MATH529 – Fundamentals of Optimization

Fundamentals of Constrained Optimization VIII: Algorithms

Marco A. Montes de Oca

Mathematical Sciences, University of Delaware, USA

Algorithms for Nonlinear Constrained Optimization

One basic idea:

- Use Lagrangian-like functions as proxies (or analytical tools) for dealing with a constrained problem.

Algorithms for Nonlinear Constrained Optimization

In this course:

- Penalty Methods
- Interior-Point Methods

Penalty Methods for Nonlinear Constrained Optimization

Idea:

- Have a mechanism that generates solutions using information about their quality. (Favoring better quality solutions.)

Penalty Methods for Nonlinear Constrained Optimization

Idea:

- Have a mechanism that generates solutions using information about their quality. (Favoring better quality solutions.)
- In the process of determining the quality of those solutions, **penalize** those that are infeasible by reducing their quality **based on the degree they violate the constraints**.

Example:

Say $c_1(\mathbf{x}) = x_1 - x_2 = 3$.

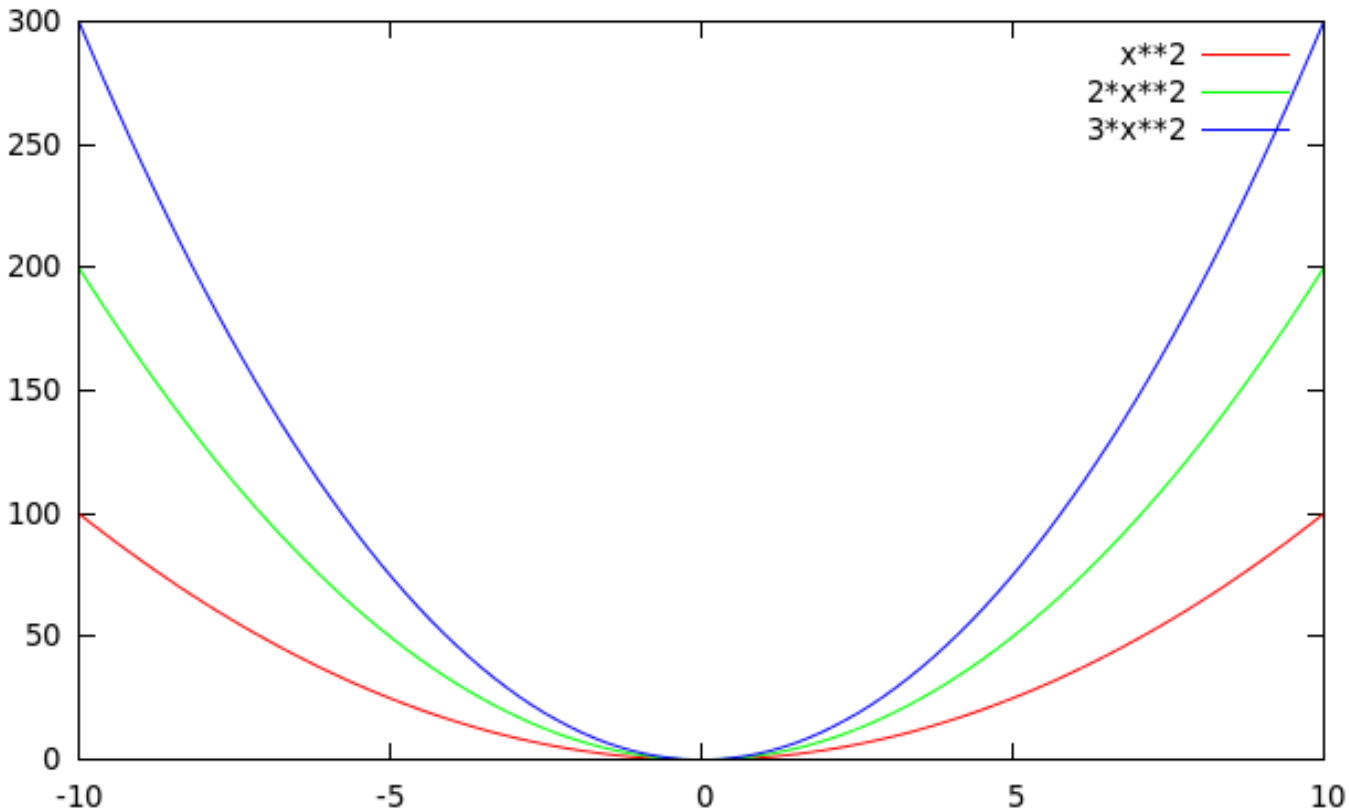
Given $\mathbf{u} = (2, -0.5)^T$, and $\mathbf{v} = (-1, 0)^T$, $\mathbf{u}$ should receive a better score than $\mathbf{v}$ because 2.5 is closer to 3 than -1.

A common way to implement these ideas in order to deal with equality constraints is to define a proxy for the objective function as follows:

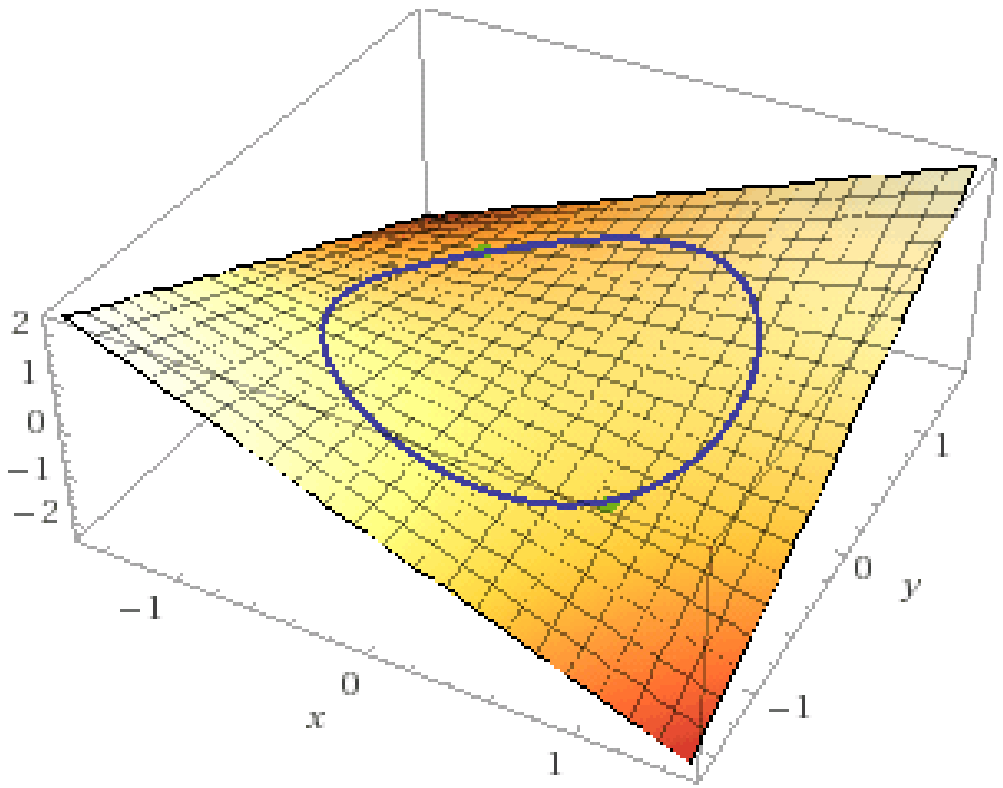
$$Q(\mathbf{x}, \mu) = f(\mathbf{x}) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} (c_i(\mathbf{x}))^2$$

where $\mu > 0$ is called the *penalty parameter*.

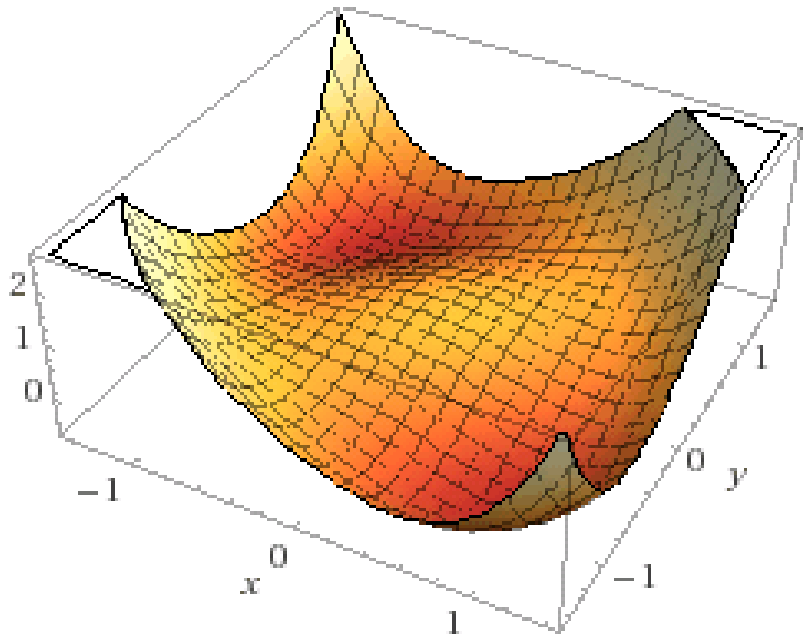
Effects of the penalty parameter:



Minimize xy subject to $x^2 + y^2 = 1$:

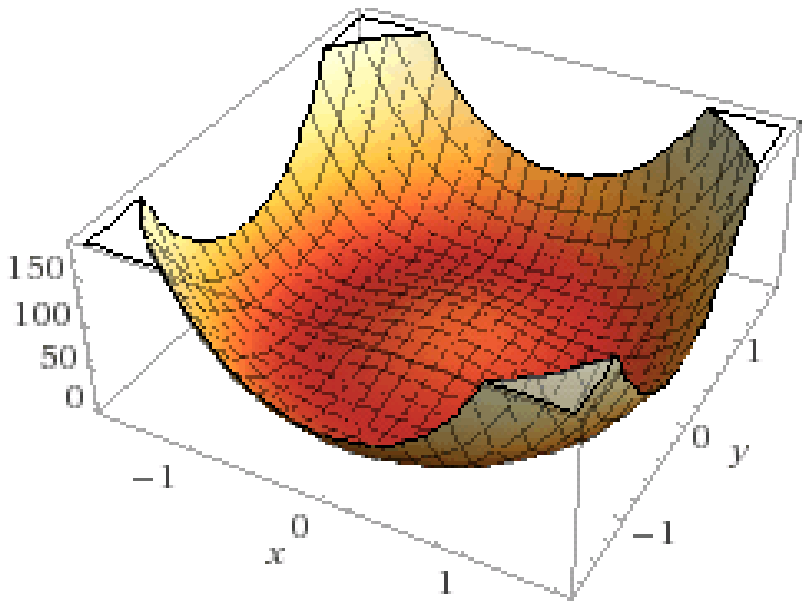


$Q(x, y, 1) = xy + \frac{1}{2}(x^2 + y^2 - 1)^2$:



$(x^\star, y^\star) = (-0.8660, 0.8660)$, or $(x^\star, y^\star) = (0.866, -0.866)$.

$Q(x, y, 40) = xy + 20(x^2 + y^2 - 1)^2$:



$(x^\star, y^\star) = (-0.7115, 0.7115)$, or $(x^\star, y^\star) = (0.7115, -0.7115)$.

Penalty Methods for Nonlinear Constrained Optimization

Penalty Methods:

- Initialize $\mu_0 > 0$, $\mathbf{x}_0$.
- for i = 1,. . . ,k
  - Use your favorite algorithm to find an approximate minimizer of $Q(\mathbf{x}_k, \mu_k)$, call it $\mathbf{m}_k$;
  - If $\mathbf{m}_k$ is good enough, break and return $\mathbf{m}_k$ as solution.
  - Else choose new $\mu_{k+1} > \mu_k$, and set $\mathbf{x}_{x+1} = \mathbf{m}_k$.
- endfor

MATLAB Example

Issues:

- Divergence
- The Hessian becomes ill-conditioned with large values of μ_k.
- It is harder to deal with inequality constraints because in this case:

$$Q(\mathbf{x}, \mu) = f(\mathbf{x}) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} (c_i(\mathbf{x}))^2 + \frac{\mu}{2} \sum_{i \in \mathcal{I}} (\min\{c_i(\mathbf{x}), 0\})^2$$

therefore, Q is no longer twice continuously differentiable.

Augmented Lagrangian Method

We saw that:

$$\nabla_{\mathbf{x}} Q(\mathbf{x}, \mu_k) = \nabla f(\mathbf{x}) + \mu_k \sum_{i \in \mathcal{E}} c_i(\mathbf{x}) \nabla c_i(\mathbf{x})$$

Now, compare this equation with the gradient of the Lagrangian:

$$\nabla_{\mathbf{x}} L(\mathbf{x}, \lambda_i) = \nabla f(\mathbf{x}) - \sum_{i \in \mathcal{E}} \lambda_i \nabla c_i(\mathbf{x})$$

At a solution point of Q, we can say that $c_i(\mathbf{x}_k) \approx -\dfrac{\lambda_i^\star}{\mu_k}$ for all $i \in \mathcal{E}$.

This means that $c_i(\mathbf{x}_k) \to 0$ as $\mu_k \to \infty$, but in general a solution to Q is biased.

A way to reduce this bias, is to use what is called the *augmented Lagrangian* which is defined as:

$$L_A(\mathbf{x}, \lambda, \mu) = f(\mathbf{x}) - \sum_{i \in \mathcal{E}} \lambda_i c_i(\mathbf{x}) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} (c_i(\mathbf{x}))^2$$

The idea is then to use $L_A(\mathbf{x}, \lambda^k, \mu_k)$ instead of $Q(\mathbf{x}, \mu_k)$ as proxy for the constrained problem.

This works because the optimality conditions for $L_A(\mathbf{x}, \lambda^k, \mu_k)$ say that $\nabla L_A(\mathbf{x}_k, \lambda^k, \mu_k) = \mathbf{0}$ and therefore

$$\nabla L_A(\mathbf{x}_k, \lambda^k, \mu_k) = \nabla f(\mathbf{x}_k) - \sum_{i \in \mathcal{E}} (\lambda_i^k - \mu_k c_i(\mathbf{x}_k)) \nabla c_i(\mathbf{x}_k) = \mathbf{0}$$

and so

$$\lambda_i^\star \approx \lambda_i^k - \mu_k c_i(\mathbf{x}_k)$$

for all $i \in \mathcal{E}$.

We can see now that $c_i(\mathbf{x}_k) \approx -\frac{1}{\mu_k}(\lambda^\star - \lambda_i^k)$. So, $c_i(\mathbf{x}_k)$ would be much smaller than before provided that λ_i^k is close to $\lambda_i^\star$.

A method that implements the augmented Lagrangian method would use the formula

$$\lambda_i^{k+1} = \lambda_i^k - \mu_k c_i(\mathbf{x}_k)$$

to have a better behaved algorithm that does not require $\mu_k \to \infty$ (at least not as fast) to have accurate solutions.

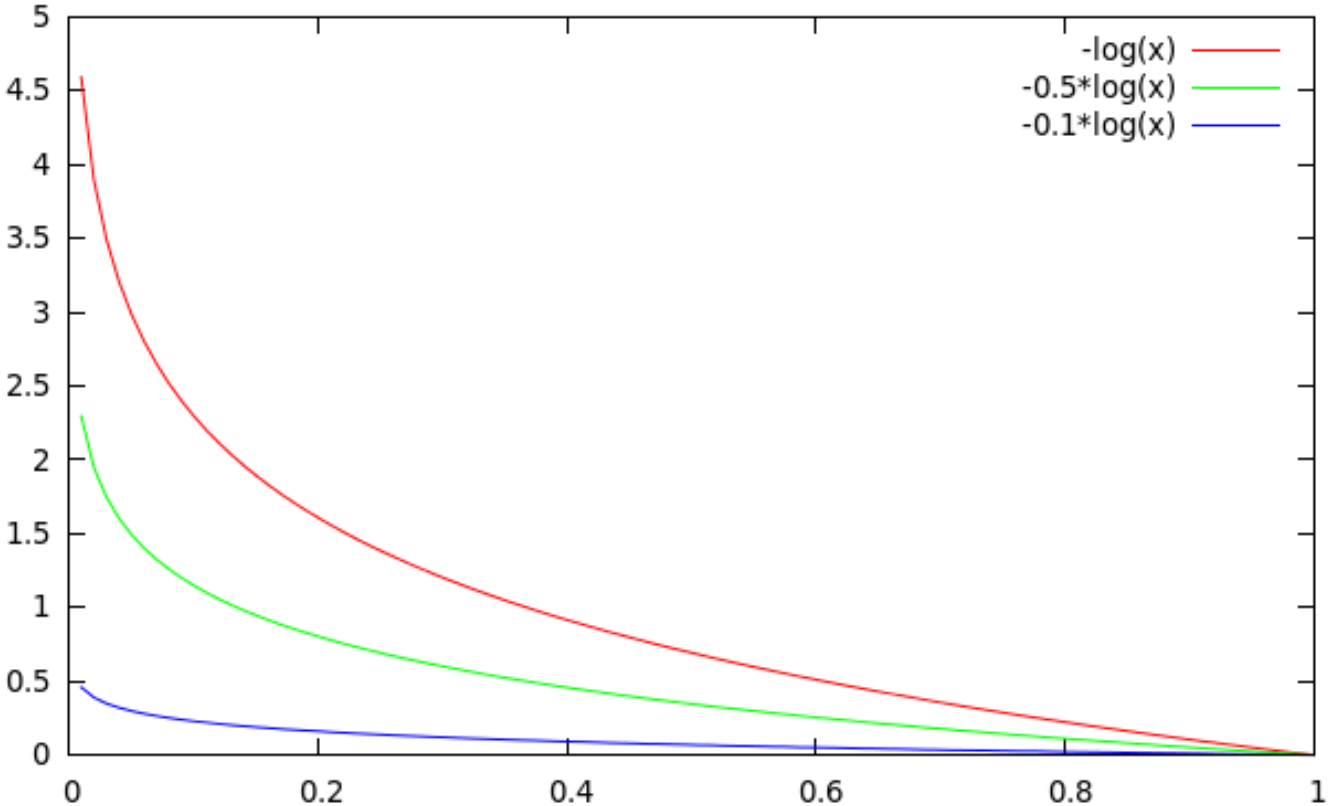
It is possible to modify a problem with inequality constraints so that the augmented Lagrangian method can be used without modification:

The idea is to transform $c_i(\mathbf{x}) \geq 0$ into $c_i(\mathbf{x}) - s_i = 0$ subject to bound constraints (which are easier to deal with).

Another approach is to use:

Barrier Methods

Similar to the penalty method, but now the penalty is smooth:



Common barrier functions:

- $\frac{1}{x}$
- $-\ln(x)$

Example: Minimize $2x^2 + 9y$ subject to $x + y \geq 4$.