**Utilización del comando *fsolve* de Scilab para la determinación de las Raíces de un Sistema de Ecuaciones No Lineales**

**Comando fsolve**

find a zero of a system of n nonlinear functions

**Syntax**

[x [,v [,info]]]=fsolve(x0,fct [,fjac] [,tol])

**Arguments**

x0: real vector (initial value of function argument).

fct: external (i.e function or list or string).

fjac: external (i.e function or list or string).

tol: real scalar. precision tolerance: termination occurs when the algorithm estimates that the relative error between x and the solution is at most tol. (tol=1.d-10 is the default value).

x: real vector (final value of function argument, estimated zero).

v: real vector (value of function at x).

info:

 termination indicator

 0: improper input parameters.

 1: algorithm estimates that the relative error between x and the solution is at most tol.

 2: number of calls to fcn reached

 3: tol is too small. No further improvement in the approximate solution x is possible.

 4: iteration is not making good progress.

**Determinación de las Raíces del Sistema de Ecuaciones del Ejemplo 04**

**Función a evaluar: *Ex\_04\_func***

function f = Ex\_04\_func(x,c0,K1,K2)

// Evaluación del sistema de ecuaciones del Ejemplo 04

// c0(1) = ca0 / c0(2) = cb0 / c0(3) = cc0 / c0(4) = cd0

ca = c0(1) - 2\*x(1)\*c0(2) - x(2)\*c0(4);

cb = (1 - x(1))\*c0(2);

cc = c0(3) + x(1)\*c0(2) + x(2)\*c0(4);

cd = (1 - x(2))\*c0(4);

f(1) = cc / ca^2 / cb - K1;

f(2) = cc / ca / cd - K2;

f = [f(1);f(2)]; // Vector columna

endfunction

**Se ejecuta la function *Ex\_04\_func***

**Se ingresan los datos del sistema:**

--> x0=[0.1 0.1]' // Valores iniciales de la conversión (vector columna)

 x0 =

 0.1

 0.1

--> c0=[40 15 0 10] // Concentraciones iniciales

 c0 =

 40. 15. 0. 10.

--> K1=5e-04 // Constante de equilibrio de la 1er. reacción química

 K1 =

 0.0005

--> K2=4e-02 // Constante de equilibrio de la 2da. reacción química

 K2 =

 0.04

**Se ejecuta fsolve en la ventana de comandos**

[x,f,inf]=fsolve(x0,Ex\_04\_func) // Ejecución del comando fsolve

inf =

 1.

 f =

 3.047D-17

 -3.261D-16

 x =

 0.1202667

 0.4786707

Eventualmente al comando ***fsolve*** se le puede ingresar la función Jacobiana,

function J = Jacobian(x,c0)

// Jacobian for Example\_04

// c0(1) = ca0 / c0(2) = cb0 / c0(3) = cc0 / c0(4) = cd0

ca = c0(1) - 2\*x(1)\*c0(2) - x(2)\*c0(4);

cb = (1 - x(1))\*c0(2);

cc = c0(3) + x(1)\*c0(2) + x(2)\*c0(4);

cd = (1 - x(2))\*c0(4);

J(1,1)=c0(2)\*((ca^2)\*cb + cc\*(4\*ca\*cb + ca^2))/(ca^4)/(cb^2);

J(1,2)=c0(4)\*((ca^2)\*cb + 2\*cc\*ca\*cb)/(ca^4)/(cb^2);

J(2,1)=c0(2)\*(ca+2\*cc)/(ca^2)/cd;

J(2,2)=c0(4)\*(ca\*cd+cc\*(cd+ca))/(ca^2)/(cd^2);

endfunction

En este caso, en la línea de comandos se ejecuta la función ***Jacobian*** y a continuación se escribe:

--> [x,f,inf]=fsolve(x0,Ex\_04\_func,Jacobian)

 inf =

 1.

 f =

 3.057D-17

 -3.261D-16

 x =

 0.1202667

 0.4786707