Universität Konstanz
Fachbereich Mathematik und Statistik
Prof. Dr. Stefan Volkwein
Sabrina Rogg

Sommersemester 2015

Optimierung

http://www.math.uni-konstanz.de/numerik/personen/rogg/de/teaching/

Program 2 (6 Points)

Submission by E-Mail: 2015/06/08, 10:00 h

Optimization with boundary constraints
Implementation of the Gradient Projection Algorithm

So far we looked for (local) minimizer $x^* \in \mathbb{R}^n$ of a sufficiently smooth and real valued function $f : \mathbb{R}^n \to \mathbb{R}$ in an open set $\Omega \subseteq \mathbb{R}^n$:

$$x^* = \underset{x\in\Omega}{\operatorname{argmin}} f(x).$$

The first order necessary optimality condition is $\nabla f(x^*) = 0$.

If Ω is given as the closed and bounded domain

$$\Omega = \prod_{i=1}^{n} [a_i, b_i] = \{x \in \mathbb{R}^n \mid \forall i = 1, ..., n : \ a_i \le x_i \le b_i\, ,\ a_i, b_i \in \mathbb{R},\, a_i < b_i\},$$

the above condition must be changed to admit the possibility that a (local) minimizer is located on the boundary of the domain. In Exercise 11 we prove the following modified first order condition:

$$\nabla f(x^*)^\top (x - x^*) \ge 0 \quad \text{for all } x \in \Omega. \tag{1}$$

The *canonical projection* of $x \in \mathbb{R}^n$ on the closed set Ω is given by $P : \mathbb{R}^n \to \Omega$,

$$\big(P(x)\big)_i := \begin{cases} a_i & \text{if } x_i \le a_i \\ x_i & \text{if } x_i \in (a_i, b_i) \\ b_i & \text{if } x_i \ge b_i \end{cases}.$$

It can be shown:

$$x^* \text{ satisfies condition (1)} \quad \Leftrightarrow \quad x^* = P(x^* - \lambda \nabla f(x^*)) \quad \text{for all } \lambda \ge 0$$

The gradient projection algorithm (using the normalized gradient as descent direction) works as follows: Given a current iterate x^k. Let $d^k := -\frac{\nabla f(x_k)}{\|\nabla f(x_k)\|}$. The next iterate is set to

$$x^{k+1} = P(x^k + t_k d^k), \tag{2}$$

where t_k is a step length satisfying the following modified Armijo rule (compare Exercise 12):

$$f(x^{k+1}) - f(x^k) \leq \frac{-\alpha}{t_k} \|x^k - x^{k+1}\|^2. \tag{3}$$

As termination criterion we use

$$\|x^k - P(x^k - \nabla f(x^k))\| < \epsilon.$$

Part 1: Write a file `projection.m` for the function

```
function [px] = projection(x, a, b)
```

with the current point $\mathtt{x} \in \mathbb{R}^n$, lower bound $\mathtt{a} \in \mathbb{R}^n$ and upper bound $\mathtt{b} \in \mathbb{R}^n$ as input arguments. The function returns the (pointwise) projected point $\mathtt{px} \in \mathbb{R}^n$ according to the canonical projection P. Note that this function can be implemented in one line. Test your function for the rectangular 2-D domain defined by the lower bound (lower left corner) $a = (-1; -1)^\top$ and the upper bound (upper right corner) $b = (1; 1)^\top$: compute the projection $P(x)$ of points $x = y + td \in \mathbb{R}^2$ with $y \in \mathbb{R}^2$ as given in the table below, direction $d = (1.5; 1.5)^\top \in \mathbb{R}^2$, step sizes $t = 0$ and $t = 1$. For validation compare your results to the projections given in the table:

Points y:	$P(x)$ for $t = 0$	$P(x)$ for $t = 1$
$(-2; -2)$	$(-1; -1)$	$(-0.5; -0.5)$
$(-1; -1)$	$(-1; -1)$	$(0.5; 0.5)$
$(-0.5; 0.5)$	$(-0.5; 0.5)$	$(1; 1)$
$(2; 0.5)$	$(1; 0.5)$	$(1; 1)$
$(1; -0.5)$	$(1; -0.5)$	$(1; 1)$

Table 1: Testing points and their projections with respect to t

Part 2: Write a function

```
function [t] = modarmijo(fhandle, x, d, t0, alpha, beta, amax, a, b)
```

for the Armijo step size strategy with termination condition (3). The input arguments are as follows:

- `fhandle`: function handle
- `x`: current point
- `d`: descent direction
- `t0`: initial step size
- `alpha`, `beta`: parameters for the Armijo rule, the backtracking strategy
- `amax` : maximum number of iterations
- `a`, `b`: projection bounds

Part 3: Implement the gradient projection algorithm as described above. Generate a file `gradproj.m` for the function

```
function [X] = gradproj(fhandle, x0, epsilon, nmax, t0, alpha, beta, amax, a, b)
```

with input parameters:

- `fhandle`: function handle
- `x0`: initial point
- `epsilon`: for the termination condition.
- `nmax`: maximum number of iteration steps
- `alpha`, `beta`, `amax`: parameters for the Armijo algorithm
- `a`, `b`: projection bounds

The program should return a matrix `X = [x0, x1, x2, ...]` containing the whole iterations.

Part 4: Call the function `gradproj` from a main file `main.m` to test your program for the Rosenbrock function

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

with input argument `x` $\in \mathbb{R}^2$ and output arguments the corresponding function value `f` $\in \mathbb{R}$ and gradient `g` $\in \mathbb{R}^2$. Use the parameters `epsilon=1.0e-2`, `nmax=1.5e+3`, `t0=1`, `alpha=1.0e-2`, `beta=0.5`, `amax = 30`. Take the following initial values and bounds:

1. `x0=[1;-0.5]`, `a=[-1;-1]` and `b=[2;2]`
2. `x0=[-1;-0.5]`, `a=[-2;-2]` and `b=[2;0]`
3. `x0=[-2;2]`, `a=[-2;-2]` and `b=[2;2]`

Visualize the results in suitable plots and write your observations in the written report.