MATH-329 Continuous optimization
Exercise session 12: Constrained algorithms

Instructor: Nicolas Boumal
TAs: Guifré Sánchez, Antoine Gonon

Document compiled on September 23, 2025

1. Implementation of quadratic penalty method. Write code to apply the quadratic penalty method to the simple optimization problem

$$\min_{x \in \mathbb{R}^2} x_1 + x_2 \quad \text{subject to} \quad x_1^2 + x_2^2 - 2 = 0.$$

To solve the subproblems, that is, to minimize F_β for each individual value β, you can use code you wrote in previous exercise sessions / homework assignments, or you can use Matlab's Optimization Toolbox: the code below requires you to provide a function `[val, grad] = F(x, beta)` implementing $F_\beta(x)$ (as the first output) and $\nabla F_\beta(x)$ (as the second output) as well as an initialization `x_in`, and it attempts to return a minimizer `x_out`.

```
% See 'help fminunc': Matlab's unconstrained minimizer.
options = optimoptions('fminunc', 'SpecifyObjectiveGradient', true);
x_out = fminunc(@(x) F(x, beta), x_in, options);
```

It is instructive to visualize the penalized function F_β for various values of β to get a sense of how the penalty shapes the 'landscape' of the cost function, and to display on those plots the sequence of solutions (x_k) that you compute.

Answer.

```
clear all; close all; clc;

% See 'help fminunc'. Requires Matlab's Optimization Toolbox.
% Otherwise, you can also just use your own codes for unconstrained
% optimization which you write earlier in the course.
options = optimoptions('fminunc', 'SpecifyObjectiveGradient', true);

% Store all solutions x_k as columns of X, for display
X = zeros(2, 10);
X(:, 1) = randn(2, 1); % random initialization at first
beta = 1; % pick some initial value for beta
for k = 2 : size(X, 2)
X(:, k) = fminunc(@(x) F(x, beta), X(:, k - 1), options);
beta = 2*beta;
end
```

```
disp(X)

function [val, grad] = F(x, beta)
val = (x(1) + x(2)) + beta*(1/2)*(x(1)^2 + x(2)^2 - 2)^2;
grad = [1, 1]' + beta*(x(1)^2 + x(2)^2 - 2)*2*x;
end
```

■

2. Unbounded penalized function. The function F_β from the previous exercise may fail to be bounded below, in which case minimizing it can completely fail. Verify this claim on the following example:

$$\min_{x\in\mathbb{R}^2} -5x_1^2 + x_2^2 \quad \text{subject to} \quad x_1 = 1.$$

Argue that for $\beta < 10$ the function F_β is unbounded below. What happens for $\beta \geq 10$? Can we still hope to find a solution for this problem via some instantiation of the quadratic penalty method?

Answer. We have

$$\begin{aligned} F_\beta(x) &= -5x_1^2 + x_2^2 + \frac{\beta}{2}(x_1 - 1)^2 \\ &= \left(\frac{\beta}{2} - 5\right) x_1^2 + x_2^2 - \beta x_1 + \frac{\beta}{2}. \end{aligned}$$

When $\frac{\beta}{2} - 5 < 0$, that is, when $\beta < 10$, it is clear that F_β is unbounded below: simply take $x_1 \to +\infty$ with $x_2 = 0$ fixed. However, with $\beta \geq 10$ the function F_β is convex. Even nicer, when $\beta > 10$ the function is strongly convex: it has a unique minimizer corresponding to its critical point. Explicitly,

$$\nabla F_\beta(x) = \begin{bmatrix} (\beta - 10)x_1 - \beta \\ 2x_2 \end{bmatrix}$$

and the critical point is

$$x = \begin{bmatrix} \frac{\beta}{\beta - 10} \\ 0 \end{bmatrix}.$$

It is clear that as $\beta \to \infty$ the global minimizer of F_β converges to $[1, 0]^\top$, which is indeed the correct solution to our target problem

$$\min_{x\in\mathbb{R}^2} -5x_1^2 + x_2^2 \quad \text{subject to} \quad x_1 = 1.$$

Thus, so long as we initialize the quadratic penalty method with $\beta > 10$, we should be fine. The issue of course is: in practice, on a more complicated problem, how would we know? The fact that such issues can arise implies that we, as algorithm designers and coders, must incorporate some mechanisms / heuristics to act appropriately if unboundedness occurs. ■