Summer School on Numerical Optimization Software - University of Freiburg

**Exercise 1: Unconstrained Optimization**

Gianluca Frison, Hans Mittelmann, Moritz Diehl

The unconstrained (nonlinear) optimization problem is

$$\min_x f(x)$$

where $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ is a (possibly nonlinear) function.

The solution methods and software that can deal with this optimization problem depend e.g. on the smoothness and convexity of $f$.

# Beale's function

Beale's test function

$$f(x, y) = (x(1 - y) - 1.5)^2 + (x(1 - y^2) - 2.25)^2 + (x(1 - y^3) - 2.625)^2 \tag{1}$$

has an unique global minimizer at $(3, 0.5)$ and a saddle point at $(0, 1)$. There is a further saddle point at $(0.100538, -2.644514)$. For $x = 0$ or $y = 1$, $f$ is constant with value 14.203125. For $x < 0$ there exists an arc $y(x) > 0$, where $f$ decreases monotically as $x \to -\infty$. Initial values near $x = 0$ for large $|y|$ or $x < 0$, $y > 0$ will let descent methods fail.

**1.1** Use the `contour` function in Matlab or Octave to draw a contour plot of $f$. Locate possible stationary points of the function `f` and their type (maximum, minimum, saddle-point).

A similar contour plot can be found at the page

`http://plato.asu.edu/sub/nlounres.html`

**1.2** Examine the file `beale.mod`, that contains an AMPL model of the minimization problem (1).

The variables $x$ and $y$ are defined and their initial value is set as

```
var x :=  0.0;
var y := 10.0;
```

The keyword `minimize` is used to specify that the optimization problem is a minimization problem. Afterwards, the name of the objective function is defined (`f` in this case) followed by a colon and the function expression (and a semicolon at the end of the line).

```
minimize f: (x*(1.0-y)-1.5)^2+(x*(1.0-y^2)-2.25)^2+(x*(1.0-y^3)-2.625)^2;
```

In AMPL, a model (saved in the `beale.mod` file) can be loaded using the command

```
model beale.mod;
```

A solver can be chosen using the command (in the case of the Ipopt solver)

```
option solver ipopt;
```

Afterwards, the solver can be used to solve the previously loaded problem with the command

```
solve;
```

and the value of the variables at the solution can be printed with the command

```
display x, y;
```

Test different solvers and starting points. Can all solvers always find the solution?

In case you use AMPL interactively, note that you need to reset the variables to the chosen initial value in between a call to different solvers, otherwise the next solver would start at the solution returned by the previous one. One way to do it without exiting the AMPL session is to use the command

```
reset;
```

A convenient way to deal with that is to add the reset command at the beginning of the model file, and the solver choice, solution and display commands at the end of the model file.

**1.3** Compute the gradient of $f$ on paper. Write a AMPL model to solve the system of nonlinear equations $\nabla f(x) = 0$, that gives the first order necessary conditions for optimality. Notice that this is a purely constraint satisfaction problem. In AMPL, constraints can be specified as

```
subject to
f1: _constraint_1_expression_ = 0;
f2: _constraint_2_expression_ = 0;
```

How many solutions to this system of nonlinear equations can you find? What kind of stationary points are they (maximum, minimum, saddle point)?

Compute the Hessian of $f$ on paper and verify your guesses.

**1.4** In order to find all the stationary points, the NEOS-ICOS solver can be used. The solver page is

```
https://neos-server.org/neos/solvers/go:icos/AMPL.html
```

Submit the model from point 1.3 to the server, choosing 'AMPL model for satisfiability problem'. Read carefully the instructions on the page.

**Important**: when submitting a job to the NEOS server, you must note (copy and paste somewhere) the job number and relative password. These are used to retrieve the results and to kill the job if it hangs. This last point is especially important when using ICOS, that is the most critical NEOS solver.

**Hint**: you may need to add bounds on your variables in order to get meaningful solutions form the solver. In AMPL, bounds on a variable can be easily added to the variable definition and initialization expression, as e.g.

```
var x :=  0.0, >= -5, <= 5;
```

# Rosenbrock's function

The Rosenbrock's function is a non-convex function used as a performance test problem for optimization algorithms. The two dimensional version (the best known one) reads

$$f(x, y) = (a - x)^2 + b(y - x^2)^2$$

and it has a global minimum at $(x, y) = (a, a^2)$. Usually $a = 1$ and $b = 100$. The minimum is found at the bottom of a narrow curved valley, that resembles a banana.

An involved generalization to higher dimension $N$ is

$$f(x) = \sum_{i=1}^{N-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$$

where $x \in \mathbb{R}^N$.

The function has exactly one minimum for $N = 3$, at $(1, 1, 1)$, and exactly two minima for $4 \leq N \leq 7$, the global one at $(1, 1, \ldots, 1)$ and the local one near $(-1, 1, \ldots, 1)$.

**1.5** For $N = 4$, compute the gradient of $f$ on paper. Write an AMPL model to solve the system of nonlinear equations $\nabla f = 0$, that gives the first order necessary conditions for optimality. Try different initial values, and test different solvers. How many stationary points can you find?

**Hint**: you may want to make use of the vector notation in AMPL for variables and constraints. Defined the dimension as a parameter

```
param n := 4;
```

it is possible to define, initialize and set bounds on the variables as

```
var x{1..n} := 2, >= -2, <= 2;
```

It is possible to define a vector of constraints indexed by an index $i$ as

```
cons_name{i in 2..n-1} = cons_expression(x[i], x[i-1], x[i+1]) = 0;
```

where it is possible to use $i$ to index the variable components.

**Hint**: you may want to initialize the components of the variable $x$ to different values. This can be done by initializing them one by one using the `let` command, as e.g.

```
let x[1] := -1;
let x[2] :=  1;
```

**1.6** Submit the model from point 1.5 to the NEOS-ICOS solver. How many stationary points are found by the solver? Guess on their kind (maximum, minimum, saddle-point).

**1.7** Optional. For $N = 4$, compute the Hessian of $f$ on paper and use second-order information to verify your guess on the stationary point kind.