Práctica 1 Optimización de Procesos Químicos

Uso de funciones de la Toolbox de optimización de Matlab

Objetivo:

El objetivo de esta práctica es doble:

- Familiarizarse con la Toolbox de Optimización de Matlab
- Aplicar los distintos métodos de optimización de funciones sin restricciones tanto de una como de varias variables vistos en la teoría

Cada grupo de alumnos entregará un guión con los resultados.

Preparación de la práctica

Antes de comenzar a trabajar en problemas de optimización numéricos, es conveniente que el alumno aprenda a:

- Definir una función de una y de varias variables en Matlab
- Representar gráficamente funciones de una y de varias variables en Matlab y sus curvas de nivel.
- Calcular analíticamente los mínimos (o máximos) y comprobar en las gráficas el resultado
- Estudiar la convexidad de las funciones y de las regiones definidas por las curvas de nivel.

Para este fin pueden usarse las funciones que figuran más adelante en los apartados de Ejercicios propuestos.

A continuación se da un ejemplo del código en Matlab que permite dibujar una gráfica, y las curvas de nivel para una función de dos variables ($J = x_1^2 - x_2$) en el intervalo [-3,3] para x_1 y [-3,5] para x_2 :

```
l1=-3;
L1=3;
l2=-3;
L2=5;
g=0.2;
x1=[l1:g:L1];
x2=[l2:g:L2];
N1=fix((L1-l1)/g);
N2=fix((L2-l2)/g);
for i=1:N1,
   for j=1:N2,
J(j,i)=x1(i)^2-x2(j);
end
end
figure(1)
surf(x1(1:N1),x2(1:N2),J(1:N2,1:N1))
```

```
figure(2)
contour(x1(1:N1),x2(1:N2),J(1:N2,1:N1))
```

Otra opción es utilizar la función *meshgrid*. Esta función define una matriz de puntos sobre los cuales se evalúa la función (más de una variable), para hacer su representación gráfica. Y luego se utiliza la función *surf* o *contour*, tal como se ilustra en el siguiente ejemplo, donde se dibujan las curvas de nivel de la función $J(x,y) = xexp(x^2-y^2)$:

```
x = -2:.2:2;
y = -2:.2:2;
[X,Y] = meshgrid(x,y);
J = X.*exp(-X.^2-Y.^2);
figure(1)
surf(J)
figure(2)
contour(J)
```

Problemas de Optimización en Matlab

Los problemas de optimización de funciones de una variable sin restricciones, conocida como *optimización escalar*, que matemáticamente se formulan de la siguiente manera:

$$\min_x J(x)$$
$$x \in R$$

Donde se trata de encontrar la variable x (escalar), en el dominio de los reales, que minimice la función J, es decir que el valor de la función evaluada en x sea el valor mínimo que puede alcanzar J en la región definida por los "límites" impuestos a x.

La función de Matlab que resuelve este tipo de problemas es **fminbnd,** que tiene la siguiente sintaxis:

```
x = fminbnd(@fun,x1,x2,opciones)
```

Esta función retorna o devuelve a la variable x la solución al problema, que está definido en *fun*.

En las funciones de optimización, el nombre de la función va precedido de @

El algoritmo de optimización que utiliza esta función es el método de búsqueda de la Sección Dorada y el de interpolación parabólica.

Nótese que la solución no es más que el escalar que minimiza el valor de la función J, y los parámetros x_1 y x_2, definen la región de búsqueda de la solución. En el parámetro definido *opciones*, se pueden especificar parámetros tanto del algoritmo como de ejecución de la función. En caso de no ser utilizado, ya que estos parámetros tienen valores por defecto, se puede poner en su lugar [], o simplemente omitirlo.

[x,fval] = fminbnd(@fun,x1,x2,opciones)

En este caso, además en *fval*, se devuelve el valor de la función *J* evaluada en la solución *x*

[x,fval,exitflag] = fminbnd(@fun,x1,x2,opciones)

En el parámetro *exitflag* la función devuelve un valor que describe la condición de salida de *fminbnd*:

- *> 0* indica que la función converge a la solución
- *0* indica que se ha excedido el máximo número de evaluaciones de la función
- *< 0* indica que no se ha encontrado solución

Ilustración de como resolver un problema de optimización sin restricciones de funciones de una variable en Matlab, utilizando la función **fminbnd** de la toolbox de optimización.

Obtener el mínimo de la función: $f(x) = x^2 - 12x + 3$ en el intervalo $-4 \le x \le 4$

a) Lo primero es escribir un archivo de matlab donde se define la función que se quiere minimizar (*fun)*. Los archivos en matlab pueden ser editados con cualquier editor de textos y deben guardarse con la extensión **.m**

```
function f = mifun1(x)
f = x^3 - 12*x + 3;
```

Se define con la palabra clave function, y se asigna al parámetro **f** el nombre de **nuestra** función que tendrá como parámetros de entrada la variable **x**. A continuación definimos la función objetivo y guardamos el archivo con el nombre que hemos dado a la función, en este ejemplo es "mifun1", que en la sintaxis se corresponde con *fun*

b) Seguidamente se debe realizar la llamada a la función fminbnd que es la encargada de solucionar el problema y transferirle como parámetro de entrada nuestra función objetivo. Esta llamada puede efectuarse de dos formas:

   i) Llamando a la función *fminbnd* desde la propia línea de comandos de Matlab:

   x = fminbnd(@mifun1,-4,4)

   donde mifun1 es el archivo donde se define la función objetivo que se desea minimizar y los límites de la solución.

   Creando un archivo, desde el cual se hace la llamada a la función fminbnd. Esta forma es más recomendable, ya que en este archivo se pueden definir y modificar parámetros. Por ejemplo:

```
% Ejemplo con la funcion fminbnd

%  En este archivo:
%  1) se modifican los parametros de la optimizacion
%  2) se llama a la funcion fminbnd

opciones = optimset('Display','iter','MaxFunEvals',400)

x = fminbnd(@mifun1,-4,4,opciones)
```

En el vector de parámetros opciones, especifico que solo se visualice el valor final, y que el máximo número de evaluaciones de la función sea 400.

(Ver en el help de matlab las funciones *optimset* y *foptions*)

Este ejemplo se encuentra en el archivo *practica1.m*

Ejercicios propuestos:

1. Minimizar las siguientes funciones:

a) $f(x) = 3x^4 + (x-1)^2$ en el intervalo [0,4]

b) $f(x) = x + \cos(x^2)$ en el intervalo [0,pi]

Modificar el máximo número de iteraciones, el máximo número de evaluaciones de la función y la tolerancia sobre x.

Problemas de varias variables

Los problemas de optimización de funciones de más de una variables sin restricciones se formulan de la siguiente manera:

$$\min_x J(x)$$
$$x \in R$$

Donde se trata de encontrar el **vector** *x*, cuyos componentes pertenecen al dominio real, que minimice la función *J*, es decir que el valor de la función evaluada en *x* sea el valor mínimo que puede alcanzar *J*. A este tipo de problema se suele llamar Optimización No lineal sin restricciones.

Una función de matlab que resuelve este tipo de problemas es **fminsearch,** que tiene la siguiente sintaxis:

```
x = fminsearch(@fun,x0,opciones)

[x,fval] = fminsearch(@fun,x0,opciones)
```

[x,fval,exitflag] = fminsearch(@fun,x0,opciones)

fminsearch encuentra el valor de las variables *x* que minimizan la función descrita en *fun*, comenzando por el valor inicial especificado en *x0*.

El algoritmo implementado en esta función, es el método Simplex. Este método es de búsqueda directa por lo que no utiliza gradientes ni numéricos ni analíticos.

La utilización de esta función y las siguientes es la misma que se ha descrito para el caso de la función *fminbnd*.

Ejercicios propuestos

1. Minimizar las siguientes funciones:

a) $f(x) = 3x_1^2 + x_2{}^2$

b) $f(x) = 4(x_1 - 5)^2 + (x_2 - 6)^2$

Para ambos ejercicios el punto inicial es $x = (1,1)^T$

Para ambos ejemplos modifica el parámetro de tolerancia sobre el valor de la función, y la tolerancia para las x.

Otra función de Matlab que proporciona el mínimo de una función de variables sin restricciones es **fminunc**, cuya sintaxis es la siguiente:

x = fminunc(@fun, x0, opciones)
[x,fval,exitflag,grad,hessian] = fminunc (@fun,x0,opciones)

y resuelve el mismo tipo de problema de optimización que *fminsearch*, pero utiliza información del gradiente y el hessiano, devolviendo el valor del gradiente y el hessiano de la función objetivo evaluados en la solución *x*, en las variables *grad* y *hessian* respectivamente.

Por defecto (cuando está activado el parámetro 'GradObj') el método que utiliza esta función es un método de Newton con reflexión interior. Cuando por el contrario este parámetro está desactivado, se utiliza el método de Quasi-Newton.

Ejemplo:

Minimizar la función $f(x) = 3x_1^2 + 2x_1x_2 + x_2^2$

1. Se crea el archivo mifun3.m

```
function [f,g] = mifun3(x)

f = 3*x(1)^2 + 2*x(1)*x(2) + x(2)^2;     %funcion objetivo
```

```
if nargout > 1        % mifun3 tiene dos parametros de salida
   g(1) = 6*x(1) + 2*x(2);          % calcula el gradiente
   g(2) = 2*x(1) + 2*x(2);
end
```

2. Se crea el archivo practica4.m que contiene las siguientes sentencias:

```
opciones = optimset ('GradObj', 'on');    % se activa la opción de gradiente

x0 = [1,1];                               % condiciones iniciales

[x,fval] = fminunc(@mifun3,x0,opciones)    % llamada a la función
```

En el caso de utilizar el Hessiano:

```
function [f,g,H] = mifun3(x)

f = 3*x(1)^2 + 2*x(1)*x(2) + x(2)^2;     %funcion objetivo

if nargout > 1        % mifun3 tiene dos parametros de salida
   g(1) = 6*x(1) + 2*x(2);          % calcula el gradiente
   g(2) = 2*x(1) + 2*x(2);

   if nargout  > 2
     H = …
     end
end
```

<u>Ejercicios propuestos</u>

1. Utilizando la función *fminunc*, maximizar: $f(x) = -x_1^2 + x_1 - x_2^2 + x_2 + 4$

2. Dadas las siguientes funciones :

   a) $f(x) = x + \log(x), \quad x > 0$

   b) $f(x) = \frac{1}{x^2}$

2.1 Estudiar su convexidad
2.2 Representarlas gráficamente

Formulación y resolución de problemas de optimización sin restricciones

1. Sabiendo que la relación entre las variables y, x_1 ,x_2 viene dada por:

$$y = \frac{k_1 x_1}{1 + k_2 x_1 + k_3 x_2}$$

Estimar los valores de los parámetros k_1, k_2 y k_3, que ajustan de la mejor manera posible dicha expresión a los siguientes datos experimentales:

$y_{observada}$	X_1	X_2
0.126	1	1
0.219	2	1
0.076	1	2
0.126	2	2
0.186	0.1	0