Curso de Optimización, 2022

Instituto de Matemática y Estadística (IMERL)

**Práctico 1: Optimización sin restricciones**

**Máxima pendiente en una función cuadrática y uso de fminunc**

Considere el problema , que se desea resolver en forma iterativa usando un algoritmo de descenso, que se puede programar con la siguiente función Matlab/Octave, de nombre **algo1.m**:

function [x\_opt,val\_f,val\_x,iter]=algo1(x0,tol1)

[f,g]=fun1(x0);

val\_x=x0;

val\_f=f;

Niter=100;

iter=0;

x=x0;

while iter<Niter & norm(g)>tol1,

 iter=iter+1;

 d=...; % dirección de descenso

 t=busqueda1(x,d); % búsqueda lineal

 x=x+t\*d ;

 [f,g]=fun1(x);

 val\_x=[val\_x x];

 val\_f=[val\_f f];

end

x\_opt=x;

Se usa:

1. Una función **fun1.m** , que debe calcular el valor de  y de su vector gradiente  (que será una matriz columna)
2. Una función **busqueda1.m** que hace la búsqueda lineal en la dirección **d**

En este primer práctico, el vector **d** debe elegirse en forma adecuada para que la dirección de descenso sea la de máxima pendiente.

**Uso en función cuadrática** **con búsqueda lineal exacta:**

Se quiere probar el algoritmo con una función cuadrática de n=8 variables:



donde Q es una matriz nxn simétrica y definida positiva, b es una matriz nx1, y c es un número. Estos datos pueden generarse en forma aleatoria, confirmando que la matriz *Q* sea definida positiva (recuerde que una matriz diagonal dominante cumplirá esa condición).

Observe que para esta función cuadrática el gradiente es el vector , o sea que se puede hallar directamente la solución exacta  resolviendo un sistema lineal con el comando x=Q\b.

**Se pide:**

1. Implemente la función **fun1.m** para esta función cuadrática, y una función **busqueda1.m** para calcular exactamente el valor mínimo en la búsqueda lineal (utilice una expresión analítica para el valor exacto de *t* que anula , en función de las matrices *Q* y *b)*.
2. Pruebe el programa, comparando con la solución exacta.

**c)** Estudie experimentalmente el orden de convergencia del método (usando la matriz val\_x con los sucesivos vectores x hallados al iterar se puede estudiar la evolución de la norma del error).

**d)** Resuelva el mismo problema con la función **fminunc** del paquete de optimización de Matlab/Octave.

**Opcionales:**

**e)** Estudie como varían los resultados del punto c) al variar el tamaño del problema. Para ello, pruebe con datos de la forma Q=sprandsym(n,0.1); b=zeros(n,1); x0=ones(n,1), tol1=1e-3, y algunos valores de n entre 50 y 100. Trate de definir una matriz algo mal condicionada, por ejemplo con condición del orden de 10. En Matlab se puede indicar en el comando sprandsym con rcond=0.1 (inverso del número de condición).

**f)** Estudie como varían los resultados del punto anterior al aumentar el número de condición de la matriz *Q*: rcond=0.1, 0.05, 0.025.

g) Investigue las opciones de uso de **fminunc**.