Optimization-Based Modeling and Design of Electronic Circuits
Assignment 2

Prof. J. E. Rayas-Sánchez
February 2019

Develop three Matlab functions to implement the following gradient-based optimization algorithms:

1) The Steepest Descent optimization algorithm.
2) The Conjugate Gradient optimization algorithm using either the Fletcher-Reeves formula or the Polak-Ribiére formula.
3) The Quasi-Newton optimization algorithm using either the Broyden, Fletcher, Goldfarb and Shanno (BFGS) formula, or the Davidon, Fletcher and Powell (DFP) formula.

The three Matlab functions will use exactly the same termination criteria, and will be tested exactly with the same objective functions.

Each algorithm should be terminated when the relative change in the optimization variables is small enough, when the gradient is small enough, or when a maximum number of iterations is reached:

$$\| \boldsymbol{x}_{i+1} - \boldsymbol{x}_i \|_2 < \varepsilon_{\mathrm{x}} (\| \boldsymbol{x}_i \|_2 + \varepsilon_{\mathrm{x}}) \ \vee \ \| \nabla u(\boldsymbol{x}_i) \|_2 < \varepsilon_{\mathrm{g}} \ \vee \ i > i_{\max}$$

where $\boldsymbol{x}_i \in \Re^n$ is the vector containing the n optimization variables at the i-th iteration, and u: $\Re^n \rightarrow \Re$ is the objective function. Scalar limits are defined by ε_{x}, ε_{g} and $i_{\max}$.

The implemented algorithms should have the following functionality:

```
% Usage: [x,Iter,FunEval,EF] = OptAlgName(fun,x0,MaxIter,epsg,epsx)
%        fun: name of the multidimensional scalar objective function
%             (string). This function takes a vector argument of length
%             n and returns a scalar.
%        x0: starting point (row vector of length n).
%        MaxIter: maximum number of iterations to find a solution.
%        epsg: minimum norm of the objective function gradient at the solution
%        epsx: minimum relative change in the optimization variables x.
%        x: solution found (row vector of length n).
%        Iter: number of iterations needed to find the solution.
%        FunEval: Number of function evaluations needed.
%        EF: exit flag,
%        EF=1: successful optimization (gradient is small enough).
%        EF=2: algorithm converged (relative change in x is small enough).
%        EF=-1: maximum number of iterations exceeded.
```

where `OptAlgName` is the selected name for the Matlab file that implements each algorithm.

Using ε_x = `epsx` = 10^{-6}, ε_g = `epsg` = 10^{-5}, and $i_{\max}$ = `MaxIter` = 10,000, test each of your algorithms with at least the following objective functions and starting points:

Name	$u(\boldsymbol{x}) =$	$\boldsymbol{x}^*$	$\boldsymbol{x}_0^{\mathrm{T}}$
Quadratic 1	$\frac{5}{2}x_1^2 + \frac{1}{2}x_2^2 + 2x_1x_2 - 3x_1 - x_2$	$[1 \quad -1]^{\mathrm{T}}$	[0 0] [10 -10] [-25 49] [-500 1000]
Quadratic 2	$\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{x} + c$, where $\boldsymbol{Q} = \begin{bmatrix} 550 & -37 & -34 & -30 & 35 \\ -37 & 310 & 22 & 18 & -17 \\ -34 & 22 & 310 & 15 & -20 \\ -30 & 18 & 15 & 310 & -24 \\ 35 & -17 & -20 & -24 & 550 \end{bmatrix}$, $\boldsymbol{b} = [100 \quad -200 \quad 300 \quad -400 \quad 500]^{\mathrm{T}}$, $c = 10$	$\begin{bmatrix} -0.0891 \\ 0.5956 \\ -1.1358 \\ 1.2345 \\ -0.8724 \end{bmatrix}$	[0 0 0 0 0] [1 1 1 1 1] [-1 2 -3 4 -5] [2 -2 -2 2 -2]
Bowl	$(x_1 - 6)^2 + \frac{1}{25}(x_2 - 4.5)^4$	$[6 \quad 4.5]^{\mathrm{T}}$	[1 1] [20 10] [-40 -5] [60 20]
Rosenbrock	$100(x_2 - x_1^2)^2 + (1 - x_1)^2$	$[1 \quad 1]^{\mathrm{T}}$	[0 0] [-1 -1] [0 10] [25 100]

For each objective function and starting point, you must report the solution found (`x`), the number of iterations needed (`Iter`), the number of function evaluations needed (`FunEval`), the exit flag value (`EF`), and the Euclidean norm of the error between the solution found and the exact solution. Summarize your results (for each algorithm) in one table per objective function, with the following structure:

$\boldsymbol{x}_0^{\mathrm{T}}$	$\boldsymbol{x}^{\mathrm{T}}$	`Iter`	`FunEval`	`EF`	$\lVert \boldsymbol{x}^* - \boldsymbol{x} \rVert_2$
…					

To enhance the performance of your implementation of the conjugate gradient method, consider resetting the search direction to the steepest descent direction every n+1 iterations.

Indicate the reference(s) used to write your algorithms.

Include in your report the main conclusions for each problem and for the overall assignment.

Optional: compare your results with those using the Matlab command `fminsearch` on the same test cases (with the same starting points).

Submission deadline: March 6, 2019.