1 Métodos iterativos y de gradiente

En esta práctica se va a estudiar las instrucciones de Matlab para resolver sistemas no lineales. Se utilizaran tanto los métodos iterativos como los obtenidos en la minimización no restringida de funciones de varias variables.

1.1 Resolución analítica: solve

En aquellos casos en que sea posible obtener una solución analítica, se puede utilizar la instrucción solve. Por ejemplo, el siguiente sistema

$$F(X)=0\rightarrow\begin{pmatrix} f_1(x,y)\\ f_2(x,y)\end{pmatrix}=\begin{pmatrix} 5x^2-y^2\\ y-\frac{1}{4}(\sin x+\cos y)\end{pmatrix}=\begin{pmatrix}0\\0\end{pmatrix}\ \text{con}\ D=\{-2\le x,y\le 2\}$$

Se resuelve mediante las instrucciones

```
» syms x y real
» F=[5*x^2-y^2; y-(sin(x)+cos(y))/4];
» S=solve(F(1),F(2))  % Solución doble
```

Se pueden obtener los puntos de corte de forma gráfica mediante

```
» t=linspace(-2,2);[X,Y]=meshgrid(t,t);
» Z1=subs(F(1),{x,y},{X,Y});Z2=subs(F(2),{x,y},{X,Y});
» figure(1);
» hold off; contour(X,Y,Z1,[-1,0,1]);hold on;
» contour(X,Y,Z2,[-1,0,1]);grid on; axis('equal');
» legend('negativo','nulo','positivo');
```

Sin embargo, esta instrucción no siempre funciona correctamente.

Se pide:

1. Verificar el funcionamiento de *solve* cuando se intenta resolver el sistema.

$$\begin{aligned}4x^2+y^2-4&=0\\ e^x+y-1&=0\end{aligned}\ \text{con}\ D=\{-2\le x,y\le 2\}$$

1.2 Métodos Iterativos

MATLAB implementa la función ***fsolve*** para resolver (obtener las raíces) sistemas de ecuaciones no lineales. A continuación resumimos las diferentes sintaxis.

X=fsolve(fun,x0)	Intenta resolver el sistema no lineal fun(X)=0 partiendo del valor X=x0
X=fsolve(fun,x0,options)	Resuelve el sistema con las opciones seleccionadas mediante la instrucción *optimset*
[X,F]=fsolve(fun,x0)	Devuelve también el valor de la función
[X,F,finflag]=fsolve(…)	Devuelve también la condición que ha motivado el final del programa.

	1 La función converge a la solución 2 El cambio de x fue menor que la tolerancia especificada 3 El cambio del residuo fue menor que la tolerancia especificada 4 La magnitud de la dirección de búsqueda fue menor que la tolerancia especificada 5 El número de iteraciones supera el valor MaxIter o el número de evaluaciones de la función supera FunEvals. -1 El algoritmo fue finalizado por la función de salida. -2 El algoritmo parece converger a un punto que no es la raíz. -3 El radio de confianza es demasiado pequeño. -4 La dirección de búsqueda no disminuye el valor del residual.
[X,F,finflag,info]=fsolve(…)	Devuelve también una estructura con información sobre el proceso. Sus campos son: iterations Numero de iteraciones realizadas count Número de evaluaciones de la función algorithm Algoritmo usado cgiterations Número de iteraciones PCG (solo con algoritmos de gran escala) stepsize Tamaño final del paso (solo con algoritmos de mediana escala)
[X,F,finflag,info,jaco]=fsolve(…)	Devuelve también el valor del jacobiano

La función viene descrita en un m-archivo, pudiendo tener uno o dos valores de salida, el valor de la función en el punto y el del jacobiano. Para usar el valor del jacobiano, es necesario que se haya activado previamente mediante

```
» options = optimset('Jacobian','on')
```

1.2.1 Opciones

Las opciones se definen con la instrucción *optimset* y estan asociadas al tipo de algoritmo usado en la resolución así como al tamaño del problema (media o gran escala). La opcion *'LargeScale' 'on'* indica que se prefiere utilizar un algoritmo de gran escala; esto es simplemente una preferncia, puesto que depende de que la función verifique las condiciones exigidas en tales algoritmos. En todo caso el número de ecuaciones debe coincidir o superar el número de incógnitas (no puede ser indeterminado).

1.2.1.1 Opciones comunes

DerivativeCheck	Compara las derivadas calculadas mediante el Jacobiano (dadas por la función) con las calculadas mediante diferencias finitas.
Diagnostics	Muestra información diagnóstica sobre la function a resolver.
DiffMaxChange	Maximo cambio de las variables cuando se usan diferencias finitas.
DiffMinChange	Mínimo cambio de las variables cuando se usan diferencias finitas.
Display	Nivel de información en pantalla: 'off' ninguna; 'iter' muestra cada iteración; 'final' (defecto) sólo muestra la última.
FunValCheck	Verifica la validez de los valores de la función objetivo. 'on' muestra un mensaje de error cuando la función objetivo devuelve un valor complejo, Inf,o NaN. 'off' (defecto) no muestra los errors.
Jacobian	Si 'on', utiliza el Jacobiano definido por la función, en otro caso lo aproxima mediante diferencias finitas.
MaxFunEvals	Máximo número de evaluaciones de la función permitido.
MaxIter	Máximo número de iteraciones permitido.
PlotFcns	Dibuja varias medidas de progreso del algoritmo durante su ejecución; puede ser una función definida por el usuario o utilizar alguna de las opciones de MATLAB, tales como @optimplotx (punto actual), @optimplotfunccount

(contador de la función), @optimplotfval (valor de la función), @optimplotresnorm (norma del residuo), @optimplotstepsize (paso actual).

TolFun Tolerancia admisible del valor de la funtion.

TolX Tolerancia admisible de la variación de x.

1.2.1.2 Opciones de los algoritmos de mediana escala

NonlEqnAlgorithm especifica el algoritmo de resolución del sistema no lineal: 'dogleg' (defecto, newton con regiones de confianza), 'lm' (Levenberg-Marquardt), 'gn' (Gauss-Newton).

LineSearchType Elección del algoritmo de búsqueda, sólo aplicable a Levenberg-Marquardt y Gauss-Netwton. Puede ser 'cubicpoly' o 'quadcubic' (defecto).

1.2.2 Ejemplos

Ejemplo 1: Resolver, comenzando en x0=[-5,-5], el siguiente sistema nolineal del apartado anterior

$$F(X)=0\rightarrow\begin{pmatrix} f_1(x,y)\\ f_2(x,y)\end{pmatrix}=\begin{pmatrix} 5x^2-y^2\\ y-\frac{1}{4}(\sin x+\cos y)\end{pmatrix}=\begin{pmatrix}0\\0\end{pmatrix}\ \text{con}\ D=\{-2\le x,y\le 2\}$$

Se comienza creando el archivo que define esa función:

```
function F = mifun1(x)
F = [5*x(1).^2 - x(2).^2; x(2) - 0.25*(sin(x(1))+cos(x(2)))];
```

A continuación definimos el punto de inicio y seleccionamos las opciones del método

```
» x0 = [-5; -5];
» options=optimset('Display','iter');
» [x,fval] = fsolve(@mifun1,x0,options)
```

Se pide:

2. Comprobar si existe variación en los resultados cuando se utiliza el jacobiano, para lo cual la definición de la función debería ser

```
function [F,JF] = mifun1(x)
...
if nargout==1;return;end % Definicion del Jacobiano
JF(1,:)=[10*x(1) , -2*x(2); -0.25*cos(x(1)), 1+0.25*sin(x(2))];
```

Ejemplo 2. Comenzando con los valores x0= [1,1; 1,1] encontrar la matriz que satisface:

$$X^3=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$$

Como en el caso anterior, se comienza por definir el m-archivo de la función, se da valor al punto inicial, se definen las opciones y se ejecuta la instrucción *fsolve*.

```
function F = mifun2(x)
F = x*x*x-[1,2;3,4];
```

```
» x0 = ones(2);
» options=optimset('Display','off');
» [x,fval,finflag] = fsolve(@mifun2,x0,options)
```

Se pide:

3. Resolver el siguiente sistema no lineal (ejercicio 1 del apartado 1.1):

$$\begin{aligned}4x^2+y^2-4&=0\\ e^x+y-1&=0\end{aligned}\ \text{con}\ D=\{-2\le x,y\le 2\}$$

a. Utilizando las opciones por defecto y partiendo de los puntos: [1,-1],[-1,1].

b. Repetir partiendo del origen y representando las iteraciones.

c. Repetir el caso anterior pero utilizando el jacobiano y otros métodos.

Para todos los casos se puede utilizar el siguiente código en la definición de la función:

```
function [F,JF]=mifun3(X)
if nargin~=1;error('numero de argumetos no valido');end
[n,nc]=size(X);if nc~=1;error('X no es un vector columna'),end
%
% Definicion de la funcion
F=zeros(n,1);
F(1)=4*X(1).^2+X(2).^2-4;
F(2)=exp(X(1))+X(2)-1;
if nargout==1;return;end % Definicion del Jacobiano
JF=zeros(n,n);
JF(1,:)=[4*X(1) , 2*X(2)];
JF(2,:)=[exp(X(1)),1];
```

1.3 Minimización no restringida

MATLAB implementa la función ***lsqnonlin*** para este caso. A continuación resumimos las diferentes sintaxis.

X=lsqnonlin(fun,x0)	Intenta obtener el $\min_x \|F(X)\|_2^2 = \min_x \sum_i f_i(X)^2$ partiendo de X=x0
X=lsqnonlin(fun,x0,lb,ub,options)	Intenta obtener el mínimo con los valores acotados por lb≤X≤ub y las opciones seleccionadas mediante la instrucción *optimset*
[X,resn,resid]=lsqnonlin(…)	Devuelve también la norma dos del residuo y el valor del residual

Para resolver el ejemplo del ejercicio3 anterior mediante esta instrucción partiendo del origen, esto es.

$$\min_x \|F(X)\|_2^2 = \min_x \sum_i f_i(X)^2 = \min_x \left(\left(4x^2 + y^2 - 4\right)^2 + \left(e^x + y - 1\right)^2 \right)$$

se tendría

```
» x0 = [0; 0];
» options=optimset('Display','iter');
» [x,resnorm,residue] = lsqnonlin(@mifun3,x0,options)
```