 \\ \dots\dots\dots\dots\dots\dots \\ f_n(x_1, x_2, ..., x_n) = 0 \end{array}\right\}$$

o más brevemente como $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ donde $\mathbf{0}$ es el vector nulo de n componentes, $\mathbf{x}$ es un vector de $\mathbb{R}^n$ y $\mathbf{f}$ es la función vectorial dependiente de n variables reales dada por:

$$\begin{array}{rcl} \mathbf{f} : \mathbb{R}^n & \to & \mathbb{R}^n \\ \mathbf{x} = (x_1, x_2, ..., x_n)^T & \to & \mathbf{f}(\mathbf{x}) = \left\{\begin{array}{c} f_1(x_1, x_2, ..., x_n) \\ f_2(x_1, x_2, ..., x_n) \\ \dots\dots\dots\dots\dots\dots \\ f_n(x_1, x_2, ..., x_n) \end{array}\right\} \end{array}$$

Los métodos de Newton-Raphson y sus variantes, y los métodos iterativos estudiados en temas anteriores para el caso de una única ecuación o un SEL, pueden extenderse fácilmente al caso de sistemas de n ecuaciones no lineales con n incógnitas.

Definición 5.6.1 ***Matriz Jacobiana*** *Dadas n funciones con n variables independientes :*

$$\begin{gathered} f_1(x_1, x_2, x_3, \cdots, x_n) \\ f_2(x_1, x_2, x_3, \cdots, x_n) \\ f_3(x_1, x_2, x_3, \cdots, x_n) \\ \vdots \\ f_n(x_1, x_2, x_3, \cdots, x_n) \end{gathered}$$

Se define el Jacobiano como la matriz $J(x_1, x_2, \cdots, x_n)$ dada por:

$$J(x_1, x_2, \cdots, x_n) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

Método de Newton-Raphson para SENL:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} - J(x_0, y_0)^{-1} \begin{bmatrix} f_1(x_0, y_0) \\ f_2(x_0, y_0) \end{bmatrix}$$

Se considera un punto de aproximación inicial definido por $X_{(0)}=(x_{(0)1}, x_{(0)2}, \cdots, x_{(0n)})$ y un punto $X_{(1)}$ cercano a $X_{(0)}$ definido por $X_{(1)}=(x_{(1)1}, x_{(1)2}, \cdots, x_{(1)n})$. Reemplazando en la ecuación anterior los siguientes términos:

$$\begin{bmatrix} x \\ y \end{bmatrix} \quad por \quad \mathbf{X}^1$$

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \quad por \quad \mathbf{X}^0$$

$$J(x_0, y_0)^{-1} \quad por \quad \mathbf{J}^{-1}(x_1^0, x_2^0, \cdots, x_n^0)$$

$$\begin{bmatrix} f_1(x_0, y_0) \\ f_2(x_0, y_0) \end{bmatrix} \quad por \quad \mathbf{F}(x_1^0, x_2^0, \cdots, x_n^0)$$

5.6.1. Algoritmo del método de Newton para sistemas de ecuaciones no lineales

Sea un sistema de ecuaciones no lineales definido por:

$$
\begin{aligned}
f_1(x_1, x_2, \cdots, x_n) &= 0 \\
f_2(x_1, x_2, \cdots, x_n) &= 0 \\
&\vdots \\
f_n(x_1, x_2, \cdots, x_n) &= 0
\end{aligned}
$$

dada la aproximación inicial $\mathbf{X}^0 = (x_1^0, x_2^0, \cdots, x_n^0)^t$, con un error de tolerancia permitido ε. Se desea calcular una sucesión de puntos $\mathbf{X}^{1,\ldots,N} = \{x_1^i, x_2^i, \cdots, x_n^i\}$ (donde N representa el número de iteraciones) que se aproximan a la solución buscada, para ello se deben seguir los pasos descritos a continuación:

paso1: i=0

paso2: Se evalúan las funciones dadas en la aproximación x^i

$$\mathbf{F} = \begin{bmatrix} f_1(x_1^i, x_2^i, \cdots, x_n^i) \\ f_2(x_1^i, x_2^i, \cdots, x_n^i) \\ \vdots \\ f_n(x_1^i, x_2^i, \cdots, x_n^i) \end{bmatrix}$$

Paso 3: Se evalúa la matriz jacobiana en la aproximación x^i, $\mathbf{J}(x_1^i, x_2^i, \cdots, x_n^i)$

$$\mathbf{J} = \begin{bmatrix} \frac{\partial}{\partial x_1} f_1(x_1^i, x_2^i, \cdots, x_n^i) & \frac{\partial}{\partial x_2} f_1(x_1^i, x_2^i, \cdots, x_n^i) & \cdots & \frac{\partial}{\partial x_n} f_1(x_1^i, x_2^i, \cdots, x_n^i) \\ \frac{\partial}{\partial x_1} f_2(x_1^i, x_2^i, \cdots, x_n^i) & \frac{\partial}{\partial x_2} f_2(x_1^i, x_2^i, \cdots, x_n^i) & \cdots & \frac{\partial}{\partial x_n} f_2(x_1^i, x_2^i, \cdots, x_n^i) \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial}{\partial x_1} f_n(x_1^i, x_2^i, \cdots, x_n^i) & \frac{\partial}{\partial x_2} f_n(x_1^i, x_2^i, \cdots, x_n^i) & \cdots & \frac{\partial}{\partial x_n} f_n(x_1^i, x_2^i, \cdots, x_n^i) \end{bmatrix}$$

Paso 4: Se resuelve el sistema lineal para obtener $\triangle \mathbf{X}$

$$\mathbf{J}(x_1^i, x_2^i, \cdots, x_n^i) \triangle \mathbf{X} = -\mathbf{F}$$

Paso 5: Se obtiene el vector solución

$$\mathbf{X}^{i+1} = \mathbf{X}^i + \triangle \mathbf{X}$$

Paso 6: Para calcular el error de tolerancia permitido se determina la norma del vector y se compara con la tolerancia dada.

La norma 2 esta dada por la expresión:

$$\| (\mathbf{x}^{i+1} - \mathbf{x}^{i}) \|_2 = \sqrt{(x_j^{i+1} - x_j^{i})^2 + \cdots + (x_n^{i+1} - x_n^{i})^2}$$

Si:

- $\| (\mathbf{x}^{i+1} - \mathbf{x}^{i}) \|_2 < \varepsilon$ vaya al paso 6
- $\| (\mathbf{x}^{i+1} - \mathbf{x}^{i}) \|_2 > \varepsilon$ vaya al paso 5

Paso 7: Haga i=i+1 vaya al paso 2

Paso 8: Termina el proceso

Ejemplo:

Encuentre una solución a los siguientes sistemas no lineales usando el método de Newton. Itere hasta que $\left\|x^{(k)} - x^{(k-1)}\right\|_\infty < 10^{-5}$, y obtenga la solución cercana a $(2.5, 0.2, 1.6)^t$.

1.
$$\begin{array}{rcrcrcrcl} 3x_1^2 & - & 10x_1 & + & x_2^2 & + & 8 & = & 0 \\ 3x_1x_2^2 & + & x_1 & - & 10x_2 & + & 8 & = & 0 \end{array}$$

2.
$$\begin{array}{rcrcl} 3x_1^2 & - & 1x_2^2 & = & 0 \\ 3x_1x_2^2 & - & 1x_1^3 - 1 & = & 0 \end{array}$$

Código

```
function P=NewtonSNL(F,J,x,Error,N)
for i=1:N
    Dx=-J(x)\F(x);
    disp([i x']);
    if norm[Dx,inf]<Error;
        disp('Solucion')
        disp(x+Dx)
        break;
    end
    x=x+Dx;
end
if i==N
    disp('Se ha excedido el numero de iteraciones')
end
P=x;
```

Script del ejemplo

```
clc
x=[1 2];
F=@(x)[3*x(1).^2-10*x(1)+x(2).^2+8; 3*x(1).*x(2).^2+x(1)-10*x(2)+8];
J=@(x)[6*x(1)-10,2*x(2);3*x(2).^2+1,6*x(1).*x(2)-10];
Error=1e-02
N=60;
NewtonSNL(F,J,x',Error,N)
%xo=[0.1 0.2];
%syms x, syms y, v=[x y]
%A=jacobian([3*x^2-10*x+y^2+8; 3*x*y^2+x-10*y+8],[x y])
%J= subs(A, v, x0)
```

Salida

```
   52.0000    0.5904    0.9992

   53.0000    1.4050    1.5597

   54.0000    1.2293    0.7324

   55.0000    2.3531    2.2149

   56.0000    0.8972    2.2171

   57.0000    1.0559    0.9484

   58.0000    2.5429    2.9328

   59.0000   -1.0947    4.3911

   60.0000    4.7700   10.6931

Se ha excedido el numero de iteraciones

ans =

    9.8240
   -0.3892
```