Un subproblema muy frecuente en optimización numérica es el siguiente:

Sean conocidos $f : \mathbb{R}^n \to \mathbb{R}$, $x \in \mathbb{R}^n$, $d \in \mathbb{R}^n$, $d \neq 0$, $a, b \in \mathbb{R}$, $a < b$. Es necesario resolver el siguiente problema

$$\begin{aligned} \min \ & f(x + td) \\ & a \leq t \leq b \end{aligned}$$

Este problema es de una sola variable, la variable t. Hay muchos métodos para encontrar numéricamente una aproximación de t^* minimizador del problema.

Un método ineficiente pero que se puede programar muy fácilmente es el siguiente:

$$\text{buscar el mejor} \quad t \text{ en } \{a, a+h, a+2h, ..., b\}$$

donde h es un valor positivo pequeño. El mejor t es el que hace que $f(x+td)$ sea más pequeño. O sea, hay que buscar el mínimo de los valores
$f(x + ad), f(x + (a + h)d), f(x + (a + 2h)d), f(x + (a + 3h)d), ..., f(x + bd)$

Hacer un pequeño programa en Python o Scilab que tenga: dos funciones, una para f y otra para este método ineficiente; las asignaciones para x, d, a, b, h; el llamado a la función que "minimiza"; la escritura de resultados.

```
def f1(x):
    t = x[0]*x[0] + x[1]*x[1]
    return t
#-----------------------------------------------------------
def min1abFB(f, x, d, a, b, h):
    #
    # min f( x + t d )  t en [a,b]
    #
    # evalua f( x + t d ) para t = a, a+h, a+2h, ..., b
    #
    # devuelve  tmin,  fmin
    #

    ...
```

```
    return [tmin, fmin]
#------------------------------------------------------------

x = ...
d = ...

res = min1abFB(f1, x, d, -20, 20, 0.0001)

print( ...
```

En Scilab

```
function fx = f1(x)
    fx = x(1)*x(1) + x(2)*x(2)
endfunction
//-----------------------------------------------------------
function [tmin, ftmin] = min1abFB(f, x, d, a, b, h)
    //
    // min f( x + t d )  t en [a,b]
    //
    // evalua f( x + t d ) para t = a, a+h, a+2h, ..., b
    //
    // devuelve  tmin,  fmin
    //

    ...

endfunction

x = [-2; 2]
d = [1; 0]
a = -20
b = 20
[tmin, ft] = min1abFB(f1, x, d, a, b, 0.001)
```