An l_1-Augmented Lagrangian algorithm and why, at least sometimes, it is a very good idea

Andrew R. Conn
arconn@us.ibm.com

IBM T.J. Watson Research Center

December 14, 2018, NYU Courant Computer Science,

Numerical Analysis and Scientific Computing Seminar.

Four Fundamental Transparent(?) Unproved Statements:

All computational mathematics is essentially linear.

First derivatives characterise optima.

The derivative of a quadratic is linear

So we need only talk about quadratic problems to understand optimization.

Four Fundamental Transparent(?) Unproved Statements:

All computational mathematics is essentially linear.

First derivatives characterise optima.

The derivative of a quadratic is linear

So we need only talk about quadratic problems to understand optimization.

Four Fundamental Transparent(?) Unproved Statements:

All computational mathematics is essentially linear.

First derivatives characterise optima.

The derivative of a quadratic is linear

So we need only talk about quadratic problems to understand optimization.

Four Fundamental Transparent(?) Unproved Statements:

All computational mathematics is essentially linear.

First derivatives characterise optima.

The derivative of a quadratic is linear

So we need only talk about quadratic problems to understand optimization.

The paradigm unconstrained problem:

Optimize a quadratic.

For example

$\min_x \; f(x) = a + b^\top x + \frac{1}{2} x^\top H x$

Motivated by the statements

The derivative of a quadratic is linear.
First derivatives characterise optima.
and
All computational mathematics is essentially linear.

The paradigm unconstrained problem:

Optimize a quadratic.

For example

$$\min_x \; f(x) = a + b^\top x + \frac{1}{2} x^\top H x$$

Motivated by the statements

The derivative of a quadratic is linear.
First derivatives characterise optima.
and
All computational mathematics is essentially linear.

The paradigm unconstrained problem:

Optimize a quadratic.

For example

$\min_x \; f(x) = a + b^\top x + \frac{1}{2} x^\top H x$

Motivated by the statements

The derivative of a quadratic is linear.
First derivatives characterise optima.
and
All computational mathematics is essentially linear.

The paradigm unconstrained problem:

Optimize a quadratic.

For example

$$\min_x \; f(x) = a + b^\top x + \tfrac{1}{2} x^\top H x$$

Motivated by the statements

The derivative of a quadratic is linear.
First derivatives characterise optima.
and
All computational mathematics is essentially linear.

The paradigm constrained problem:

Optimize a quadratic subject to linear constraints

For example

$\min_x f(x) = a + b^\top x + \frac{1}{2}x^\top Hx$

subject to $Ax \leq c$

Motivated by the statements?

The derivative of a quadratic is linear

and

All computational mathematics is essentially linear.

The paradigm constrained problem:

Optimize a quadratic subject to linear constraints

For example

$\min_x \; f(x) = a + b^\top x + \frac{1}{2} x^\top H x$

subject to $Ax \leq c$

Motivated by the statements?

The derivative of a quadratic is linear

and

All computational mathematics is essentially linear.

The paradigm constrained problem:

Optimize a quadratic subject to linear constraints

For example

$\min_x f(x) = a + b^\top x + \frac{1}{2} x^\top H x$

subject to $Ax \leq c$

Motivated by the statements?

The derivative of a quadratic is linear

and

All computational mathematics is essentially linear.

The paradigm constrained problem:

Optimize a quadratic subject to linear constraints

For example

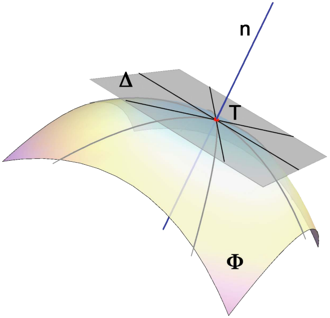
$\min_x \; f(x) = a + b^\top x + \frac{1}{2} x^\top H x$

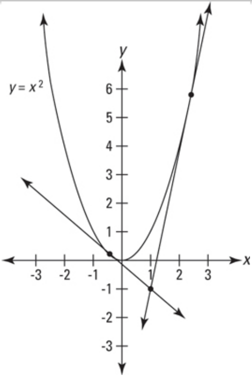
subject to $Ax \leq c$

Motivated by the statements?

The derivative of a quadratic is linear
and
All computational mathematics is essentially linear.

Characterisation of Optimality: Unconstrained Case.

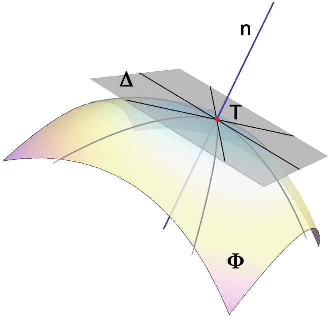




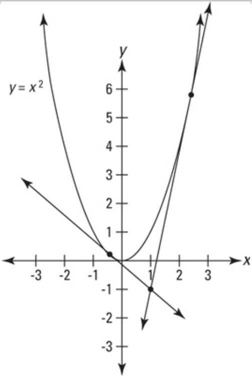
$$\nabla f(x) = 0$$

$$f'(x) = 0$$

Characterisation of Optimality: Unconstrained Case.



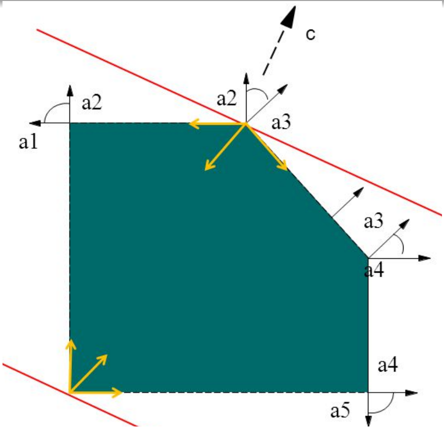
$\nabla f(x) = 0$



$f'(x) = 0$

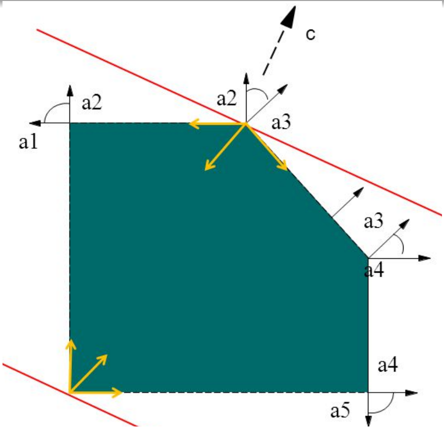
Note: the (first-order) characterisation for a min or max is the same.

Characterisation of Optimality: Constrained Case.



$\max_x c^T x$ subject to $a_i^T x \leq b_i$

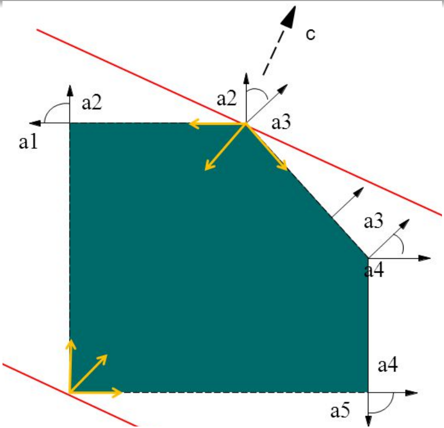
Characterisation of Optimality: Constrained Case.



$\max_x c^T x$ subject to $a_i^T x \leq b_i$

$$\nabla g_0(x) = \sum_i \lambda_i \nabla g_i(x), \lambda_i \geq 0$$

Characterisation of Optimality: Constrained Case.

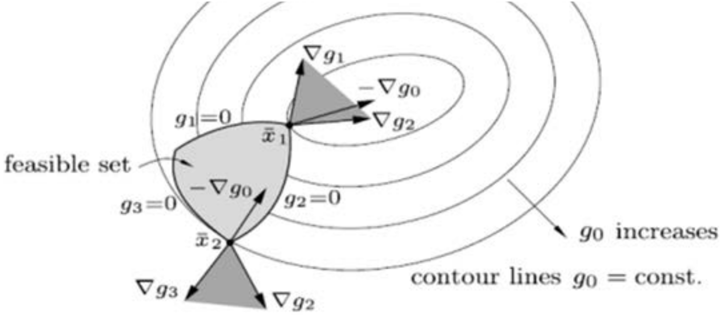


$\max_x c^T x$ subject to $a_i^T x \leq b_i$

$\nabla g_0(x) = \sum_i \lambda_i \nabla g_i(x), \lambda_i \geq 0$

Furthermore the characterisation is purely local.

Characterisation of Optimality: Constrained Case.



$\min_{x|g_i(x)\le 0} g_0(x)$ optimal if $\nabla g_0(x) = \sum_i \lambda_i \nabla g_i(x), \lambda_i \le 0$

Equivalent to a stationary point of the Lagrangian

$$L(x,\lambda) = g_0(x) - \sum_i \lambda_i g_i(x)$$

Characterisation of Optimality: Constrained Case.

We note that the characterisation

$$\nabla f(x) = \sum_i \lambda_i \nabla c_i(x)$$

Characterisation of Optimality: Constrained Case.

We note that the characterisation

$\nabla f(x) = \sum_i \lambda_i \nabla c_i(x)$

should not be disturbed by the constraints being quadratic.

Characterisation of Optimality: Constrained Case.

We note that the characterisation

$$\nabla f(x) = \sum_i \lambda_i \nabla c_i(x)$$

should not be disturbed by the constraints being quadratic.

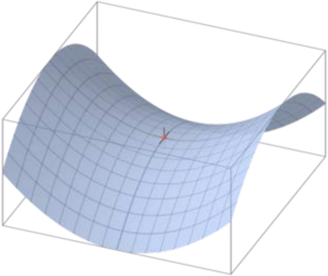
Although it suggests the Lagrangian is only locally meaningful!

Characterisation of Optimality: Constrained Case.

One issue is that the stationary point of the Lagrangian of interest is a saddle-point

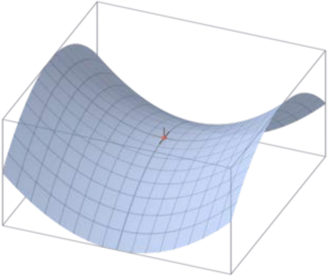
Characterisation of Optimality: Constrained Case.

One issue is that the stationary point of the Lagrangian of interest is a saddle-point



Characterisation of Optimality: Constrained Case.

One issue is that the stationary point of the Lagrangian of interest is a saddle-point



We are not very good at designing algorithms to find saddle points

Background

The intuitive idea of penalty functions:

$$\min_{x\in\mathbb{R}^n} \quad f(x)$$

subject to $\quad c_j(x) = 0, \quad j \in \{1, \ldots, m\}.$

The idea[**Courant, 1943**], was to replace it with

$$\min_{x\in\mathbb{R}^n} \quad p(x,\mu) = f(x) + \mu \sum_{j\in\{1,\ldots,m\}} c_j^2(x)$$

and then

$$\lim_{\mu\to\infty} x(\mu) = x^*$$

The disadvantage: Infinite number of minimizations

Background

The intuitive idea of penalty functions:

$$\min_{x\in\mathbb{R}^n} \quad f(x)$$

subject to $\quad c_j(x) = 0, \quad j \in \{1, \dots, m\}.$

The idea[**Courant, 1943**], was to replace it with

$$\min_{x\in\mathbb{R}^n} \quad p(x,\mu) = f(x) + \mu \sum_{j\in\{1,\dots,m\}} c_j^2(x)$$

and then

$$\lim_{\mu\to\infty} x(\mu) = x^*$$

The disadvantage: Infinite number of minimizations

Background

The intuitive idea of penalty functions:

$$\min_{x\in\mathbb{R}^n} \quad f(x)$$

subject to $\quad c_j(x) = 0, \quad j \in \{1, \ldots, m\}.$

The idea[**Courant**, **1943**], was to replace it with

$$\min_{x\in\mathbb{R}^n} \quad p(x,\mu) = f(x) + \mu \sum_{j\in\{1,\ldots,m\}} c_j^2(x)$$

and then

$$\lim_{\mu\to\infty} x(\mu) = x^*$$

The disadvantage: Infinite number of minimizations

Background

The intuitive idea of penalty functions:

$$\min_{x\in\mathbb{R}^n} \quad f(x)$$

subject to $\quad c_j(x) = 0, \quad j \in \{1, \dots, m\}.$

The idea[**Courant**, **1943**], was to replace it with

$$\min_{x\in\mathbb{R}^n} \quad p(x,\mu) = f(x) + \mu \sum_{j\in\{1,\dots,m\}} c_j^2(x)$$

and then

$$\lim_{\mu\to\infty} x(\mu) = x^*$$

The disadvantage: Infinite number of minimizations

Background

The intuitive idea of penalty functions:

$$\min_{x\in\mathbb{R}^n} \quad f(x)$$

subject to $\quad c_j(x) = 0, \quad j \in \{1, \ldots, m\}.$

The idea[**Courant**, **1943**], was to replace it with

$$\min_{x\in\mathbb{R}^n} \quad p(x,\mu) = f(x) + \mu \sum_{j\in\{1,\ldots,m\}} c_j^2(x)$$

and then

$$\lim_{\mu\to\infty} x(\mu) = x^*$$

The disadvantage: Infinite number of minimizations

The Augmented Lagrangian

The above motivated the idea of combining the Lagrangian with the penalty function

The augmented Lagrangian [Hestenes and Powell, 1969], is defined by $\min_{x \in \mathbb{R}^n} \quad \mathcal{L}(x, \lambda, \mu) =$

$f(x) + \sum_{i \in \{1,\ldots,m\}} \lambda_i c_i(x) + \mu \sum_{j \in \{1,\ldots,m\}} c_j^2(x)$

and then only one of μ and the λs needs to be updated

Furthermore, the gradient of the augmented Lagrange compared with the Lagrangian gives the λ update

The Augmented Lagrangian

The above motivated the idea of combining the Lagrangian with the penalty function

The augmented Lagrangian [Hestenes and Powell, 1969], is defined by $\min_{x \in \mathbb{R}^n} \quad \mathcal{L}(x, \lambda, \mu) =$

$f(x) + \sum_{i \in \{1,\ldots,m\}} \lambda_i c_i(x) + \mu \sum_{j \in \{1,\ldots,m\}} c_j^2(x)$

and then only one of μ and the λs needs to be updated

Furthermore, the gradient of the augmented Lagrange compared with the Lagrangian gives the λ update

The Augmented Lagrangian

The above motivated the idea of combining the Lagrangian with the penalty function

The augmented Lagrangian [Hestenes and Powell, 1969], is defined by $\min_{x \in \mathbb{R}^n} \quad \mathcal{L}(x, \lambda, \mu) =$
$f(x) + \sum_{i \in \{1,\ldots,m\}} \lambda_i c_i(x) + \mu \sum_{j \in \{1,\ldots,m\}} c_j^2(x)$

and then only one of μ and the λs needs to be updated

Furthermore, the gradient of the augmented Lagrange compared with the Lagrangian gives the λ update

The Augmented Lagrangian

The above motivated the idea of combining the Lagrangian with the penalty function

The augmented Lagrangian [Hestenes and Powell, 1969], is defined by $\min_{x \in \mathbb{R}^n} \quad \mathcal{L}(x, \lambda, \mu) =$

$$f(x) + \sum_{i \in \{1,\dots,m\}} \lambda_i c_i(x) + \mu \sum_{j \in \{1,\dots,m\}} c_j^2(x)$$

and then only one of μ and the λs needs to be updated

Furthermore, the gradient of the augmented Lagrange compared with the Lagrangian gives the λ update

The Augmented Lagrangian

$$\nabla \mathcal{L}(x, \lambda, \mu) =$$
$$\nabla f(x) + \sum_{i \in \{1,\ldots,m\}} \lambda_i \nabla c_i(x) + 2.0\ \mu \sum_{j \in \{1,\ldots,m\}} c_j(x)\ \nabla c_j(x)$$

which looking at the terms in $\nabla c_j(x)$ suggests the update formula,

$$\lambda_j^+ = \lambda_j + 2.0\ \mu c_j(x)$$

If not sufficiently feasible μ alone is increased, otherwise the λs alone are updated

and one can prove that eventually μ is never increased.

Perhaps the best known implementation is LANCELOT

The Augmented Lagrangian

$$\nabla \mathcal{L}(x, \lambda, \mu) =$$

$$\nabla f(x) + \sum_{i \in \{1,\ldots,m\}} \lambda_i \nabla c_i(x) + 2.0\ \mu \sum_{j \in \{1,\ldots,m\}} c_j(x)\ \nabla c_j(x)$$

which looking at the terms in $\nabla c_j(x)$ suggests the update formula,

$$\lambda_j^+ = \lambda_j + 2.0\ \mu c_j(x)$$

If not sufficiently feasible μ alone is increased, otherwise the λs alone are updated

and one can prove that eventually μ is never increased.

Perhaps the best known implementation is LANCELOT

The Augmented Lagrangian

$$\nabla \mathcal{L}(x, \lambda, \mu) =$$

$$\nabla f(x) + \sum_{i \in \{1,\ldots,m\}} \lambda_i \nabla c_i(x) + 2.0\ \mu \sum_{j \in \{1,\ldots,m\}} c_j(x)\ \nabla c_j(x)$$

which looking at the terms in $\nabla c_j(x)$ suggests the update formula,

$$\lambda_j^+ = \lambda_j + 2.0\ \mu c_j(x)$$

If not sufficiently feasible μ alone is increased, otherwise the λs alone are updated

and one can prove that eventually μ is never increased.

Perhaps the best known implementation is LANCELOT

The Augmented Lagrangian

$\nabla \mathcal{L}(x, \lambda, \mu) =$

$\nabla f(x) + \sum_{i \in \{1,\ldots,m\}} \lambda_i \nabla c_i(x) + 2.0\ \mu \sum_{j \in \{1,\ldots,m\}} c_j(x)\ \nabla c_j(x)$

which looking at the terms in $\nabla c_j(x)$ suggests the update formula,

$$\lambda_j^+ = \lambda_j + 2.0\ \mu c_j(x)$$

If not sufficiently feasible μ alone is increased, otherwise the λs alone are updated

and one can prove that eventually μ is never increased.

Perhaps the best known implementation is LANCELOT

The Augmented Lagrangian

$\nabla \mathcal{L}(x, \lambda, \mu) =$

$\nabla f(x) + \sum_{i \in \{1,\ldots,m\}} \lambda_i \nabla c_i(x) + 2.0\ \mu \sum_{j \in \{1,\ldots,m\}} c_j(x)\ \nabla c_j(x)$

which looking at the terms in $\nabla c_j(x)$ suggests the update formula,

$$\lambda_j^+ = \lambda_j + 2.0\ \mu c_j(x)$$

If not sufficiently feasible μ alone is increased, otherwise the λs alone are updated

and one can prove that eventually μ is never increased.

Perhaps the best known implementation is LANCELOT

Background

An **exact** penalty function.

The idea: [Zangwill, 1967], replace the constrained problem with

$$\min_{x \in \mathbb{R}^n} \quad p(x, \mu) = f(x) + \mu \sum_{j \in \{1,\dots,m\}} |c_j(x)|$$

and then

$$\mu < \mu^* \text{ implies that } x(\mu) = x^*$$

The disadvantage: Function is not differentiable (non-smooth)

Consequently considered to only be of theoretical interest

Background

An **exact** penalty function.

The idea: [Zangwill, 1967], replace the constrained problem with

$$\min_{x \in \mathbb{R}^n} \quad p(x, \mu) = f(x) + \mu \sum_{j \in \{1,\dots,m\}} |c_j(x)|$$

and then

$$\mu < \mu^* \text{ implies that } x(\mu) = x^*$$

The disadvantage: Function is not differentiable (non-smooth)

Consequently considered to only be of theoretical interest

Background

An **exact** penalty function.

The idea: [Zangwill, 1967], replace the constrained problem with

$$\min_{x\in\mathbb{R}^n} \quad p(x,\mu) = f(x) + \mu \sum_{j\in\{1,\ldots,m\}} |c_j(x)|$$

and then

$$\mu < \mu^* \text{ implies that } x(\mu) = x^*$$

The disadvantage: Function is not differentiable (non-smooth)

Consequently considered to only be of theoretical interest

Background

An **exact** penalty function.

The idea: [Zangwill, 1967], replace the constrained problem with

$$\min_{x \in \mathbb{R}^n} \quad p(x,\mu) = f(x) + \mu \sum_{j \in \{1,\ldots,m\}} |c_j(x)|$$

and then

$$\mu < \mu^* \text{ implies that } x(\mu) = x^*$$

The disadvantage: Function is not differentiable (non-smooth)

Consequently considered to only be of theoretical interest

Background

An **exact** penalty function.

The idea: [Zangwill, 1967], replace the constrained problem with

$$\min_{x \in \mathbb{R}^n} \quad p(x, \mu) = f(x) + \mu \sum_{j \in \{1,\ldots,m\}} |c_j(x)|$$

and then

$$\mu < \mu^* \text{ implies that } x(\mu) = x^*$$

The disadvantage: Function is not differentiable (non-smooth)

Consequently considered to only be of theoretical interest

Claim: The paradigm constrained problem should be:

Optimize a quadratic subject to **quadratic** constraints

Motivated by the statements

The derivative of a quadratic is linear

and

All computational mathematics is essentially linear.

Claim: The paradigm constrained problem should be:

Optimize a quadratic subject to **quadratic** constraints

Motivated by the statements

The derivative of a quadratic is linear

and

All computational mathematics is essentially linear.

Claim: The paradigm constrained problem should be:

Optimize a quadratic subject to **quadratic** constraints

Motivated by the statements

The derivative of a quadratic is linear

and

All computational mathematics is essentially linear.

Claim: The paradigm constrained problem should be:

Optimize a quadratic subject to **quadratic** constraints

Motivated by the statements

The derivative of a quadratic is linear

and

All computational mathematics is essentially linear.

We note that the characterisation

$$\nabla f(x) = \sum_i \lambda_i \nabla c_i(x)$$

should not be disturbed by the constraints being quadratic.

Significant observation

If I have quadratic constraints

the quadratic penalty function is a **quartic**.

but

the exact penalty function is **piecewise quadratic**.

CLAIM:
Minimizing a piecewise quadratic is **almost** as easy as a quadratic.

Significant observation

If I have quadratic constraints

the quadratic penalty function is a **quartic**.

but

the exact penalty function is **piecewise quadratic**.

CLAIM:
Minimizing a piecewise quadratic is **almost** as easy as a quadratic.

Significant observation

If I have quadratic constraints

the quadratic penalty function is a **quartic**.

but

the exact penalty function is **piecewise quadratic**.

CLAIM:
Minimizing a piecewise quadratic is **almost** as easy as a quadratic.

Significant observation

If I have quadratic constraints

the quadratic penalty function is a **quartic**.

but

the exact penalty function is **piecewise quadratic**.

CLAIM:
Minimizing a piecewise quadratic is **almost** as easy as a quadratic.

Significant observation

If I have quadratic constraints

the quadratic penalty function is a **quartic**.

but

the exact penalty function is **piecewise quadratic**.

CLAIM:
Minimizing a piecewise quadratic is **almost** as easy as a quadratic.

Projection onto a Hyperplane

Assume that the columns of A, a_i, $j \in \{1, \ldots, m\}$ are a **basis** of normals for the hyperplane in question

Define the projection operator P by

$$P = I - A(A^\top A)^{-1} A^\top$$

Note that, assuming that $A^\top N = 0$,

$$PA = 0 \quad \text{and} \quad PN = N,$$

where the columns of N are the generators for the hyperplane.

So Px projects x **on** to the hyperplane.

In practise one uses the orthogonal decomposition, QR, to compute the projection

Projection onto a Hyperplane

Assume that the columns of A, a_i, $j \in \{1, \ldots, m\}$ are a **basis** of normals for the hyperplane in question

Define the projection operator P by

$$P = I - A(A^\top A)^{-1} A^\top$$

Note that, assuming that $A^\top N = 0$,

$$PA = 0 \quad \text{and} \quad PN = N,$$

where the columns of N are the generators for the hyperplane.

So Px projects x **on** to the hyperplane.

In practise one uses the orthogonal decomposition, QR, to compute the projection

Projection onto a Hyperplane

Assume that the columns of A, a_i, $j \in \{1, \ldots, m\}$ are a **basis** of normals for the hyperplane in question

Define the projection operator P by

$$P = I - A(A^\top A)^{-1} A^\top$$

Note that, assuming that $A^\top N = 0$,

$$PA = 0 \quad \text{and} \quad PN = N,$$

where the columns of N are the generators for the hyperplane.

So Px projects x **on** to the hyperplane.

In practise one uses the orthogonal decomposition, QR, to compute the projection

Projection onto a Hyperplane

Assume that the columns of A, a_i, $j \in \{1, \ldots, m\}$ are a **basis** of normals for the hyperplane in question

Define the projection operator P by

$$P = I - A(A^\top A)^{-1} A^\top$$

Note that, assuming that $A^\top N = 0$,

$$PA = 0 \quad \text{and} \quad PN = N,$$

where the columns of N are the generators for the hyperplane.

So Px projects x **on** to the hyperplane.

In practise one uses the orthogonal decomposition, QR, to compute the projection

Projection onto a Hyperplane

Assume that the columns of A, a_i, $j \in \{1, \ldots, m\}$ are a **basis** of normals for the hyperplane in question

Define the projection operator P by

$$P = I - A(A^\top A)^{-1} A^\top$$

Note that, assuming that $A^\top N = 0$,

$$PA = 0 \quad \text{and} \quad PN = N,$$

where the columns of N are the generators for the hyperplane.

So Px projects x **on** to the hyperplane.

In practise one uses the orthogonal decomposition, QR, to compute the projection

Making an Exact Penalty Function Practical

Basic idea was my Ph.D thesis [1971]

Consider $\min_{x \in \mathbb{R}^n} \quad f(x) \quad$ subject to $\quad c_j(x) \leq 0, \quad j \in I.$

The penalty function $p(x,\mu) = f(x) + \mu \sum_{j \in I} \max\{c_j(x), 0\}$ is only **non-differentiable in the neighbourhood of active constraints**

Define $A_\epsilon = \{j \in I : |c_j^k(x)| \leq \epsilon\}$ and $V_\epsilon = \{j \in I : c_j^k(x) > \epsilon\}$

So the idea is to define $r(x,\mu) = f(x) + \mu \sum_{j \in I \setminus A_\epsilon} \max\{c_j(x), 0\}$

or equivalently $r(x,\mu) = f(x) + \mu \sum_{j \in V_\epsilon} c_j(x)$,

which is locally differentiable.

Making an Exact Penalty Function Practical

Basic idea was my Ph.D thesis [1971]

Consider $\min_{x \in \mathbb{R}^n} \quad f(x) \quad$ subject to $\quad c_j(x) \leq 0, \quad j \in I.$

The penalty function $p(x, \mu) = f(x) + \mu \sum_{j \in I} \max\{c_j(x), 0\}$ is only **non-differentiable in the neighbourhood of active constraints**

Define $A_\epsilon = \{j \in I : |c_j^k(x)| \leq \epsilon\}$ and $V_\epsilon = \{j \in I : c_j^k(x) > \epsilon\}$

So the idea is to define $r(x, \mu) = f(x) + \mu \sum_{j \in I \setminus A_\epsilon} \max\{c_j(x), 0\}$

or equivalently $r(x, \mu) = f(x) + \mu \sum_{j \in V_\epsilon} c_j(x)$,

which is locally differentiable.

Making an Exact Penalty Function Practical

Basic idea was my Ph.D thesis [1971]

Consider $\min_{x \in \mathbb{R}^n} \quad f(x) \quad$ subject to $\quad c_j(x) \leq 0, \quad j \in I.$

The penalty function $p(x, \mu) = f(x) + \mu \sum_{j \in I} \max\{c_j(x), 0\}$ is only **non-differentiable in the neighbourhood of active constraints**

Define $A_\epsilon = \{j \in I : |c_j^k(x)| \leq \epsilon\}$ and $V_\epsilon = \{j \in I : c_j^k(x) > \epsilon\}$

So the idea is to define $r(x, \mu) = f(x) + \mu \sum_{j \in I \setminus A_\epsilon} \max\{c_j(x), 0\}$

or equivalently $r(x, \mu) = f(x) + \mu \sum_{j \in V_\epsilon} c_j(x),$

which is locally differentiable.

Making an Exact Penalty Function Practical

Basic idea was my Ph.D thesis [1971]

Consider $\min_{x \in \mathbb{R}^n} \quad f(x) \quad$ subject to $\quad c_j(x) \leq 0, \quad j \in I.$

The penalty function $p(x, \mu) = f(x) + \mu \sum_{j \in I} \max\{c_j(x), 0\}$ is only **non-differentiable in the neighbourhood of active constraints**

Define $A_\epsilon = \{j \in I : |c_j^k(x)| \leq \epsilon\}$ and $V_\epsilon = \{j \in I : c_j^k(x) > \epsilon\}$

So the idea is to define $r(x, \mu) = f(x) + \mu \sum_{j \in I \setminus A_\epsilon} \max\{c_j(x), 0\}$

or equivalently $r(x, \mu) = f(x) + \mu \sum_{j \in V_\epsilon} c_j(x),$

which is locally differentiable.

Making an Exact Penalty Function Practical

Basic idea was my Ph.D thesis [1971]

Consider $\min_{x\in\mathbb{R}^n} \quad f(x) \quad$ subject to $\quad c_j(x) \le 0, \quad j \in I.$

The penalty function $p(x,\mu) = f(x) + \mu \sum_{j\in I} \max\{c_j(x), 0\}$ is only **non-differentiable in the neighbourhood of active constraints**

Define $A_\epsilon = \{j \in I : |c_j^k(x)| \le \epsilon\}$ and $V_\epsilon = \{j \in I : c_j^k(x) > \epsilon\}$

So the idea is to define $r(x,\mu) = f(x) + \mu \sum_{j\in I\setminus A_\epsilon} \max\{c_j(x), 0\}$

or equivalently $r(x,\mu) = f(x) + \mu \sum_{j\in V_\epsilon} c_j(x),$

which is locally differentiable.

Making an Exact Penalty Function Practical

Basic idea was my Ph.D thesis [1971]

Consider $\min_{x \in \mathbb{R}^n} \quad f(x) \quad$ subject to $\quad c_j(x) \leq 0, \quad j \in I.$

The penalty function $p(x,\mu) = f(x) + \mu \sum_{j \in I} \max\{c_j(x), 0\}$ is only **non-differentiable in the neighbourhood of active constraints**

Define $A_\epsilon = \{j \in I : |c_j^k(x)| \leq \epsilon\}$ and $V_\epsilon = \{j \in I : c_j^k(x) > \epsilon\}$

So the idea is to define $r(x,\mu) = f(x) + \mu \sum_{j \in I \setminus A_\epsilon} \max\{c_j(x), 0\}$

or equivalently $r(x,\mu) = f(x) + \mu \sum_{j \in V_\epsilon} c_j(x)$,

which is locally differentiable.

Making an Exact Penalty Function Practical

Basic idea was my Ph.D thesis [1971]

Consider $\min_{x \in \mathbb{R}^n} \quad f(x) \quad$ subject to $\quad c_j(x) \leq 0, \quad j \in I.$

The penalty function $p(x, \mu) = f(x) + \mu \sum_{j \in I} \max\{c_j(x), 0\}$ is only **non-differentiable in the neighbourhood of active constraints**

Define $A_\epsilon = \{j \in I : |c_j^k(x)| \leq \epsilon\}$ and $V_\epsilon = \{j \in I : c_j^k(x) > \epsilon\}$

So the idea is to define $r(x, \mu) = f(x) + \mu \sum_{j \in I \setminus A_\epsilon} \max\{c_j(x), 0\}$

or equivalently $r(x, \mu) = f(x) + \mu \sum_{j \in V_\epsilon} c_j(x)$,

which is locally differentiable.

A Practical First-Order Method

Base the search direction on $r(x,\mu)$ by projecting orthogonal to the space spanned by $\nabla c_i(x), i \in A_\epsilon$

In other words, we have completely accurate information about the change in p if we take our fundamental subproblem as
$\min_{d \in \mathbb{R}^n} \quad r(x+d,\mu) \quad$ subject to $\quad c_j(x+d) = c_j(x), \quad j \in A_\epsilon.$

So, up to first-order define $d = -P\nabla r(x,\mu)$ orthogonal to the space spanned by $\nabla c_j(x), \quad j \in A_\epsilon.$

Important Observation 1: If $-P\nabla r(x,\mu)$ is small then $\nabla r(x,\mu) \approx \sum_{j \in A_\epsilon} \lambda_j \nabla c_j(x)$ and we can obtain a good estimate for the λ's. (which connects up with the "local" issue)

This tells us if and how we can obtain descent by releasing a single activity **and gives us optimality conditions**.

Observation 2: Eventually make ϵ-active constraints active

A Practical First-Order Method

Base the search direction on $r(x,\mu)$ by projecting orthogonal to the space spanned by $\nabla c_i(x), i \in A_\epsilon$

In other words, we have completely accurate information about the change in p if we take our fundamental subproblem as
$\min_{d \in \mathbb{R}^n} \quad r(x+d,\mu) \quad$ subject to $\quad c_j(x+d) = c_j(x), \quad j \in A_\epsilon.$

So, up to first-order define $d = -P\nabla r(x,\mu)$ orthogonal to the space spanned by $\nabla c_j(x), \quad j \in A_\epsilon.$

Important Observation 1: If $-P\nabla r(x,\mu)$ is small then $\nabla r(x,\mu) \approx \sum_{j \in A_\epsilon} \lambda_j \nabla c_j(x)$ and we can obtain a good estimate for the λ's. (which connects up with the "local" issue)

This tells us if and how we can obtain descent by releasing a single activity **and gives us optimality conditions**.

Observation 2: Eventually make ϵ-active constraints active

A Practical First-Order Method

Base the search direction on $r(x,\mu)$ by projecting orthogonal to the space spanned by $\nabla c_i(x), i \in A_\epsilon$

In other words, we have completely accurate information about the change in p if we take our fundamental subproblem as
$\min_{d \in \mathbb{R}^n} \quad r(x+d,\mu) \quad$ subject to $\quad c_j(x+d) = c_j(x), \quad j \in A_\epsilon.$

So, up to first-order define $d = -P\nabla r(x,\mu)$ orthogonal to the space spanned by $\nabla c_j(x), \quad j \in A_\epsilon.$

Important Observation 1: If $-P\nabla r(x,\mu)$ is small then $\nabla r(x,\mu) \approx \sum_{j \in A_\epsilon} \lambda_j \nabla c_j(x)$ and we can obtain a good estimate for the λ's. (which connects up with the "local" issue)

This tells us if and how we can obtain descent by releasing a single activity **and gives us optimality conditions**.

Observation 2: Eventually make ϵ-active constraints active

A Practical First-Order Method

Base the search direction on $r(x,\mu)$ by projecting orthogonal to the space spanned by $\nabla c_i(x), i \in A_\epsilon$

In other words, we have completely accurate information about the change in p if we take our fundamental subproblem as
$\min_{d\in\mathbb{R}^n} \quad r(x+d,\mu) \quad$ subject to $\quad c_j(x+d) = c_j(x), \quad j \in A_\epsilon.$

So, up to first-order define $d = -P\nabla r(x,\mu)$ orthogonal to the space spanned by $\nabla c_j(x), \quad j \in A_\epsilon.$

Important Observation 1: If $-P\nabla r(x,\mu)$ is small then $\nabla r(x,\mu) \approx \sum_{j\in A_\epsilon} \lambda_j \nabla c_j(x)$ and we can obtain a good estimate for the λ's. (which connects up with the "local" issue)

This tells us if and how we can obtain descent by releasing a single activity **and gives us optimality conditions**.

Observation 2: Eventually make ϵ-active constraints active

A Practical First-Order Method

Base the search direction on $r(x, \mu)$ by projecting orthogonal to the space spanned by $\nabla c_i(x), i \in A_\epsilon$

In other words, we have completely accurate information about the change in p if we take our fundamental subproblem as
$\min_{d \in \mathbb{R}^n} \quad r(x+d, \mu) \quad$ subject to $\quad c_j(x+d) = c_j(x), \quad j \in A_\epsilon.$

So, up to first-order define $d = -P\nabla r(x, \mu)$ orthogonal to the space spanned by $\nabla c_j(x), \quad j \in A_\epsilon.$

Important Observation 1: If $-P\nabla r(x, \mu)$ is small then $\nabla r(x, \mu) \approx \sum_{j \in A_\epsilon} \lambda_j \nabla c_j(x)$ and we can obtain a good estimate for the λ's. (which connects up with the "local" issue)

This tells us if and how we can obtain descent by releasing a single activity **and gives us optimality conditions**.

Observation 2: Eventually make ϵ-active constraints active

A Practical First-Order Method

Base the search direction on $r(x, \mu)$ by projecting orthogonal to the space spanned by $\nabla c_i(x), i \in A_\epsilon$

In other words, we have completely accurate information about the change in p if we take our fundamental subproblem as
$\min_{d \in \mathbb{R}^n} \quad r(x + d, \mu) \quad$ subject to $\quad c_j(x + d) = c_j(x), \quad j \in A_\epsilon.$

So, up to first-order define $d = -P\nabla r(x, \mu)$ orthogonal to the space spanned by $\nabla c_j(x), \quad j \in A_\epsilon.$

Important Observation 1: If $-P\nabla r(x, \mu)$ is small then $\nabla r(x, \mu) \approx \sum_{j \in A_\epsilon} \lambda_j \nabla c_j(x)$ and we can obtain a good estimate for the λ's. (which connects up with the "local" issue)

This tells us if and how we can obtain descent by releasing a single activity **and gives us optimality conditions**.

Observation 2: Eventually make ϵ-active constraints active

A Practical First-Order Method

Suppose $\nabla r(x,\mu) = \sum_{j \in A_\epsilon} \lambda_j \nabla c_j(x)$.

Consider dropping one activity, j say, from the projection and projecting $-\nabla r(x,\mu)$

So, $d = -P_j \nabla r(x,\mu) = -\lambda_j P_j \nabla c_j(x)$ and

$d^\top \nabla r(x,\mu) = -\nabla r(x,\mu)^\top P_j \nabla r(x,\mu) = -\|P_j \nabla r(x,\mu)\|^2$.

Also $d^\top \nabla c_j(x) = -\lambda_j \|P_j \nabla c_j(x)\|^2 = -\frac{1}{\lambda_j}\|P_j \nabla r(x,\mu)\|^2$.

So d increases (violates) the activity c_j if and only if $\lambda_j < 0$ and then decreases the penalty function if and only if

$$-1 - \frac{\mu}{\lambda_j} < 0 \qquad \text{i.e} \qquad \lambda_j < -\mu$$

or alternatively (i.e. without violation) if and only if $\lambda_j > 0$

A Practical First-Order Method

Suppose $\nabla r(x,\mu) = \sum_{j \in A_\epsilon} \lambda_j \nabla c_j(x)$.

Consider dropping one activity, j say, from the projection and projecting $-\nabla r(x,\mu)$

So, $d = -P_j \nabla r(x,\mu) = -\lambda_j P_j \nabla c_j(x)$ and
$d^\top \nabla r(x,\mu) = -\nabla r(x,\mu)^\top P_j \nabla r(x,\mu) = -\|P_j \nabla r(x,\mu)\|^2$.

Also $d^\top \nabla c_j(x) = -\lambda_j \|P_j \nabla c_j(x)\|^2 = -\frac{1}{\lambda_j}\|P_j \nabla r(x,\mu)\|^2$.

So d increases (violates) the activity c_j if and only if $\lambda_j < 0$ and then decreases the penalty function if and only if

$$-1 - \frac{\mu}{\lambda_j} < 0 \qquad \text{i.e} \qquad \lambda_j < -\mu$$

or alternatively (i.e. without violation) if and only if $\lambda_j > 0$

A Practical First-Order Method

Suppose $\nabla r(x,\mu) = \sum_{j \in A_\epsilon} \lambda_j \nabla c_j(x)$.

Consider dropping one activity, j say, from the projection and projecting $-\nabla r(x,\mu)$

So, $d = -P_j \nabla r(x,\mu) = -\lambda_j P_j \nabla c_j(x)$ and

$d^\top \nabla r(x,\mu) = -\nabla r(x,\mu)^\top P_j \nabla r(x,\mu) = -\|P_j \nabla r(x,\mu)\|^2$.

Also $d^\top \nabla c_j(x) = -\lambda_j \|P_j \nabla c_j(x)\|^2 = -\frac{1}{\lambda_j}\|P_j \nabla r(x,\mu)\|^2$.

So d increases (violates) the activity c_j if and only if $\lambda_j < 0$ and then decreases the penalty function if and only if

$$-1 - \frac{\mu}{\lambda_j} < 0 \qquad \text{i.e} \qquad \lambda_j < -\mu$$

or alternatively (i.e. without violation) if and only if $\lambda_j > 0$

A Practical First-Order Method

Suppose $\nabla r(x,\mu) = \sum_{j \in A_\epsilon} \lambda_j \nabla c_j(x)$.

Consider dropping one activity, j say, from the projection and projecting $-\nabla r(x,\mu)$

So, $d = -P_j \nabla r(x,\mu) = -\lambda_j P_j \nabla c_j(x)$ and

$$d^\top \nabla r(x,\mu) = -\nabla r(x,\mu)^\top P_j \nabla r(x,\mu) = -\|P_j \nabla r(x,\mu)\|^2.$$

Also $d^\top \nabla c_j(x) = -\lambda_j \|P_j \nabla c_j(x)\|^2 = -\frac{1}{\lambda_j} \|P_j \nabla r(x,\mu)\|^2.$

So d increases (violates) the activity c_j if and only if $\lambda_j < 0$ and then decreases the penalty function if and only if

$$-1 - \frac{\mu}{\lambda_j} < 0 \qquad \text{i.e} \qquad \lambda_j < -\mu$$

or alternatively (i.e. without violation) if and only if $\lambda_j > 0$

A Practical First-Order Method

Suppose $\nabla r(x,\mu) = \sum_{j \in A_\epsilon} \lambda_j \nabla c_j(x)$.

Consider dropping one activity, j say, from the projection and projecting $-\nabla r(x,\mu)$

So, $d = -P_j \nabla r(x,\mu) = -\lambda_j P_j \nabla c_j(x)$ and

$$d^\top \nabla r(x,\mu) = -\nabla r(x,\mu)^\top P_j \nabla r(x,\mu) = -\|P_j \nabla r(x,\mu)\|^2.$$

Also $d^\top \nabla c_j(x) = -\lambda_j \|P_j \nabla c_j(x)\|^2 = -\frac{1}{\lambda_j} \|P_j \nabla r(x,\mu)\|^2$.

So d increases (violates) the activity c_j if and only if $\lambda_j < 0$ and then decreases the penalty function if and only if

$$-1 - \frac{\mu}{\lambda_j} < 0 \qquad \text{i.e} \qquad \lambda_j < -\mu$$

or alternatively (i.e. without violation) if and only if $\lambda_j > 0$

A Practical First-Order Method

Suppose $\nabla r(x,\mu) = \sum_{j \in A_\epsilon} \lambda_j \nabla c_j(x)$.

Consider dropping one activity, j say, from the projection and projecting $-\nabla r(x,\mu)$

So, $d = -P_j \nabla r(x,\mu) = -\lambda_j P_j \nabla c_j(x)$ and

$$d^\top \nabla r(x,\mu) = -\nabla r(x,\mu)^\top P_j \nabla r(x,\mu) = -\|P_j \nabla r(x,\mu)\|^2.$$

Also $d^\top \nabla c_j(x) = -\lambda_j \|P_j \nabla c_j(x)\|^2 = -\frac{1}{\lambda_j}\|P_j \nabla r(x,\mu)\|^2$.

So d increases (violates) the activity c_j if and only if $\lambda_j < 0$ and then decreases the penalty function if and only if

$$-1 - \frac{\mu}{\lambda_j} < 0 \qquad \text{i.e} \qquad \lambda_j < -\mu$$

or alternatively (i.e. without violation) if and only if $\lambda_j > 0$

A Practical First-Order Method

Suppose $\nabla r(x,\mu) = \sum_{j \in A_\epsilon} \lambda_j \nabla c_j(x)$.

Consider dropping one activity, j say, from the projection and projecting $-\nabla r(x,\mu)$

So, $d = -P_j \nabla r(x,\mu) = -\lambda_j P_j \nabla c_j(x)$ and

$$d^\top \nabla r(x,\mu) = -\nabla r(x,\mu)^\top P_j \nabla r(x,\mu) = -\|P_j \nabla r(x,\mu)\|^2.$$

Also $d^\top \nabla c_j(x) = -\lambda_j \|P_j \nabla c_j(x)\|^2 = -\frac{1}{\lambda_j}\|P_j \nabla r(x,\mu)\|^2$.

So d increases (violates) the activity c_j if and only if $\lambda_j < 0$ and then decreases the penalty function if and only if

$$-1 - \frac{\mu}{\lambda_j} < 0 \qquad \text{i.e} \qquad \lambda_j < -\mu$$

or alternatively (i.e. without violation) if and only if $\lambda_j > 0$

A Practical Second-Order Exact Penalty Function

Now, the subproblem we really wish to solve is

$$\min_{d \in \mathbb{R}^n} \quad \nabla_x r(\tilde{x}, \mu)^\top d + \frac{1}{2} d^\top \nabla_{xx} r(\tilde{x}, \mu) d$$

$$\text{subject to} \quad \nabla c_j(\tilde{x})^\top d + \tfrac{1}{2} d^\top \nabla^2 c_j(\tilde{x}) d = 0, \quad j \in A_\epsilon.$$

In other words, we have completely accurate information about the change in p up to second order

We can improve the activity values via

$$J_{A_\epsilon}(\tilde{x})^\top v = -\phi_{A_\epsilon}(\tilde{x} + d)$$

with $\phi_{A_\epsilon}(x) = [c_j(x)]_{j \in A_\epsilon}$ and $J_{A_\epsilon}(x) = [\nabla c_j(x)]_{j \in A_\epsilon}$.

A Practical Second-Order Exact Penalty Function

Now, the subproblem we really wish to solve is

$$\min_{d \in \mathbb{R}^n} \quad \nabla_x r(\tilde{x}, \mu)^\top d + \frac{1}{2} d^\top \nabla_{xx} r(\tilde{x}, \mu) d$$

$$\text{subject to} \quad \nabla c_j(\tilde{x})^\top d + \tfrac{1}{2} d^\top \nabla^2 c_j(\tilde{x}) d = 0, \quad j \in A_\epsilon.$$

In other words, we have completely accurate information about the change in p up to second order

We can improve the activity values via

$$J_{A_\epsilon}(\tilde{x})^\top v = -\phi_{A_\epsilon}(\tilde{x} + d)$$

with $\phi_{A_\epsilon}(x) = [c_j(x)]_{j \in A_\epsilon}$ and $J_{A_\epsilon}(x) = [\nabla c_j(x)]_{j \in A_\epsilon}$.

A Practical Second-Order Exact Penalty Function

Now, the subproblem we really wish to solve is

$$\min_{d \in \mathbb{R}^n} \quad \nabla_x r(\tilde{x}, \mu)^\top d + \frac{1}{2} d^\top \nabla_{xx} r(\tilde{x}, \mu) d$$

$$\text{subject to} \quad \nabla c_j(\tilde{x})^\top d + \tfrac{1}{2} d^\top \nabla^2 c_j(\tilde{x}) d = 0, \quad j \in A_\epsilon.$$

In other words, we have completely accurate information about the change in p up to second order

We can improve the activity values via

$$J_{A_\epsilon}(\tilde{x})^\top v = -\phi_{A_\epsilon}(\tilde{x} + d)$$

with $\phi_{A_\epsilon}(x) = [c_j(x)]_{j \in A_\epsilon}$ and $J_{A_\epsilon}(x) = [\nabla c_j(x)]_{j \in A_\epsilon}$.

A Practical Second-Order Exact Penalty Function

Now, the subproblem we really wish to solve is

$$\begin{array}{ll} \min\limits_{d \in \mathbb{R}^n} & \nabla_x r(\tilde{x}, \mu)^\top d + \frac{1}{2} d^\top \nabla_{xx} r(\tilde{x}, \mu) d \\ \text{subject to} & \nabla c_j(\tilde{x})^\top d + \tfrac{1}{2} d^\top \nabla^2 c_j(\tilde{x}) d = 0, \quad j \in A_\epsilon. \end{array}$$

In other words, we have completely accurate information about the change in p up to second order

We can improve the activity values via

$$J_{A_\epsilon}(\tilde{x})^\top v = -\phi_{A_\epsilon}(\tilde{x} + d)$$

with $\phi_{A_\epsilon}(x) = [c_j(x)]_{j \in A_\epsilon}$ and $J_{A_\epsilon}(x) = [\nabla c_j(x)]_{j \in A_\epsilon}$.

A Practical Second-Order Method:Observations

We know how to solve

$$\min_{d \in \mathbb{R}^n} \quad \nabla_x r(\tilde{x}, \mu)^\top d + \frac{1}{2} d^\top \nabla_{xx} r(\tilde{x}, \mu) d$$

$$\text{subject to} \quad \nabla c_j(\tilde{x})^\top d = 0, \quad j \in A_\epsilon.$$

Claim: If $-P\nabla r(x, \mu)$ is large then we can ignore the constraint curvature

We already observed that if $-P\nabla r(x, \mu)$ is small then we can obtain a good estimate for the λ's in.

$$\nabla r(x, \mu) \approx \sum_{j \in A_\epsilon} \lambda_j \nabla c_j(x).$$

What if we cannot ignore the constraint curvature?

A Practical Second-Order Method:Observations

We know how to solve

$$\min_{d\in\mathbb{R}^n} \quad \nabla_x r(\tilde{x},\mu)^\top d + \frac{1}{2} d^\top \nabla_{xx} r(\tilde{x},\mu) d$$

$$\text{subject to} \quad \nabla c_j(\tilde{x})^\top d = 0, \quad j \in A_\epsilon.$$

Claim: If $-P\nabla r(x,\mu)$ is large then we can ignore the constraint curvature

We already observed that if $-P\nabla r(x,\mu)$ is small then we can obtain a good estimate for the λ's in.

$$\nabla r(x,\mu) \approx \sum_{j\in A_\epsilon} \lambda_j \nabla c_j(x).$$

What if we cannot ignore the constraint curvature?

A Practical Second-Order Method:Observations

We know how to solve

$$\min_{d\in\mathbb{R}^n} \quad \nabla_x r(\tilde{x},\mu)^\top d + \frac{1}{2} d^\top \nabla_{xx} r(\tilde{x},\mu) d$$

$$\text{subject to} \quad \nabla c_j(\tilde{x})^\top d = 0, \quad j \in A_\epsilon.$$

Claim: If $-P\nabla r(x,\mu)$ is large then we can ignore the constraint curvature

We already observed that if $-P\nabla r(x,\mu)$ is small then we can obtain a good estimate for the λ's in.

$$\nabla r(x,\mu) \approx \textstyle\sum_{j\in A_\epsilon} \lambda_j \nabla c_j(x).$$

What if we cannot ignore the constraint curvature?

A Practical Second-Order Method:Observations

We know how to solve

$$\min_{d\in\mathbb{R}^n} \quad \nabla_x r(\tilde{x},\mu)^\top d + \frac{1}{2} d^\top \nabla_{xx} r(\tilde{x},\mu) d$$

$$\text{subject to} \quad \nabla c_j(\tilde{x})^\top d = 0, \quad j \in A_\epsilon.$$

Claim: If $-P\nabla r(x,\mu)$ is large then we can ignore the constraint curvature

We already observed that if $-P\nabla r(x,\mu)$ is small then we can obtain a good estimate for the λ's in.

$$\nabla r(x,\mu) \approx \sum_{j\in A_\epsilon} \lambda_j \nabla c_j(x).$$

What if we cannot ignore the constraint curvature?

A Practical Second-Order Method:Observations

We know how to solve

$$\min_{d\in\mathbb{R}^n} \quad \nabla_x r(\tilde{x},\mu)^\top d + \frac{1}{2} d^\top \nabla_{xx} r(\tilde{x},\mu) d$$

$$\text{subject to} \quad \nabla c_j(\tilde{x})^\top d = 0, \quad j \in A_\epsilon.$$

Claim: If $-P\nabla r(x,\mu)$ is large then we can ignore the constraint curvature

We already observed that if $-P\nabla r(x,\mu)$ is small then we can obtain a good estimate for the λ's in.

$$\nabla r(x,\mu) \approx \sum_{j\in A_\epsilon} \lambda_j \nabla c_j(x).$$

What if we cannot ignore the constraint curvature?

A Practical Second-Order Exact Penalty Function

Consider the KKT conditions for

$$\begin{array}{lll} \min\limits_{d\in\mathbb{R}^n} & \nabla_x r(\tilde{x},\mu)^\top d + \frac{1}{2} d^\top \nabla_{xx} r(\tilde{x},\mu) d & \\ \text{subject to} & \nabla c_j(\tilde{x})^\top d + \frac{1}{2} d^\top \nabla^2 c_j(\tilde{x}) d = 0, & j \in A_\epsilon. \end{array}$$

We need

$$\nabla_x r(\tilde{x},\mu) + \nabla_{xx} r(\tilde{x},\mu) d = \sum_{j\in A_\epsilon} \lambda_j \left\{ \nabla c_j(\tilde{x}) + \nabla_{xx} c_j(\tilde{x},\mu) d \right\}$$

Or using our good lambda estimate, $\hat{\lambda}$

$$\nabla_x r(\tilde{x},\mu) + \nabla_{xx} r(\tilde{x},\mu) d \approx \sum_{j\in A_\epsilon} \left\{ \lambda_j \nabla c_j(\tilde{x}) + \hat{\lambda}_j \nabla_{xx} c_j^k(\tilde{x},\mu) d \right\} .$$

Compare with KKT for

$$\begin{array}{lll} \min\limits_{d\in\mathbb{R}^n} & \nabla_x r(\tilde{x},\mu)^\top d + \frac{1}{2} d^\top \left(\nabla_{xx} r(\tilde{x},\mu) - \hat{\lambda} \nabla_{xx} c_j^k(\tilde{x},\mu) \right) d & \\ \text{subject to} & \nabla c_j(\tilde{x})^\top d = 0, & j \in A_\epsilon. \end{array}$$

A Practical Second-Order Exact Penalty Function

Consider the KKT conditions for

$$\begin{aligned}
\min_{d\in\mathbb{R}^n} \quad & \nabla_x r(\tilde{x},\mu)^\top d + \tfrac{1}{2} d^\top \nabla_{xx} r(\tilde{x},\mu) d \\
\text{subject to} \quad & \nabla c_j(\tilde{x})^\top d + \tfrac{1}{2} d^\top \nabla^2 c_j(\tilde{x}) d = 0, \quad j \in A_\epsilon.
\end{aligned}$$

We need

$$\nabla_x r(\tilde{x},\mu) + \nabla_{xx} r(\tilde{x},\mu) d = \textstyle\sum_{j\in A_\epsilon} \lambda_j \left\{ \nabla c_j(\tilde{x}) + \nabla_{xx} c_j(\tilde{x},\mu) d \right\}$$

Or using our good lambda estimate, $\hat{\lambda}$

$$\nabla_x r(\tilde{x},\mu) + \nabla_{xx} r(\tilde{x},\mu) d \approx \textstyle\sum_{j\in A_\epsilon} \left\{ \lambda_j \nabla c_j(\tilde{x}) + \hat{\lambda}_j \nabla_{xx} c_j^k(\tilde{x},\mu) d \right\}.$$

Compare with KKT for

$$\begin{aligned}
\min_{d\in\mathbb{R}^n} \quad & \nabla_x r(\tilde{x},\mu)^\top d + \frac{1}{2} d^\top \left(\nabla_{xx} r(\tilde{x},\mu) - \hat{\lambda} \nabla_{xx} c_j^k(\tilde{x},\mu) \right) d \\
\text{subject to} \quad & \nabla c_j(\tilde{x})^\top d = 0, \quad j \in A_\epsilon.
\end{aligned}$$

A Practical Second-Order Exact Penalty Function

Consider the KKT conditions for

$$\begin{array}{ll} \min_{d\in\mathbb{R}^n} & \nabla_x r(\tilde{x},\mu)^\top d + \frac{1}{2} d^\top \nabla_{xx} r(\tilde{x},\mu) d \\ \text{subject to} & \nabla c_j(\tilde{x})^\top d + \frac{1}{2} d^\top \nabla^2 c_j(\tilde{x}) d = 0, \quad j \in A_\epsilon. \end{array}$$

We need

$$\nabla_x r(\tilde{x},\mu) + \nabla_{xx} r(\tilde{x},\mu) d = \sum_{j\in A_\epsilon} \lambda_j \left\{\nabla c_j(\tilde{x}) + \nabla_{xx} c_j(\tilde{x},\mu) d\right\}$$

Or using our good lambda estimate, $\hat{\lambda}$

$$\nabla_x r(\tilde{x},\mu) + \nabla_{xx} r(\tilde{x},\mu) d \approx \sum_{j\in A_\epsilon} \left\{\lambda_j \nabla c_j(\tilde{x}) + \hat{\lambda}_j \nabla_{xx} c_j^k(\tilde{x},\mu) d\right\}.$$

Compare with KKT for

$$\begin{array}{ll} \min_{d\in\mathbb{R}^n} & \nabla_x r(\tilde{x},\mu)^\top d + \frac{1}{2} d^\top \left(\nabla_{xx} r(\tilde{x},\mu) - \hat{\lambda} \nabla_{xx} c_j^k(\tilde{x},\mu)\right) d \\ \text{subject to} & \nabla c_j(\tilde{x})^\top d = 0, \quad j \in A_\epsilon. \end{array}$$

A Practical Second-Order Exact Penalty Function

Consider the KKT conditions for

$$\begin{array}{ll} \min\limits_{d \in \mathbb{R}^n} & \nabla_x r(\tilde{x},\mu)^\top d + \frac{1}{2} d^\top \nabla_{xx} r(\tilde{x},\mu) d \\ \text{subject to} & \nabla c_j(\tilde{x})^\top d + \frac{1}{2} d^\top \nabla^2 c_j(\tilde{x}) d = 0, \quad j \in A_\epsilon. \end{array}$$

We need

$$\nabla_x r(\tilde{x},\mu) + \nabla_{xx} r(\tilde{x},\mu) d = \sum_{j \in A_\epsilon} \lambda_j \left\{ \nabla c_j(\tilde{x}) + \nabla_{xx} c_j(\tilde{x},\mu) d \right\}$$

Or using our good lambda estimate, $\hat{\lambda}$

$$\nabla_x r(\tilde{x},\mu) + \nabla_{xx} r(\tilde{x},\mu) d \approx \sum_{j \in A_\epsilon} \left\{ \lambda_j \nabla c_j(\tilde{x}) + \hat{\lambda}_j \nabla_{xx} c_j^k(\tilde{x},\mu) d \right\}.$$

Compare with KKT for

$$\begin{array}{ll} \min\limits_{d \in \mathbb{R}^n} & \nabla_x r(\tilde{x},\mu)^\top d + \dfrac{1}{2} d^\top \left(\nabla_{xx} r(\tilde{x},\mu) - \hat{\lambda} \nabla_{xx} c_j^k(\tilde{x},\mu) \right) d \\ \text{subject to} & \nabla c_j(\tilde{x})^\top d = 0, \quad j \in A_\epsilon. \end{array}$$

A Practical Second-Order Exact Penalty Function

Consider the KKT conditions for

$$\begin{array}{ll} \min\limits_{d\in\mathbb{R}^n} & \nabla_x r(\tilde{x},\mu)^\top d + \frac{1}{2} d^\top \nabla_{xx} r(\tilde{x},\mu) d \\ \text{subject to} & \nabla c_j(\tilde{x})^\top d + \frac{1}{2} d^\top \nabla^2 c_j(\tilde{x}) d = 0, \quad j \in A_\epsilon. \end{array}$$

We need

$$\nabla_x r(\tilde{x},\mu) + \nabla_{xx} r(\tilde{x},\mu) d = \sum_{j\in A_\epsilon} \lambda_j \left\{ \nabla c_j(\tilde{x}) + \nabla_{xx} c_j(\tilde{x},\mu) d \right\}$$

Or using our good lambda estimate, $\hat{\lambda}$

$$\nabla_x r(\tilde{x},\mu) + \nabla_{xx} r(\tilde{x},\mu) d \approx \sum_{j\in A_\epsilon} \left\{ \lambda_j \nabla c_j(\tilde{x}) + \hat{\lambda}_j \nabla_{xx} c_j^k(\tilde{x},\mu) d \right\} .$$

Compare with KKT for

$$\begin{array}{ll} \min\limits_{d\in\mathbb{R}^n} & \nabla_x r(\tilde{x},\mu)^\top d + \frac{1}{2} d^\top \left(\nabla_{xx} r(\tilde{x},\mu) - \hat{\lambda} \nabla_{xx} c_j^k(\tilde{x},\mu) \right) d \\ \text{subject to} & \nabla c_j(\tilde{x})^\top d = 0, \quad j \in A_\epsilon. \end{array}$$

A Practical Second-Order Exact Penalty Function

Consider the KKT conditions for

$$\begin{array}{ll} \min\limits_{d\in\mathbb{R}^n} & \nabla_x r(\tilde{x},\mu)^\top d + \frac{1}{2} d^\top \nabla_{xx} r(\tilde{x},\mu) d \\ \text{subject to} & \nabla c_j(\tilde{x})^\top d + \frac{1}{2} d^\top \nabla^2 c_j(\tilde{x}) d = 0, \quad j \in A_\epsilon. \end{array}$$

We need

$$\nabla_x r(\tilde{x},\mu) + \nabla_{xx} r(\tilde{x},\mu) d = \sum_{j\in A_\epsilon} \lambda_j \left\{\nabla c_j(\tilde{x}) + \nabla_{xx} c_j(\tilde{x},\mu) d\right\}$$

Or using our good lambda estimate, $\hat{\lambda}$

$$\nabla_x r(\tilde{x},\mu) + \nabla_{xx} r(\tilde{x},\mu) d \approx \sum_{j\in A_\epsilon} \left\{\lambda_j \nabla c_j(\tilde{x}) + \hat{\lambda}_j \nabla_{xx} c_j^k(\tilde{x},\mu) d\right\}.$$

Compare with KKT for

$$\begin{array}{ll} \min\limits_{d\in\mathbb{R}^n} & \nabla_x r(\tilde{x},\mu)^\top d + \dfrac{1}{2} d^\top \left(\nabla_{xx} r(\tilde{x},\mu) - \hat{\lambda} \nabla_{xx} c_j^k(\tilde{x},\mu)\right) d \\ \text{subject to} & \nabla c_j(\tilde{x})^\top d = 0, \quad j \in A_\epsilon. \end{array}$$

An l_1 augmented Lagrangian?

Remark

The usual (l_2)-Augmented Lagrangian is a combination of the (inexact) quadratic penalty with the Lagrangian which is motivated as a means to prevent the necessity of requiring the penalty parameter for the quadratic penalty to tend to infinity

Thus it seems eccentric to augment an exact penalty function

But one can obtain a better estimate of the multipliers

I hope to convince you that there are significant advantages for our approach, although, for very large problems, per iteration, there can be clear computational disadvantages because of the required projections.

But this is true for all active set methods; including the simplex method

An l_1 augmented Lagrangian?

Remark

The usual (l_2)-Augmented Lagrangian is a combination of the (inexact) quadratic penalty with the Lagrangian which is motivated as a means to prevent the necessity of requiring the penalty parameter for the quadratic penalty to tend to infinity

Thus it seems eccentric to augment an exact penalty function

But one can obtain a better estimate of the multipliers

I hope to convince you that there are significant advantages for our approach, although, for very large problems, per iteration, there can be clear computational disadvantages because of the required projections.

But this is true for all active set methods; including the simplex method

An l_1 augmented Lagrangian?

Remark

The usual (l_2)-Augmented Lagrangian is a combination of the (inexact) quadratic penalty with the Lagrangian which is motivated as a means to prevent the necessity of requiring the penalty parameter for the quadratic penalty to tend to infinity

Thus it seems eccentric to augment an exact penalty function

But one can obtain a better estimate of the multipliers

I hope to convince you that there are significant advantages for our approach, although, for very large problems, per iteration, there can be clear computational disadvantages because of the required projections.

But this is true for all active set methods; including the simplex method

An l_1 augmented Lagrangian?

Remark

The usual (l_2)-Augmented Lagrangian is a combination of the (inexact) quadratic penalty with the Lagrangian which is motivated as a means to prevent the necessity of requiring the penalty parameter for the quadratic penalty to tend to infinity

Thus it seems eccentric to augment an exact penalty function

But one can obtain a better estimate of the multipliers

I hope to convince you that there are significant advantages for our approach, although, for very large problems, per iteration, there can be clear computational disadvantages because of the required projections.

But this is true for all active set methods; including the simplex method

An l_1 augmented Lagrangian?

Remark

The usual (l_2)-Augmented Lagrangian is a combination of the (inexact) quadratic penalty with the Lagrangian which is motivated as a means to prevent the necessity of requiring the penalty parameter for the quadratic penalty to tend to infinity

Thus it seems eccentric to augment an exact penalty function

But one can obtain a better estimate of the multipliers

I hope to convince you that there are significant advantages for our approach, although, for very large problems, per iteration, there can be clear computational disadvantages because of the required projections.

But this is true for all active set methods; including the simplex method

The l_1 augmented Lagrangian

It is just a small step to define an l_1-analogue to the augmented Lagrangian

$\min_{x \in \mathbb{R}^n} \quad \mathcal{L}_1(x, \lambda, \mu) =$

$f(x) - \sum_i \lambda_i c_i(x) + \mu \sum_{j \in \{1,\dots,m\}} \max\{c_j(x), 0\}$

and again only one of μ and the λs needs to be updated

Furthermore, the "gradient" of the augmented Lagrange compared with the Lagrangian gives the λ update.

Where "gradient" is interpreted via first-order conditions in that it is those that we are trying to drive to zero.

The l_1 augmented Lagrangian

It is just a small step to define an l_1-analogue to the augmented Lagrangian

$$\min_{x \in \mathbb{R}^n} \ \mathcal{L}_1(x, \lambda, \mu) =$$

$$f(x) - \sum_i \lambda_i c_i(x) + \mu \sum_{j \in \{1,\ldots,m\}} \max\{c_j(x), 0\}$$

and again only one of μ and the λs needs to be updated

Furthermore, the "gradient" of the augmented Lagrange compared with the Lagrangian gives the λ update.

Where "gradient" is interpreted via first-order conditions in that it is those that we are trying to drive to zero.

The l_1 augmented Lagrangian

It is just a small step to define an l_1-analogue to the augmented Lagrangian

$$\min_{x\in\mathbb{R}^n} \quad \mathcal{L}_1(x,\lambda,\mu) =$$

$$f(x) - \sum_i \lambda_i c_i(x) + \mu \sum_{j\in\{1,\dots,m\}} \max\{c_j(x), 0\}$$

and again only one of μ and the λs needs to be updated

Furthermore, the "gradient" of the augmented Lagrange compared with the Lagrangian gives the λ update.

Where "gradient" is interpreted via first-order conditions in that it is those that we are trying to drive to zero.

The l_1 augmented Lagrangian

It is just a small step to define an l_1-analogue to the augmented Lagrangian

$$\min_{x\in\mathbb{R}^n} \ \mathcal{L}_1(x,\lambda,\mu) =$$

$$f(x) - \sum_i \lambda_i c_i(x) + \mu \sum_{j\in\{1,\dots,m\}} \max\{c_j(x), 0\}$$

and again only one of μ and the λs needs to be updated

Furthermore, the "gradient" of the augmented Lagrange compared with the Lagrangian gives the λ update.

Where "gradient" is interpreted via first-order conditions in that it is those that we are trying to drive to zero.

The l_1 augmented Lagrangian

It is just a small step to define an l_1-analogue to the augmented Lagrangian

$\min_{x\in\mathbb{R}^n} \; \mathcal{L}_1(x,\lambda,\mu) =$

$f(x) - \sum_i \lambda_i c_i(x) + \mu \sum_{j\in\{1,\ldots,m\}} \max\{c_j(x), 0\}$

and again only one of μ and the λs needs to be updated

Furthermore, the "gradient" of the augmented Lagrange compared with the Lagrangian gives the λ update.

Where "gradient" is interpreted via first-order conditions in that it is those that we are trying to drive to zero.

The l_1 augmented Lagrangian λ updates

$$\nabla \mathcal{L}_1(x^{l+1}, \lambda, \mu) =$$

$$\nabla f(x^{l+1}) - \sum_{j=1}^{m} \lambda_j \nabla c_j(x^{l+1}) + \mu \sum_{j \in V_\epsilon} \nabla c_j(x^{l+1}) - \sum_{j \in A_\epsilon} \bar{\lambda}_j \nabla c_j(x^{l+1}) = 0.$$

where $\bar{\lambda}$ are (least square) estimates.

Comparing coefficients of $\nabla c_j(x)$ suggests the update formula,

$$\begin{aligned} \lambda_i^+ &= \lambda_j + \bar{\lambda}_j, && \text{if } j \in A_\epsilon \\ &= \lambda_j - \mu, && \text{if } j \in V_\epsilon \\ &= \lambda_j, && \text{if } j \notin A_\epsilon \cup V_\epsilon \end{aligned}$$

Optimality conditions come from the gradient of the Lagrangian with the updated multipliers, which is indeed differentiable

The l_1 augmented Lagrangian λ updates

$$\nabla \mathcal{L}_1(x^{l+1}, \lambda, \mu) =$$

$$\nabla f(x^{l+1}) - \sum_{j=1}^{m} \lambda_j \nabla c_j(x^{l+1}) + \mu \sum_{j \in V_\epsilon} \nabla c_j(x^{l+1}) - \sum_{j \in A_\epsilon} \bar{\lambda}_j \nabla c_j(x^{l+1}) = 0.$$

where $\bar{\lambda}$ are (least square) estimates.

Comparing coefficients of $\nabla c_j(x)$ suggests the update formula,

$$\begin{aligned} \lambda_i^+ &= \lambda_j + \bar{\lambda}_j, && \text{if } j \in A_\epsilon \\ &= \lambda_j - \mu, && \text{if } j \in V_\epsilon \\ &= \lambda_j, && \text{if } j \notin A_\epsilon \cup V_\epsilon \end{aligned}$$

Optimality conditions come from the gradient of the Lagrangian with the updated multipliers, which is indeed differentiable

The l_1 augmented Lagrangian λ updates

$$\nabla \mathcal{L}_1(x^{l+1}, \lambda, \mu) =$$

$$\nabla f(x^{l+1}) - \sum_{j=1}^{m} \lambda_j \nabla c_j(x^{l+1}) + \mu \sum_{j \in V_\epsilon} \nabla c_j(x^{l+1}) - \sum_{j \in A_\epsilon} \bar{\lambda}_j \nabla c_j(x^{l+1}) = 0.$$

where $\bar{\lambda}$ are (least square) estimates.

Comparing coefficients of $\nabla c_j(x)$ suggests the update formula,

$$\begin{aligned} \lambda_i^+ &= \lambda_j + \bar{\lambda}_j, && \text{if } j \in A_\epsilon \\ &= \lambda_j - \mu, && \text{if } j \in V_\epsilon \\ &= \lambda_j, && \text{if } j \notin A_\epsilon \cup V_\epsilon \end{aligned}$$

Optimality conditions come from the gradient of the Lagrangian with the updated multipliers, which is indeed differentiable

The l_1 augmented Lagrangian λ updates

$$\nabla \mathcal{L}_1(x^{l+1}, \lambda, \mu) =$$

$$\nabla f(x^{l+1}) - \sum_{j=1}^{m} \lambda_j \nabla c_j(x^{l+1}) + \mu \sum_{j \in V_\epsilon} \nabla c_j(x^{l+1}) - \sum_{j \in A_\epsilon} \bar{\lambda}_j \nabla c_j(x^{l+1}) = 0.$$

where $\bar{\lambda}$ are (least square) estimates.

Comparing coefficients of $\nabla c_j(x)$ suggests the update formula,

$$\begin{aligned} \lambda_i^+ &= \lambda_j + \bar{\lambda}_j, && \text{if } j \in A_\epsilon \\ &= \lambda_j - \mu, && \text{if } j \in V_\epsilon \\ &= \lambda_j, && \text{if } j \notin A_\epsilon \cup V_\epsilon \end{aligned}$$

Optimality conditions come from the gradient of the Lagrangian with the updated multipliers, which is indeed differentiable

The l_1 augmented Lagrangian λ updates

$$\nabla \mathcal{L}_1(x^{l+1}, \lambda, \mu) =$$

$$\nabla f(x^{l+1}) - \sum_{j=1}^{m} \lambda_j \nabla c_j(x^{l+1}) + \mu \sum_{j \in V_\epsilon} \nabla c_j(x^{l+1}) - \sum_{j \in A_\epsilon} \bar{\lambda}_j \nabla c_j(x^{l+1}) = 0.$$

where $\bar{\lambda}$ are (least square) estimates.

Comparing coefficients of $\nabla c_j(x)$ suggests the update formula,

$$\begin{aligned} \lambda_i^+ &= \lambda_j + \bar{\lambda}_j, && \text{if } j \in A_\epsilon \\ &= \lambda_j - \mu, && \text{if } j \in V_\epsilon \\ &= \lambda_j, && \text{if } j \notin A_\epsilon \cup V_\epsilon \end{aligned}$$

Optimality conditions come from the gradient of the Lagrangian with the updated multipliers, which is indeed differentiable

The l_1 augmented Lagrangian subproblem

Now, the subproblem we wish to solve for a second-order method is

$$\min_{d\in\mathbb{R}^n} \quad \nabla_x D(x^l,\lambda^l,\mu^l)^\top d + \frac{1}{2} d^\top \nabla_{xx} D(x^l,\lambda^l,\mu^l) d$$

$$\text{subject to} \quad \nabla c_j(x^l)^\top d + \tfrac{1}{2} d^\top \nabla^2 c_j(x^l) d = 0, \quad j \in A_\epsilon.$$

where $\quad D(x,\lambda,\mu) = f(x) - \sum_{j=1}^{m} \lambda_j c_j(x) + \mu \sum_{j\in V_\epsilon} c_j(x)$

is locally differentiable

and we already know how to do that appropriately.

The l_1 augmented Lagrangian subproblem

Now, the subproblem we wish to solve for a second-order method is

$$\min_{d\in\mathbb{R}^n} \quad \nabla_x D(x^l,\lambda^l,\mu^l)^\top d + \frac{1}{2} d^\top \nabla_{xx} D(x^l,\lambda^l,\mu^l) d$$

$$\text{subject to} \quad \nabla c_j(x^l)^\top d + \tfrac{1}{2} d^\top \nabla^2 c_j(x^l) d = 0, \quad j \in A_\epsilon.$$

where $D(x,\lambda,\mu) = f(x) - \sum_{j=1}^{m} \lambda_j c_j(x) + \mu \sum_{j\in V_\epsilon} c_j(x)$

is locally differentiable

and we already know how to do that appropriately.

The l_1 augmented Lagrangian subproblem

Now, the subproblem we wish to solve for a second-order method is

$$\min_{d\in\mathbb{R}^n} \quad \nabla_x D(x^l,\lambda^l,\mu^l)^\top d + \frac{1}{2} d^\top \nabla_{xx} D(x^l,\lambda^l,\mu^l) d$$

subject to $\quad \nabla c_j(x^l)^\top d + \frac{1}{2} d^\top \nabla^2 c_j(x^l) d = 0, \quad j \in A_\epsilon.$

where $\quad D(x,\lambda,\mu) = f(x) - \sum_{j=1}^{m} \lambda_j c_j(x) + \mu \sum_{j\in V_\epsilon} c_j(x)$

is locally differentiable

and we already know how to do that appropriately.

Model Problem Observation

As already stated, the model based upon a quadratic objective function and quadratic constraints should be the paradigm optimization problem

If solved via the quadratic penalty function or (l_2-)augmented Lagrangian it is more complex than quadratic (i.e. quartic)

If solved via an l_1-penalty function or an l_1-augmented Lagrangian it is piecewise quadratic

We claim that this is a significant advantage.

Model Problem Observation

As already stated, the model based upon a quadratic objective function and quadratic constraints should be the paradigm optimization problem

If solved via the quadratic penalty function or (l_2-)augmented Lagrangian it is more complex than quadratic (i.e. quartic)

If solved via an l_1-penalty function or an l_1-augmented Lagrangian it is piecewise quadratic

We claim that this is a significant advantage.

Model Problem Observation

As already stated, the model based upon a quadratic objective function and quadratic constraints should be the paradigm optimization problem

If solved via the quadratic penalty function or (l_2-)augmented Lagrangian it is more complex than quadratic (i.e. quartic)

If solved via an l_1-penalty function or an l_1-augmented Lagrangian it is piecewise quadratic

We claim that this is a significant advantage.

Model Problem Observation

As already stated, the model based upon a quadratic objective function and quadratic constraints should be the paradigm optimization problem

If solved via the quadratic penalty function or (l_2-)augmented Lagrangian it is more complex than quadratic (i.e. quartic)

If solved via an l_1-penalty function or an l_1-augmented Lagrangian it is piecewise quadratic

We claim that this is a significant advantage.

An l_1 augmented Lagrangian for equality constraints

For simplicity assume all general constraints are equality constraints and now

The l_1-exact penalty is $f(x) + \mu \sum_{j=1}^{m_e} |c_j(x)|$,

The l_1-augmented Lagrangian is $f^k(x) + \sum_{j=1}^{m_e} \lambda_j c_j(x) + \mu \sum_{j=1}^{m_e} |c_j(x)|$,

The multiplier update rule is

$$\lambda_j^{k+1} = \begin{cases} \lambda_j^k - \hat{\lambda}_j, & \text{if } j \in A_\epsilon \\ \lambda_j^k + \sigma_j^k \, \mu^{(k)}, & \text{if } j \in V_\epsilon \\ \lambda_j^k, & \text{if } j \notin A_\epsilon \cup V_\epsilon \end{cases},$$

where σ_j is the sign of $c_j(x)$.

The optimality conditions are that, for all $j \in A_\epsilon$

$$-\mu^{(k)} \le \hat{\lambda}_j^{(k)} \le 0, \qquad \text{if } \sigma_j^k = 1, \qquad \text{or}$$

$$0 \le \hat{\lambda}_j^{(k)} \le \mu^{(k)}, \qquad \text{if } \sigma_j^k = -1.$$

An l_1 augmented Lagrangian for equality constraints

For simplicity assume all general constraints are equality constraints and now

The l_1-exact penalty is $f(x) + \mu \sum_{j=1}^{m_e} |c_j(x)|$,

The l_1-augmented Lagrangian is $f^k(x) + \sum_{j=1}^{m_e} \lambda_j c_j(x) + \mu \sum_{j=1}^{m_e} |c_j(x)|$,

The multiplier update rule is

$$\lambda_j^{k+1} = \begin{cases} \lambda_j^k - \hat{\lambda}_j, & \text{if } j \in A_\epsilon \\ \lambda_j^k + \sigma_j^k\ \mu^{(k)}, & \text{if } j \in V_\epsilon \\ \lambda_j^k, & \text{if } j \notin A_\epsilon \cup V_\epsilon \end{cases},$$

where σ_j is the sign of $c_j(x)$.

The optimality conditions are that, for all $j \in A_\epsilon$

$$-\mu^{(k)} \leq \hat{\lambda}_j^{(k)} \leq 0, \qquad \text{if } \ \sigma_j^k = \ \ 1, \qquad \text{or}$$

$$0 \leq \hat{\lambda}_j^{(k)} \leq \mu^{(k)}, \qquad \text{if } \ \sigma_j^k = -1.$$

An l_1 augmented Lagrangian for equality constraints

For simplicity assume all general constraints are equality constraints and now

The l_1-exact penalty is $f(x) + \mu \sum_{j=1}^{m_e} |c_j(x)|$,

The l_1-augmented Lagrangian is $f^k(x) + \sum_{j=1}^{m_e} \lambda_j c_j(x) + \mu \sum_{j=1}^{m_e} |c_j(x)|$,

The multiplier update rule is

$$\lambda_j^{k+1} = \begin{cases} \lambda_j^k - \hat{\lambda}_j, & \text{if } j \in A_\epsilon \\ \lambda_j^k + \sigma_j^k \ \mu^{(k)}, & \text{if } j \in V_\epsilon \\ \lambda_j^k, & \text{if } j \notin A_\epsilon \cup V_\epsilon \end{cases},$$

where σ_j is the sign of $c_j(x)$.

The optimality conditions are that, for all $j \in A_\epsilon$

$$-\mu^{(k)} \leq \hat{\lambda}_j^{(k)} \leq 0, \qquad \text{if } \sigma_j^k = \ \ 1, \qquad \text{or}$$

$$0 \leq \hat{\lambda}_j^{(k)} \leq \mu^{(k)}, \qquad \text{if } \sigma_j^k = -1.$$

An l_1 augmented Lagrangian for equality constraints

For simplicity assume all general constraints are equality constraints and now

The l_1-exact penalty is $f(x) + \mu \sum_{j=1}^{m_e} |c_j(x)|$,

The l_1-augmented Lagrangian is $f^k(x) + \sum_{j=1}^{m_e} \lambda_j c_j(x) + \mu \sum_{j=1}^{m_e} |c_j(x)|$,

The multiplier update rule is

$$\lambda_j^{k+1} = \begin{cases} \lambda_j^k - \hat{\lambda}_j, & \text{if } j \in A_\epsilon \\ \lambda_j^k + \sigma_j^k \ \mu^{(k)}, & \text{if } j \in V_\epsilon \\ \lambda_j^k, & \text{if } j \notin A_\epsilon \cup V_\epsilon \end{cases},$$

where σ_j is the sign of $c_j(x)$.

The optimality conditions are that, for all $j \in A_\epsilon$

$$-\mu^{(k)} \leq \hat{\lambda}_j^{(k)} \leq 0, \qquad \text{if } \ \sigma_j^k = \ \ 1, \qquad \text{or}$$

$$0 \leq \hat{\lambda}_j^{(k)} \leq \mu^{(k)}, \qquad \text{if } \ \sigma_j^k = -1.$$

An l_1 augmented Lagrangian for equality constraints

For simplicity assume all general constraints are equality constraints and now

The l_1-exact penalty is $f(x) + \mu \sum_{j=1}^{m_e} |c_j(x)|$,

The l_1-augmented Lagrangian is $f^k(x) + \sum_{j=1}^{m_e} \lambda_j c_j(x) + \mu \sum_{j=1}^{m_e} |c_j(x)|$,

The multiplier update rule is

$$\lambda_j^{k+1} = \begin{cases} \lambda_j^k - \hat{\lambda}_j, & \text{if } j \in A_\epsilon \\ \lambda_j^k + \sigma_j^k \ \mu^{(k)}, & \text{if } j \in V_\epsilon \\ \lambda_j^k, & \text{if } j \notin A_\epsilon \cup V_\epsilon \end{cases},$$

where σ_j is the sign of $c_j(x)$.

The optimality conditions are that, for all $j \in A_\epsilon$

$$-\mu^{(k)} \leq \hat{\lambda}_j^{(k)} \leq 0, \qquad \text{if } \ \sigma_j^k = \ 1, \qquad \text{or}$$

$$0 \leq \hat{\lambda}_j^{(k)} \leq \mu^{(k)}, \qquad \text{if } \ \sigma_j^k = -1.$$

Adding simple bounds to the l_1 augmented Lagrangian

For bounds we need an additional (trivial) projection, $\mathcal{P}_{\mathcal{B}}$

In other words we deal with them directly. If we have just

$$x \geq 0$$

For any $x^{(k)}$, we have two possibilities for each component:

Dominated (*i*) $0 \leq x_i^{(k)} \leq \left(\nabla_x L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}) \right)_i$, or

Floating (*ii*) $\left(\nabla_x L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}) \right)_i < x_i^{(k)}$, where

$L_D(x, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}) = D^k(x, \lambda^{(k)}, \mu^{(k)}) - \sum_{j \in A_\epsilon} \hat{\lambda}_j^{(k)} c_j^k(x)$ and

$$D^k(x, \lambda^{(k)}, \mu^{(k)}) = f(x) + \sum_{j=1}^{m_e} \lambda_j^{(k)} c_j^k(x) + \mu^{(k)} \sum_{j \in V_\epsilon} \sigma_j^k c_j^k(x)$$

The algorithm forces , $\mathcal{P}_{\mathcal{B}}(x, \nabla_x L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}))$ to zero as k increases.

Adding simple bounds to the l_1 augmented Lagrangian

For bounds we need an additional (trivial) projection, $\mathcal{P}_{\mathcal{B}}$

In other words we deal with them directly. If we have just

$$x \geq 0$$

For any $x^{(k)}$, we have two possibilities for each component:

Dominated (i) $0 \leq x_i^{(k)} \leq \left(\nabla_x L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}) \right)_i$, or

Floating (ii) $\left(\nabla_x L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}) \right)_i < x_i^{(k)}$, where

$L_D(x, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}) = D^k(x, \lambda^{(k)}, \mu^{(k)}) - \sum_{j \in A_\epsilon} \hat{\lambda}_j^{(k)} c_j^k(x)$ and

$$D^k(x, \lambda^{(k)}, \mu^{(k)}) = f(x) + \sum_{j=1}^{m_e} \lambda_j^{(k)} c_j^k(x) + \mu^{(k)} \sum_{j \in V_\epsilon} \sigma_j^k c_j^k(x)$$

The algorithm forces , $\mathcal{P}_{\mathcal{B}}(x, \nabla_x L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}))$ to zero as k increases.

Adding simple bounds to the l_1 augmented Lagrangian

For bounds we need an additional (trivial) projection, $\mathcal{P}_{\mathcal{B}}$

In other words we deal with them directly. If we have just

$$x \geq 0$$

For any $x^{(k)}$, we have two possibilities for each component:

Dominated (i) $0 \leq x_i^{(k)} \leq \left(\nabla_x L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}) \right)_i$, or

Floating (ii) $\left(\nabla_x L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}) \right)_i < x_i^{(k)}$, where

$L_D(x, \lambda^{(k)} \mu^{(k)} \hat{\lambda}^{(k)}) = D^k(x, \lambda^{(k)}, \mu^{(k)}) - \sum_{j \in A_\epsilon} \hat{\lambda}_j^{(k)} c_j^k(x)$ and

$$D^k(x, \lambda^{(k)}, \mu^{(k)}) = f(x) + \sum_{j=1}^{m_e} \lambda_j^{(k)} c_j^k(x) + \mu^{(k)} \sum_{j \in V_\epsilon} \sigma_j^k c_j^k(x)$$

The algorithm forces , $\mathcal{P}_{\mathcal{B}}(x, \nabla_x L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}))$ to zero as k increases.

Adding simple bounds to the l_1 augmented Lagrangian

For bounds we need an additional (trivial) projection, $\mathcal{P}_{\mathcal{B}}$

In other words we deal with them directly. If we have just

$$x \geq 0$$

For any $x^{(k)}$, we have two possibilities for each component:

Dominated (i) $0 \leq x_i^{(k)} \leq \left(\nabla_x \, L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}) \right)_i$, or

Floating (ii) $\left(\nabla_x \, L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}) \right)_i < x_i^{(k)}$, where

$L_D(x, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}) = D^k(x, \lambda^{(k)}, \mu^{(k)}) - \sum_{j \in A_\epsilon} \hat{\lambda}_j^{(k)} c_j^k(x)$ and

$$D^k(x, \lambda^{(k)}, \mu^{(k)}) = f(x) + \sum_{j=1}^{m_e} \lambda_j^{(k)} c_j^k(x) + \mu^{(k)} \sum_{j \in V_\epsilon} \sigma_j^k c_j^k(x)$$

The algorithm forces , $\mathcal{P}_{\mathcal{B}}(x, \, \nabla_x \, L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}))$ to zero as k increases.

Adding simple bounds to the l_1 augmented Lagrangian

For bounds we need an additional (trivial) projection, $\mathcal{P}_{\mathcal{B}}$

In other words we deal with them directly. If we have just

$$x \geq 0$$

For any $x^{(k)}$, we have two possibilities for each component:

Dominated (i) $0 \leq x_i^{(k)} \leq \left(\nabla_x L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}) \right)_i$, or

Floating (ii) $\left(\nabla_x L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}) \right)_i < x_i^{(k)}$, where

$L_D(x, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}) = D^k(x, \lambda^{(k)}, \mu^{(k)}) - \sum_{j \in A_\epsilon} \hat{\lambda}_j^{(k)} c_j^k(x)$ and

$$D^k(x, \lambda^{(k)}, \mu^{(k)}) = f(x) + \sum_{j=1}^{m_e} \lambda_j^{(k)} c_j^k(x) + \mu^{(k)} \sum_{j \in V_\epsilon} \sigma_j^k c_j^k(x)$$

The algorithm forces , $\mathcal{P}_{\mathcal{B}}(x, \nabla_x L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)}))$ to zero as k increases.

The Algorithm

Step 1: Initializations: $\lambda^{(0)}$ and positive constants $\eta_0,\ \mu_0,\ \omega_0,\ \tau > 1,\ \gamma_1 < 1,\ \omega_* << 1,\ \eta_* < 1,$ $\alpha_\omega,\ \beta_\omega,\ \alpha_\eta$, and $\beta_\eta < 1$ are given.
Set $\mu^{(O)} = \mu_0,\ \alpha^{(O)} = \min(\mu^{(O)}, \gamma_1),\ \omega^{(O)} = \omega_0(\alpha^{(O)})^{\alpha_\omega},\ \eta^{(O)} = \eta_O(\alpha^{(O)})^{\alpha_\eta}$, and $k = 0$.
Step 2: Inner iteration. Find $x^{(k)} \in \mathbb{R}^n$ such that $x^{(k)}$ is $\omega^{(k)}$-critical.
In other words, $-\mu^{(k)} \le \hat{\lambda}_j^{(k)} \le 0$, if $\sigma_j^k = 1$, or $0 \le \hat{\lambda}_j^{(k)} \le \mu^{(k)}$, if $\sigma_j^k = -1$, for all $j \in A_\epsilon$ and

$$\|L_D^k(x^{(k)}, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)})\| \le \omega^{(k)},$$

where $\hat{\lambda}^{(k)}$ is a (least squares) estimate for

$$\nabla_x D^k(x^{(k)}, \lambda^{(k)}, \mu^{(k)}) = \sum_{j \in A_\epsilon} \hat{\lambda}_j^{(k)} \nabla_x c_j^k(x^{(k)}).$$

If $\|c(x^{(k)}\| \le \eta^{(k)}$ execute Step 3. Otherwise execute Step 4.
Step 3: Test for convergence and update Lagrange multiplier estimates.
If $\|L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)})\| \le \omega_*$ and $\|c(x^{(k)}\| \le \eta_*$ stop. Otherwise, set

$$\begin{aligned}
\lambda^{(k+1)} &= \bar{\lambda}(x^{(k)}, \lambda^{(k)}, \mu^{(k)}), \qquad \mu^{(k+1)} = \mu^{(k)},\\
\alpha^{(k+1)} &= \min(\mu^{(k+1)}, \gamma_I),\\
\omega^{(k+1)} &= \omega^{(k)}(\alpha^{(k+1)})^{\beta_\omega},\\
\eta^{(k+1)} &= \eta^{(k)}(\alpha^{(k+1)})^{\beta_\eta}.
\end{aligned}$$

increment k by one and go to Step 2.
Step 4: Increase the penalty parameter. Set

$$\begin{aligned}
\lambda^{(k+1)} &= \lambda^{(k+1)},\\
\mu^{(k+1)} &= \tau\ \mu^{(k)},\\
\alpha^{(k+1)} &= \min(\mu^{(k+1)}, \gamma_I),\\
\omega^{(k+1)} &= \omega^{(k)}(\alpha^{(k+1)})^{\alpha_\omega},\\
\eta^{(k+1)} &= \eta^{(k)}(\alpha^{(k+1)})^{\alpha_\eta}.
\end{aligned}$$

increment k by one and go to Step 2.

What can we prove? Essentially the same as LANCELOT

x^* is any limit of the sequence $\{x^{(k)}\}$ generated by our Algorithm:

Assume: The functions and constraints are twice continuously differentiable, the iterates $x^{(k)}$ lie in a closed, bounded domain and the Jacobian matrix of active constraints, $\hat{A}(x^*)$, has full column rank at any limit point, x^*, of the sequences $x^{(k)}$,
Let K be the indices of an infinite subsequence of $x^{(k)}$ with limit x^*.

Then

(i) There are positive constants a_1, a_2 and an integer k_0 such that

$$\|\bar{\lambda}(x^{(k)}, \lambda^{(k)}, \ \mu^{(k)}) - \lambda^*\| \leq a_1 \omega^{(k)} + a_2 \|x^{(k)} - x^*\|,$$

and

$$\|\lambda(x^{(k)}) - \lambda^*\| \leq a_2 \|x^{(k)} - x^*\|,$$

for all $k \geq k_0, (k \in K)$.

What can we prove? Essentially the same as LANCELOT

x^* is any limit of the sequence $\{x^{(k)}\}$ generated by our Algorithm:

Assume: The functions and constraints are twice continuously differentiable, the iterates $x^{(k)}$ lie in a closed, bounded domain and the Jacobian matrix of active constraints, $\hat{A}(x^*)$, has full column rank at any limit point, x^*, of the sequences $x^{(k)}$,
Let K be the indices of an infinite subsequence of $x^{(k)}$ with limit x^*.

Then

(i) There are positive constants a_1, a_2 and an integer k_0 such that

$$\|\bar{\lambda}(x^{(k)}, \lambda^{(k)}, \ \mu^{(k)}) - \lambda^*\| \leq a_1 \omega^{(k)} + a_2 \|x^{(k)} - x^*\|,$$

and

$$\|\lambda(x^{(k)}) - \lambda^*\| \leq a_2 \|x^{(k)} - x^*\|,$$

for all $k \geq k_0, (k \in K)$.

What can we prove? Essentially the same as LANCELOT

x^* is any limit of the sequence $\{x^{(k)}\}$ generated by our Algorithm:

Assume: The functions and constraints are twice continuously differentiable, the iterates $x^{(k)}$ lie in a closed, bounded domain and the Jacobian matrix of active constraints, $\hat{A}(x^*)$, has full column rank at any limit point, x^*, of the sequences $x^{(k)}$,
Let K be the indices of an infinite subsequence of $x^{(k)}$ with limit x^*.

Then

(i) There are positive constants a_1, a_2 and an integer k_0 such that

$$\|\bar{\lambda}(x^{(k)}, \lambda^{(k)},\ \mu^{(k)}) - \lambda^*\| \leq a_1\omega^{(k)} + a_2\|x^{(k)} - x^*\|,$$

and

$$\|\lambda(x^{(k)}) - \lambda^*\| \leq a_2\|x^{(k)} - x^*\|,$$

for all $k \geq k_0, (k \in K)$.

What can we prove? Essentially the same as LANCELOT

x^* is any limit of the sequence $\{x^{(k)}\}$ generated by our Algorithm:

Assume: The functions and constraints are twice continuously differentiable, the iterates $x^{(k)}$ lie in a closed, bounded domain and the Jacobian matrix of active constraints, $\hat{A}(x^*)$, has full column rank at any limit point, x^*, of the sequences $x^{(k)}$,
Let K be the indices of an infinite subsequence of $x^{(k)}$ with limit x^*.

Then

(i) There are positive constants a_1, a_2 and an integer k_0 such that

$$\|\bar{\lambda}(x^{(k)}, \lambda^{(k)}, \ \mu^{(k)}) - \lambda^*\| \leq a_1 \omega^{(k)} + a_2 \|x^{(k)} - x^*\|,$$

and

$$\|\lambda(x^{(k)}) - \lambda^*\| \leq a_2 \|x^{(k)} - x^*\|,$$

for all $k \geq k_0, (k \in K)$.

What can we prove? Essentially the same as LANCELOT

x^* is any limit of the sequence $\{x^{(k)}\}$ generated by our Algorithm:

Assume: The functions and constraints are twice continuously differentiable, the iterates $x^{(k)}$ lie in a closed, bounded domain and the Jacobian matrix of active constraints, $\hat{A}(x^*)$, has full column rank at any limit point, x^*, of the sequences $x^{(k)}$,
Let K be the indices of an infinite subsequence of $x^{(k)}$ with limit x^*.

Then

(i) There are positive constants a_1, a_2 and an integer k_0 such that

$$\|\bar{\lambda}(x^{(k)}, \lambda^{(k)},\ \mu^{(k)}) - \lambda^*\| \leq a_1\omega^{(k)} + a_2\|x^{(k)} - x^*\|,$$

and

$$\|\lambda(x^{(k)}) - \lambda^*\| \leq a_2\|x^{(k)} - x^*\|,$$

for all $k \geq k_0, (k \in K)$.

What can we prove?

Furthermore, suppose, in addition, that $c(x^*) = 0$. Then

(ii) The point x^* is a Karush-Kuhn-Tucker point for the problem

$$\begin{aligned} &\min_{x\in\mathcal{B}} && f(x) \\ &\text{subject to} && c_j(x) = 0, \quad j \in E = \{1, \cdots, m_e\}, \end{aligned} \qquad (1)$$

where $\mathcal{B} = \{x \in \mathbb{R}^n \mid x \geq 0\}$, λ^* is the corresponding vector of Lagrange multipliers, and the sequences $\{\bar{\lambda}(x^{(k)}, \lambda^{(k)},\ \mu^{(k)})\}$ and $\{\lambda(x^{(k)})\}$ converges to λ^* for $k \in K$,

(iii) The gradient $\nabla_x\ L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)})$ converges so that

$$g_L\left(x^k, \lambda^{(k+1)}\right) = \nabla f^k(x^k) + \left\{\sum_{j=1}^{m_e} \lambda_j - \sum_{j\in A_\epsilon} \hat{\lambda}_j^{(k)}\right\} \nabla_x c_j^k(x^k) = 0,$$

and $g_L(x^*, \lambda^*) = 0$, for $k \in K$, where g_L is the gradient with respect to x of the Lagrangian corresponding to (1).

What can we prove?

Furthermore, suppose, in addition, that $c(x^*) = 0$. Then

(ii) The point x^* is a Karush-Kuhn-Tucker point for the problem

$$\begin{array}{ll} \min\limits_{x \in \mathcal{B}} & f(x) \\ \text{subject to} & c_j(x) = 0, \quad j \in E = \{1, \cdots, m_e\}, \end{array} \tag{1}$$

where $\mathcal{B} = \{x \in \mathbb{R}^n \mid x \geq 0\}$, λ^* is the corresponding vector of Lagrange multipliers, and the sequences $\{\bar{\lambda}(x^{(k)}, \lambda^{(k)},\ \mu^{(k)})\}$ and $\{\lambda(x^{(k)})\}$ converges to λ^* for $k \in K$,

(iii) The gradient $\nabla_x\ L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)})$ converges so that

$$g_L\left(x^k, \lambda^{(k+1)}\right) = \nabla f^k(x^k) + \left\{ \sum_{j=1}^{m_e} \lambda_j - \sum_{j \in A_\epsilon} \hat{\lambda}_j^{(k)} \right\} \nabla_x c_j^k(x^k) = 0,$$

and $g_L(x^*, \lambda^*) = 0$, for $k \in K$, where g_L is the gradient with respect to x of the Lagrangian corresponding to (1).

What can we prove?

Furthermore, suppose, in addition, that $c(x^*) = 0$. Then

(ii) The point x^* is a Karush-Kuhn-Tucker point for the problem

$$\begin{array}{ll} \min\limits_{x \in \mathcal{B}} & f(x) \\ \text{subject to} & c_j(x) = 0, \quad j \in E = \{1, \cdots, m_e\}, \end{array} \tag{1}$$

where $\mathcal{B} = \{x \in \mathbb{R}^n \mid x \geq 0\}$, λ^* is the corresponding vector of Lagrange multipliers, and the sequences $\{\bar{\lambda}(x^{(k)}, \lambda^{(k)}, \ \mu^{(k)})\}$ and $\{\lambda(x^{(k)})\}$ converges to λ^* for $k \in K$,

(iii) The gradient $\nabla_x \ L_D^k(x^k, \lambda^{(k)}, \mu^{(k)}, \hat{\lambda}^{(k)})$ converges so that

$$g_L \left(x^k, \lambda^{(k+1)}\right) = \nabla f^k(x^k) + \left\{ \sum_{j=1}^{m_e} \lambda_j - \sum_{j \in A_\epsilon} \hat{\lambda}_j^{(k)} \right\} \nabla_x c_j^k(x^k) = 0,$$

and $g_L(x^*, \lambda^*) = 0$, for $k \in K$, where g_L is the gradient with respect to x of the Lagrangian corresponding to (1).

What can we prove?

Suppose further that the iterates $\{x^{(k)}\}$ converges to a single limit point x^*, then there is a constant $\mu > 0$ such that $\mu^{(k)} \leq \mu$ for all k.

Suppose strict complementary slackness holds.
Then for k sufficiently large, the set of floating variables are precisely those which lie away from their bounds at x^*.

The iterates x^k and the Lagrange multiplier estimates are at least R-linearly convergent

For all exterior penalty function methods, it is possible to converge to a non-feasible stationary point; usually easily rectified.

Moreover the inner iterations are superlinearly/quadratically convergent if one uses a second-order method.

What can we prove?

Suppose further that the iterates $\{x^{(k)}\}$ converges to a single limit point x^*, then there is a constant $\mu > 0$ such that $\mu^{(k)} \leq \mu$ for all k.

Suppose strict complementary slackness holds.
Then for k sufficiently large, the set of floating variables are precisely those which lie away from their bounds at x^*.

The iterates x^k and the Lagrange multiplier estimates are at least R-linearly convergent

For all exterior penalty function methods, it is possible to converge to a non-feasible stationary point; usually easily rectified.

Moreover the inner iterations are superlinearly/quadratically convergent if one uses a second-order method.

What can we prove?

Suppose further that the iterates $\{x^{(k)}\}$ converges to a single limit point x^*, then there is a constant $\mu > 0$ such that $\mu^{(k)} \leq \mu$ for all k.

Suppose strict complementary slackness holds.
Then for k sufficiently large, the set of floating variables are precisely those which lie away from their bounds at x^*.

The iterates x^k and the Lagrange multiplier estimates are at least R-linearly convergent

For all exterior penalty function methods, it is possible to converge to a non-feasible stationary point; usually easily rectified.

Moreover the inner iterations are superlinearly/quadratically convergent if one uses a second-order method.

What can we prove?

Suppose further that the iterates $\{x^{(k)}\}$ converges to a single limit point x^*, then there is a constant $\mu > 0$ such that $\mu^{(k)} \leq \mu$ for all k.

Suppose strict complementary slackness holds.
Then for k sufficiently large, the set of floating variables are precisely those which lie away from their bounds at x^*.

The iterates x^k and the Lagrange multiplier estimates are at least R-linearly convergent

For all exterior penalty function methods, it is possible to converge to a non-feasible stationary point; usually easily rectified.

Moreover the inner iterations are superlinearly/quadratically convergent if one uses a second-order method.

Numerical results with MADS implemented as Nomad:Thanks to Nadir Amaioua

MADS is a well-known derivative-free direct search optimization method.

The search step of MADS uses a model trust region approach.

It naturally uses quadratic models for the objective and the constraints when doing the search step.

So it naturally would like to solve quadratic problems with quadratic constraints.

Numerical results with MADS implemented as Nomad:Thanks to Nadir Amaioua

MADS is a well-known derivative-free direct search optimization method.

The search step of MADS uses a model trust region approach.

It naturally uses quadratic models for the objective and the constraints when doing the search step.

So it naturally would like to solve quadratic problems with quadratic constraints.

Numerical results with MADS implemented as Nomad:Thanks to Nadir Amaioua

MADS is a well-known derivative-free direct search optimization method.

The search step of MADS uses a model trust region approach.

It naturally uses quadratic models for the objective and the constraints when doing the search step.

So it naturally would like to solve quadratic problems with quadratic constraints.

Numerical results with MADS implemented as Nomad:Thanks to Nadir Amaioua

MADS is a well-known derivative-free direct search optimization method.

The search step of MADS uses a model trust region approach.

It naturally uses quadratic models for the objective and the constraints when doing the search step.

So it naturally would like to solve quadratic problems with quadratic constraints.

Numerical results with MADS implemented as Nomad

These results are on 20 constrained problems, with the largest having 20 variables and 15 constraints, and 4 simulation problems, with the largest having 10 variables and 10 constraints.

The comparisons are made with the QPQC subproblems solved by:

- MADS,
- the exact penalty function,
- the usual augmented Lagrangian,
- the l_1-augmented Lagrangian, and
- IPOPT,

The latter is a well-known smooth penalty function approach that uses the log barrier with a linesearch

IBM

Numerical results with MADS implemented as Nomad

These results are on 20 constrained problems, with the largest having 20 variables and 15 constraints, and 4 simulation problems, with the largest having 10 variables and 10 constraints.

The comparisons are made with the QPQC subproblems solved by:

- MADS,
- the exact penalty function,
- the usual augmented Lagrangian,
- the l_1-augmented Lagrangian, and
- IPOPT,

The latter is a well-known smooth penalty function approach that uses the log barrier with a linesearch

Numerical results with MADS implemented as Nomad

These results are on 20 constrained problems, with the largest having 20 variables and 15 constraints, and 4 simulation problems, with the largest having 10 variables and 10 constraints.

The comparisons are made with the QPQC subproblems solved by:

- MADS,
- the exact penalty function,
- the usual augmented Lagrangian,
- the l_1-augmented Lagrangian, and
- IPOPT,

The latter is a well-known smooth penalty function approach that uses the log barrier with a linesearch

Numerical results with MADS implemented as Nomad

These results are on 20 constrained problems, with the largest having 20 variables and 15 constraints, and 4 simulation problems, with the largest having 10 variables and 10 constraints.

The comparisons are made with the QPQC subproblems solved by:

- MADS,
- the exact penalty function,
- the usual augmented Lagrangian,
- the l_1-augmented Lagrangian, and
- IPOPT,

The latter is a well-known smooth penalty function approach that uses the log barrier with a linesearch

Numerical results with MADS implemented as Nomad

These results are on 20 constrained problems, with the largest having 20 variables and 15 constraints, and 4 simulation problems, with the largest having 10 variables and 10 constraints.

The comparisons are made with the QPQC subproblems solved by:

- MADS,
- the exact penalty function,
- the usual augmented Lagrangian,
- the l_1-augmented Lagrangian, and
- IPOPT,

The latter is a well-known smooth penalty function approach that uses the log barrier with a linesearch

Numerical results with MADS implemented as Nomad

These results are on 20 constrained problems, with the largest having 20 variables and 15 constraints, and 4 simulation problems, with the largest having 10 variables and 10 constraints.

The comparisons are made with the QPQC subproblems solved by:

- MADS,
- the exact penalty function,
- the usual augmented Lagrangian,
- the l_1-augmented Lagrangian, and
- IPOPT,

The latter is a well-known smooth penalty function approach that uses the log barrier with a linesearch

Numerical results with MADS implemented as Nomad

These results are on 20 constrained problems, with the largest having 20 variables and 15 constraints, and 4 simulation problems, with the largest having 10 variables and 10 constraints.

The comparisons are made with the QPQC subproblems solved by:

- MADS,
- the exact penalty function,
- the usual augmented Lagrangian,
- the l_1-augmented Lagrangian, and
- IPOPT,

The latter is a well-known smooth penalty function approach that uses the log barrier with a linesearch

Numerical results with MADS implemented as Nomad

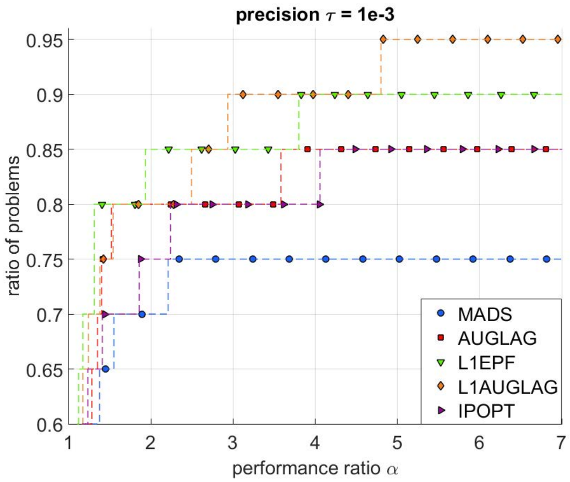
These results are on 20 constrained problems, with the largest having 20 variables and 15 constraints, and 4 simulation problems, with the largest having 10 variables and 10 constraints.

The comparisons are made with the QPQC subproblems solved by:

- MADS,
- the exact penalty function,
- the usual augmented Lagrangian,
- the l_1-augmented Lagrangian, and
- IPOPT,

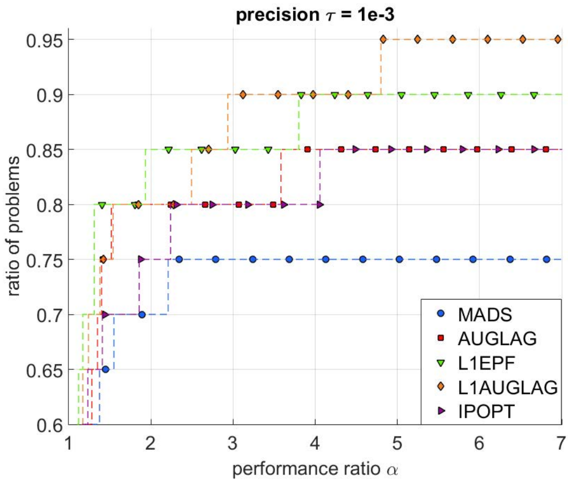
The latter is a well-known smooth penalty function approach that uses the log barrier with a linesearch

MADS on constrained problems in brief:



Fraction of problems you were less than α as bad as the best

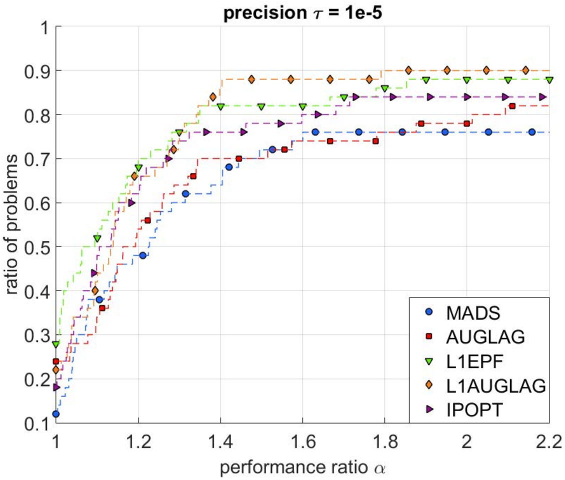
MADS on constrained problems in brief:



Fraction of problems you were less than α as bad as the best

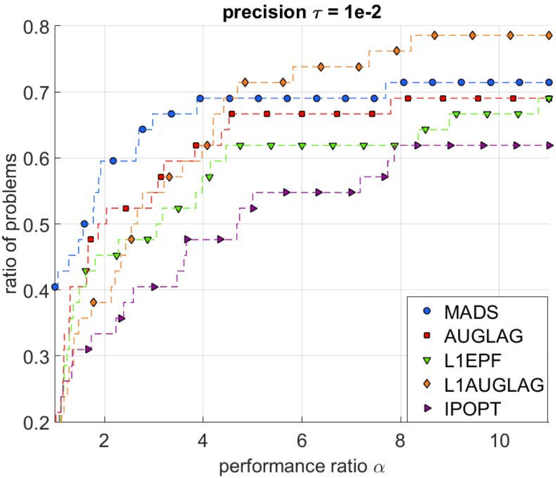
The Aircraft Range simulation problem: 50 instances

10 variables and 10 constraints   Simulation in days



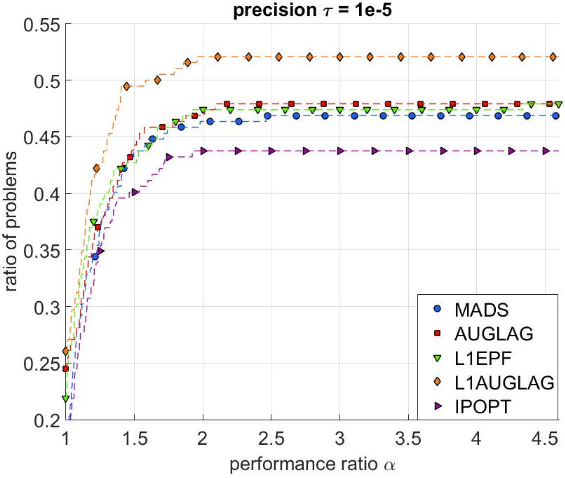
The Simplified Wing simulation problem: 50 instances

7 variables and 3 constraints   Simulation in days

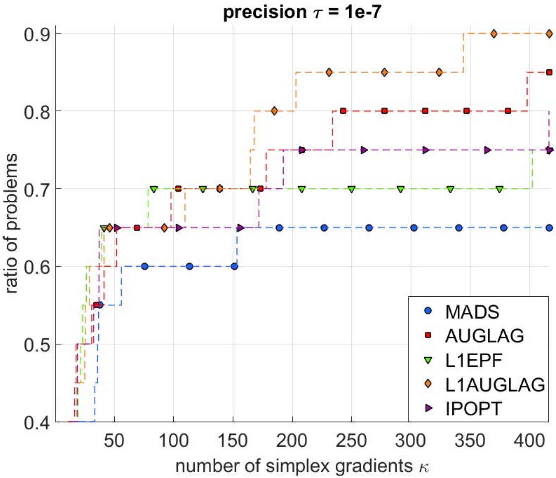


200 instances of the 4 simulation-based applications

Almost 3 weeks of simulation

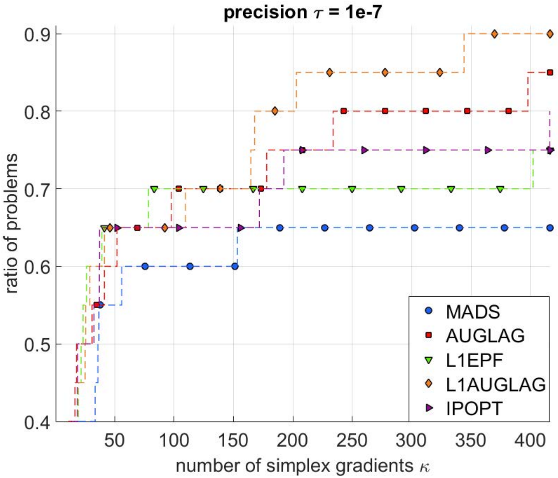


MADS on constrained problems in brief (data profiles)



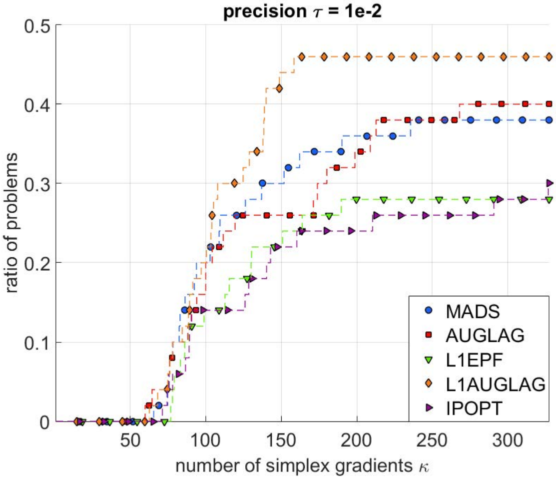
Fraction of problems that can be solved with α function evaluations

MADS on constrained problems in brief (data profiles)

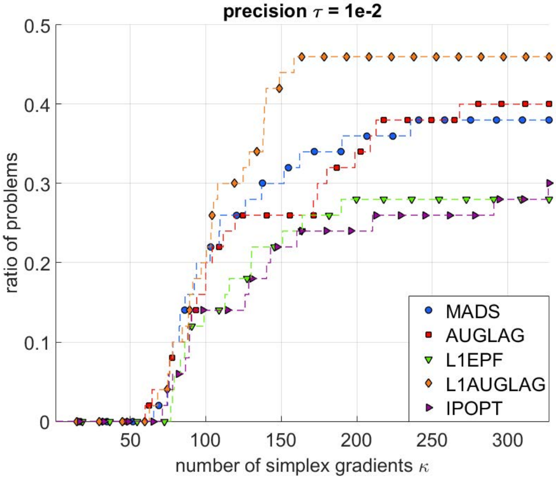


Fraction of problems that can be solved with α function evaluations

Lockwood simulation problem: 50 instances:data profile



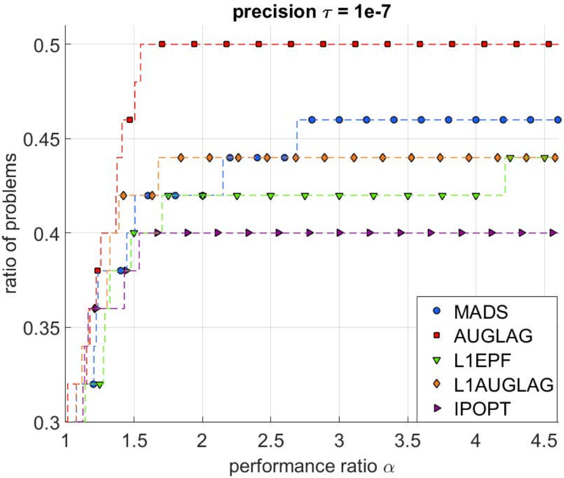
Lockwood simulation problem: 50 instances:data profile



Fraction of problems solved with α function evaluations

The Welded simulation problem:

Just in case you think we always win: 4 variables and 6 constraints



What next?

Is it better to handle inequalities directly?

Consider adaptive strategies for updating the penalty parameter.

How does it compare with other QPQC techniques like semidefinite programming, second-order cone programming, generalized eigen-decomposition, SCIP (Solving Constraint Integer Programs) and using the reformulation-linearization technique for discrete optimization problems?

What next?

Is it better to handle inequalities directly?

Can we do something about the projections for very large problems?
For example randomization techniques?

Consider adaptive strategies for updating the penalty parameter.

How does it compare with other QPQC techniques like semidefinite programming, second-order cone programming, generalized eigen-decomposition, SCIP (Solving Constraint Integer Programs) and using the reformulation-linearization technique for discrete optimization problems?

What next?

Is it better to handle inequalities directly?

Can we do something about the projections for very large problems?
For example randomization techniques?

Consider adaptive strategies for updating the penalty parameter.

How does it compare with other QPQC techniques like semidefinite programming, second-order cone programming, generalized eigen-decomposition, SCIP (Solving Constraint Integer Programs) and using the reformulation-linearization technique for discrete optimization problems?

What next?

Is it better to handle inequalities directly?

Can we do something about the projections for very large problems?
For example randomization techniques?

Consider adaptive strategies for updating the penalty parameter.

How does it compare with other QPQC techniques like semidefinite programming, second-order cone programming, generalized eigen-decomposition, SCIP (Solving Constraint Integer Programs) and using the reformulation-linearization technique for discrete optimization problems?