

7. Optimality Conditions of General Constrained Optimization

First to learn

Lagrange multiplier theorem

$$\begin{aligned} & \min_{\vec{x} \in \mathbb{R}^n} f(\vec{x}) \\ \text{s.t. } & \hat{C}_j(\vec{x}) = 0 \\ & ; j = 1, 2, \dots, m \end{aligned}$$

Equality-Constrained Optimization problem

$$\begin{aligned} & \min_{\substack{\vec{x} \in \mathbb{R}^n \\ \vec{\lambda} \in \mathbb{R}^m}} f(\vec{x}, \vec{\lambda}) \\ & \text{Lagrange multipliers} \end{aligned}$$

Unconstrained Optimization problem

$$\begin{aligned} & \min_{\vec{x} \in \mathbb{R}^n} f(\vec{x}) \\ \text{s.t. } & \hat{C}_j(\vec{x}) = 0 \\ & ; j = 1, 2, \dots, m \\ & C_j(\vec{x}) \leq 0 \\ & ; j = 1, 2, \dots, p \end{aligned}$$

General Constrained Optimization problem

$$\begin{aligned} & \min_{\vec{x} \in \mathbb{R}^n} f(\vec{x}) \\ \text{s.t. } & \hat{C}_j(\vec{x}) = 0 \\ & ; j = 1, 2, \dots, m \end{aligned}$$

Equality-Constrained Optimization problem

Second to learn

Karush-Kuhn-Tucker necessary conditions (KKT N.C)

First of all, learn what Lagrange multiplier theorem is.

Then, learn KKT conditions.

After them, we will discuss various numerical approaches to solve general Constrained Opt problems to get the optimum solutions that satisfy KKT conditions.

iterative methods.

7-1) Lagrange Multipliers Theorem

Convert Constrained opt prob w/ equality const into unconstrained opt prob

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

$$s.t. \hat{c}_i(\vec{x}) = 0; i=1 \sim m$$

Equality Constrained Opt prob

$$\begin{bmatrix} \hat{c}_{1,1}(\vec{x}) & \hat{c}_{1,2}(\vec{x}) & \cdots & \hat{c}_{1,n}(\vec{x}) \\ \hat{c}_{2,1}(\vec{x}) & \hat{c}_{2,2}(\vec{x}) & \cdots & \hat{c}_{2,n}(\vec{x}) \\ \vdots & & & \\ \hat{c}_{m,1}(\vec{x}) & \hat{c}_{m,2}(\vec{x}) & \cdots & \hat{c}_{m,n}(\vec{x}) \end{bmatrix}_{m \times n} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_{n \times 1} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{m \times 1}$$

\rightarrow $n > m$ eqn 있는 Coupled eqn. ($n > m$)

\rightarrow $(n-m)$ Ranked eqn. \rightarrow 총 변수 $m > n$

$$\begin{array}{l} \textcircled{1} \rightarrow \text{독립변수 } n-m > n. \\ \rightarrow \text{Let } \text{독립변수} = \vec{\lambda} \in \mathbb{R}^{n-m} \end{array}$$

$$\rightarrow f(\vec{x}(\vec{\lambda})) = \hat{f}(\vec{\lambda}) \rightarrow \min_{\vec{\lambda} \in \mathbb{R}^{n-m}} \hat{f}(\vec{\lambda})$$

Unconstrained Opt Prblm

\rightarrow 아래를 풀어서 \hat{f}' 를 찾으면 원래문제의 종속변수로 $\hat{c}_i(\vec{x}) = 0$ 관계식을 통해 자동으로 구해질 수 있음. (을로 explicit form 알고 implicit form)

③-1

$$\text{1st N.C of optimality for unconstrained optimization prblm} \rightarrow \nabla_{\vec{\lambda}} \hat{f} = \vec{0} \rightarrow \frac{\partial \hat{f}}{\partial \lambda_i} = \frac{\partial f}{\partial \lambda_i} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial \lambda_i} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial \lambda_i} + \dots + \frac{\partial f}{\partial x_n} \frac{\partial x_n}{\partial \lambda_i} = 0$$

\rightarrow $n-m-1$ terms are zero \rightarrow 독립변수는 그다음 독립변수로 미분하는 terms

\rightarrow $m+1$ terms are non-zero \rightarrow 종속변수를 독립변수로 미분하는 terms $m > n$.

+ 독립변수 λ_i 자체로 미분하는 term 1개

$$\text{only non-zero terms} \rightarrow \frac{\partial \hat{f}}{\partial \lambda_i} = \frac{\partial f}{\partial x_1} \left(\frac{\partial x_1}{\partial \lambda_i} \right) + \frac{\partial f}{\partial x_2} \left(\frac{\partial x_2}{\partial \lambda_i} \right) + \dots + \frac{\partial f}{\partial x_m} \left(\frac{\partial x_m}{\partial \lambda_i} \right) = 0 \quad \hat{c}_i(\vec{x}) = 0$$

$$\rightarrow \frac{\partial \hat{f}}{\partial \lambda_i} = \left[\frac{\partial f}{\partial \lambda_i} - \frac{\partial f}{\partial x_1} \left(\frac{\partial x_1}{\partial \lambda_i} \right) - \frac{\partial f}{\partial x_2} \left(\frac{\partial x_2}{\partial \lambda_i} \right) - \dots - \frac{\partial f}{\partial x_m} \left(\frac{\partial x_m}{\partial \lambda_i} \right) \right] = 0 \quad \hat{c}_i(\vec{x}) = 0$$

③-4

$$\rightarrow \frac{\partial \hat{f}}{\partial \lambda_i} = \left[\frac{\partial f}{\partial \lambda_i} - \lambda_1 \frac{\partial \hat{c}_1}{\partial \lambda_i} - \lambda_2 \frac{\partial \hat{c}_2}{\partial \lambda_i} - \dots - \lambda_m \frac{\partial \hat{c}_m}{\partial \lambda_i} \right] = 0 \quad 1 \geq 1$$

$$\text{where } \lambda_1 = \frac{\frac{\partial f}{\partial x_1}}{\frac{\partial \hat{c}_1}{\partial x_1}} \rightarrow \frac{\partial f}{\partial x_1} - \lambda_1 \frac{\partial \hat{c}_1}{\partial x_1} = 0 \quad m \geq 1$$

$$\lambda_2 = \frac{\frac{\partial f}{\partial x_2}}{\frac{\partial \hat{c}_2}{\partial x_2}} \rightarrow \frac{\partial f}{\partial x_2} - \lambda_2 \frac{\partial \hat{c}_2}{\partial x_2} = 0 \quad \vdots$$

$$\lambda_m = \frac{\frac{\partial f}{\partial x_m}}{\frac{\partial \hat{c}_m}{\partial x_m}} \rightarrow \frac{\partial f}{\partial x_m} - \lambda_m \frac{\partial \hat{c}_m}{\partial x_m} = 0$$

$$\text{Subject to } \begin{cases} \hat{c}_1(\vec{x}) = 0 \\ \hat{c}_2(\vec{x}) = 0 \\ \vdots \\ \hat{c}_m(\vec{x}) = 0 \end{cases} \quad m \geq 1$$

$\nabla_{\vec{\lambda}} \hat{f} = \vec{0}$ 의 성분 중 1번째 성분만 나타내보자.

증거

③-1 ~ ③-4

③-5

$\times n-m$ 개 독립변수 선제에 대해 반복해보기

$$\frac{\partial \hat{f}}{\partial \hat{x}_i} = 0 \quad (i=1, \dots, n-m) \text{ 을 다 만족하면,}$$

$$\begin{aligned} \frac{\partial \hat{f}}{\partial \hat{x}_1} &= \left[\frac{\partial f}{\partial x_1} - \lambda_1 \frac{\partial \hat{c}_1}{\partial x_1} - \lambda_2 \frac{\partial \hat{c}_2}{\partial x_1} - \dots - \lambda_m \frac{\partial \hat{c}_m}{\partial x_1} \right] = 0 \\ \frac{\partial \hat{f}}{\partial \hat{x}_2} &= \left[\frac{\partial f}{\partial x_2} - \lambda_1 \frac{\partial \hat{c}_1}{\partial x_2} - \lambda_2 \frac{\partial \hat{c}_2}{\partial x_2} - \dots - \lambda_m \frac{\partial \hat{c}_m}{\partial x_2} \right] = 0 \\ &\vdots \\ \frac{\partial \hat{f}}{\partial \hat{x}_{n-m}} &= \left[\frac{\partial f}{\partial x_{n-m}} - \lambda_1 \frac{\partial \hat{c}_1}{\partial x_{n-m}} - \lambda_2 \frac{\partial \hat{c}_2}{\partial x_{n-m}} - \dots - \lambda_m \frac{\partial \hat{c}_m}{\partial x_{n-m}} \right] = 0 \end{aligned}$$

m개 (Constraint 개수)

$\vec{x}_{\text{ своб}} = [x_1 \dots x_{n-m}]^T$
 $= [\hat{x}_1 \dots \hat{x}_{n-m}]^T$

$$\lambda_1 = \frac{\frac{\partial f}{\partial x_{n-m+1}}}{\frac{\partial \hat{c}_1}{\partial x_{n-m+1}}} \rightarrow \frac{\partial f}{\partial x_{n-m+1}} - \lambda_1 \frac{\partial \hat{c}_1}{\partial x_{n-m+1}} = 0$$

m개

$$\frac{\partial L}{\partial \hat{x}_{\text{ своб}}} = \vec{0}$$

n개

▶ 종속변수들.

즉, equality

Constraint $\vec{c} = \vec{0}$

으로부터 자유롭지 못한

설계변수 m개

$$\lambda_2 = \frac{\frac{\partial f}{\partial x_{n-m+2}}}{\frac{\partial \hat{c}_2}{\partial x_{n-m+2}}} \rightarrow \frac{\partial f}{\partial x_{n-m+2}} - \lambda_2 \frac{\partial \hat{c}_2}{\partial x_{n-m+2}} = 0$$

(※ 종속변수와 대응되는 $\hat{c}_j = 0$ 이 외의 $\hat{c}_j = 0$ 에 대한 편미분항 = 0)

$$\lambda_m = \frac{\frac{\partial f}{\partial x_n}}{\frac{\partial \hat{c}_m}{\partial x_n}} \rightarrow \frac{\partial f}{\partial x_n} - \lambda_m \frac{\partial \hat{c}_m}{\partial x_n} = 0$$

$$\vec{x}_{\text{ своб}} = [x_{n-m+1} \dots x_n]^T$$

④ Lagrangian --- Complete (증명완료)

④ 도입하여 표현

$n \times 1$ $m \times 1$

Subject to

$$\hat{c}_1(\vec{x}) = 0$$

$$\hat{c}_2(\vec{x}) = 0$$

$$\vdots$$

$$\hat{c}_m(\vec{x}) = 0$$

If we suppose $L(\vec{x}, \vec{\lambda}) = f(\vec{x}) - \vec{\lambda}^\top \vec{c}(\vec{x})$,

$$\rightarrow \nabla_{\vec{x}} L = \nabla_{\vec{x}} f(\vec{x}) - [\nabla_{\vec{x}}^\top \vec{c}(\vec{x})] \vec{\lambda} = \vec{0}$$

$$\rightarrow \nabla_{\vec{\lambda}} L = \vec{c}(\vec{x}) = \vec{0}$$

★ 1st-optimality condition of
 ★ Lagrangian function $L(\vec{x}, \vec{\lambda})$

$$\min f(\vec{x})$$

$\vec{x} \in \mathbb{R}^n$

$$s.t. \sum_i \hat{c}_i(\vec{x}) = 0$$

$i=1, \dots, m$

Equality-constrained
Optimization problem

Lagrange
Multiplier
Theorem

$$\min_{\vec{x} \in \mathbb{R}^n} L(\vec{x}, \vec{\lambda}) = f(\vec{x}) - \vec{\lambda}^\top \vec{c}(\vec{x})$$

Lagrangian function

$\vec{\lambda} \in \mathbb{R}^m$

Lagrange
multipliers

Unconstrained Optimization problem
(but with increase of number of design variables)

From $\nabla_{\vec{x}} L = \vec{0} \rightarrow \frac{\partial f}{\partial x_i} = \sum_{j=1}^m \lambda_j \frac{\partial c_j}{\partial x_i} \quad (i=1 \dots n)$ $\rightarrow \vec{x}$ optimal point by obj function의 기울기는
 Constraint 가로가 linear combination이어야 함.

※ 종속변수 x_i 의 독립변수 \hat{x}_j 에 대한 partial derivative는 $\frac{\partial x_i}{\partial \hat{x}_j}$ 유도
(using equality constraint $\hat{C}(\vec{x}) = 0$)

$$\begin{aligned}\hat{C}_1(x_1, \hat{x}_2, \hat{x}_3) = 0 \rightarrow x_1 = x_1(\hat{x}_2, \hat{x}_3) \rightarrow \frac{d\hat{C}_1}{d\hat{x}_2} = 0 \rightarrow d = \frac{\partial \hat{C}_1}{\partial x_1} dx_1 + \frac{\partial \hat{C}_1}{\partial \hat{x}_2} d\hat{x}_2 + \frac{\partial \hat{C}_1}{\partial \hat{x}_3} d\hat{x}_3 = 0 \\ \rightarrow \frac{d\hat{C}_1}{d\hat{x}_2} = \frac{\partial \hat{C}_1}{\partial x_1} \frac{dx_1}{d\hat{x}_2} + \frac{\partial \hat{C}_1}{\partial \hat{x}_2} + \frac{\partial \hat{C}_1}{\partial \hat{x}_3} \frac{d\hat{x}_3}{d\hat{x}_2} = 0 \\ \rightarrow \frac{d\hat{C}_1}{d\hat{x}_2} = \frac{\partial \hat{C}_1}{\partial x_1} \left(\frac{\partial x_1}{\partial \hat{x}_2} d\hat{x}_2 + \frac{\partial x_1}{\partial \hat{x}_3} d\hat{x}_3 \right) + \frac{\partial \hat{C}_1}{\partial \hat{x}_2} = 0\end{aligned}$$

Simple한 예시로 먼저 표현식에 대해 대충

값을 참고 이를 general한식으로 확장하자.

$$\rightarrow \frac{d\hat{C}_1}{d\hat{x}_2} = \frac{\partial \hat{C}_1}{\partial x_1} \frac{\partial x_1}{\partial \hat{x}_2} + \frac{\partial \hat{C}_1}{\partial \hat{x}_2} = 0$$

$$\rightarrow \frac{\partial x_1}{\partial \hat{x}_2} = -\frac{\frac{\partial \hat{C}_1}{\partial \hat{x}_2}}{\frac{\partial \hat{C}_1}{\partial x_1}}, \quad \frac{\partial x_1}{\partial \hat{x}_3} = -\frac{\frac{\partial \hat{C}_1}{\partial \hat{x}_3}}{\frac{\partial \hat{C}_1}{\partial x_1}}$$

In the same way

$$\begin{aligned}\hat{C}_2(\hat{x}_2, \hat{x}_3, x_4) = 0 \rightarrow x_4 = x_4(\hat{x}_2, \hat{x}_3) \rightarrow \frac{\partial x_4}{\partial \hat{x}_2} = -\frac{\frac{\partial \hat{C}_2}{\partial \hat{x}_2}}{\frac{\partial \hat{C}_2}{\partial x_4}}, \quad \frac{\partial x_4}{\partial \hat{x}_3} = -\frac{\frac{\partial \hat{C}_2}{\partial \hat{x}_3}}{\frac{\partial \hat{C}_2}{\partial x_4}}\end{aligned}$$

독립변수 : 2개 이상의 관계식에 포함되거나 어떤 관계식에도 포함되지 않은 선기변수.

종속변수 : 1개의 관계식에만 포함된 변수

* 관계식 : Equality constraint

$$\hat{C}(\vec{x}) = 0$$

→ i.e. 1개의 종속변수는 1개의 관계식에만 포함된다.

→ i.e. 종속변수와 관계식은 1대 1 대응이다.

→ i.e. 임의의 종속변수는 대응되는 관계식으로 표현될 수 있다.

→ 종속변수 x_i 를 독립변수 \hat{x}_j 로 편의를 위한 식은, 종속변수 x_i 가 1대1 대응되는 \hat{x}_j 로 표현 가능해지며 | 대응과 같은 | 표현 가능함.

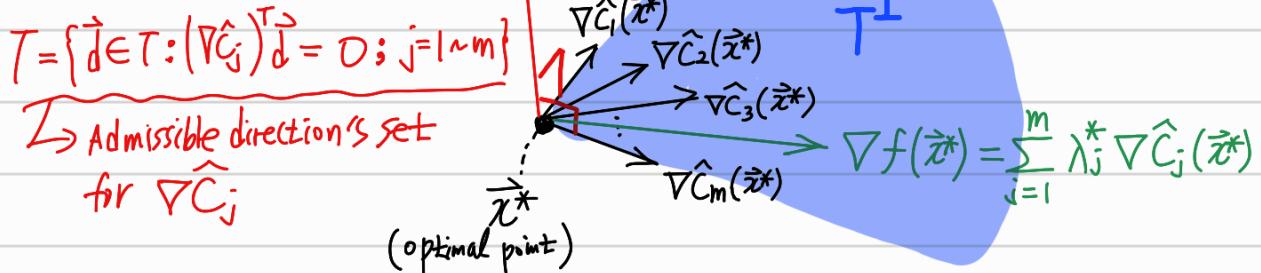
$$\rightarrow \frac{\partial x_i}{\partial \hat{x}_j} = -\frac{\frac{\partial \hat{C}_i}{\partial \hat{x}_j}}{\frac{\partial \hat{C}_i}{\partial x_i}} \quad \dots \text{general한 표현식.}$$

※ Lagrange multiplier theorem에 대한 고찰

① Equality constraints를 솔루션 간 관계식이라고 간주하여 그것을 기반으로 독립변수/종속변수를 나눈다. 독립변수에 대한 optimum condition을 만족할으면 종속변수 값들을 equality constraint 관계식을 기반으로, 기계적인 optimum 독립변수들로부터 자동으로 결정된다는 idea \rightarrow 매우 출발한다.

$$\text{② } \nabla f \Big|_{\vec{x}^*} = (\nabla \vec{\hat{C}}^T) \Big|_{\vec{x}^*} \lambda^* \stackrel{\substack{n \times m \\ \text{mx1}}}{\iff} \frac{\partial f}{\partial x_i} \Big|_{\vec{x}^*} = \sum_{j=1}^m \lambda_j^* \frac{\partial \hat{C}_j}{\partial x_i} \Big|_{\vec{x}^*} \dots \text{i.e. } \nabla f \text{는 optimum point } \vec{x}^* \text{에서 } \nabla \vec{C} \text{의 선형조합이어야 함!!}$$

Why?



① Equality constraint를 만족하는 이동방향 \vec{d} 의 set이 만들어내는 공간을 T 이다. 왜냐하면 그 방향으로 움직였을 때 모든 \hat{C}_j 에 대하여 여전히 $\hat{C}_j = 0$ 을 만족해야하기 때문이다. 즉 $d\hat{C}_j = \nabla \hat{C}_j \cdot \vec{d} = 0$ 을 만족하는 허용가능한 \vec{d} 의 set은 당연히 $\nabla \hat{C}_j$ 에 직교하는 공간 T 를 이룬다.

② 한편 local optimum \vec{x}^* 에서 ∇f 는 모든 허용 가능한 방향 $\vec{d} \in T$ 에 대하여 $d f = (\nabla f(\vec{x}^*))^T \cdot \vec{d} = 0$ 을 만족해야 한다.
따라서 $\nabla f(\vec{x}^*)$ 는 $\vec{d} \in T$ 와 직교해야 한다.

\rightarrow 따라서 $\nabla f(\vec{x}^*)$ 는 T^\perp 에 속하게 되고, 이를 곧 ∇f 는 $\nabla \hat{C}_j$ 의 linear combination 반드시 유효할 수 있어야 함을 의미한다.

※ 추가적으로 모든 j 에 대하여 $\nabla \hat{C}_j(\vec{x}^*)$ 각각은 모두 linearly independent해야 한다.

그리야만 $\nabla f(\vec{x}^*) = - \sum_{j=1}^m \lambda_j^* \nabla \hat{C}_j(\vec{x}^*)$ 이며 λ_j^* 가 uniquely determined이다.

$\nabla f(\vec{x}^*)$ 또한 unique 해가기 때문이다. $\nabla f(\vec{x}^*)$ 의 uniqueness는 \vec{x}^* 의 uniqueness를 의미하니 local optimum \vec{x}^* 에서 $\nabla \hat{C}_j$ 는 linearly independent해야 한다.

임의하는 두 번 만이지만 대충 의미론은 된다.

Appendix : Dual optimization problem (원래의 최적화 문제에 대한) (상대최적화 문제) about Primal optimization problem

$$\min_{\vec{x}} f(\vec{x})$$

$$\text{s.t. } \hat{c}_j(\vec{x}) = 0 ; j=1 \sim m$$

다시

=
말해

$$\min_{\vec{x}, \lambda} f(\vec{x}) - \sum_{j=1}^m \lambda_j \hat{c}_j(\vec{x})$$

Primal
Optimization
Problem

|| Satisfies if $H_{\vec{x}}(\vec{x}, \lambda)$ of Lagrangian function
w.r.t. \vec{x} above opt prob is locally P.D(convex).

$$\max_{\vec{\lambda}} \phi(\vec{\lambda}) \text{ where } \phi(\vec{\lambda}) = \min_{\vec{x}} (L(\vec{x}, \vec{\lambda}))$$

where $L(\vec{x}, \vec{\lambda}) = f(\vec{x}) - \sum_{j=1}^m \lambda_j \hat{c}_j(\vec{x})$

다시 말해,

$$\max_{\vec{\lambda}} \min_{\vec{x}} \left(f(\vec{x}) - \sum_{j=1}^m \lambda_j \hat{c}_j(\vec{x}) \right)$$

* No Constraint!!
Big Merit of Dual Problem!

Dual optimization problem

$\vec{\lambda}$ is solution mapping (or argmin mapping) of $\vec{\lambda}$
 $\vec{\lambda}$ 가 변하면 L 을 minimize하는 \vec{x} 도 따라 변함.
 $\vec{\lambda}$ 와 \vec{x} 가 서로에게 dependent하거나 이 관계를
 explicit하게 표기해놓은 것을 (심지어 주어진 $\vec{\lambda}$ 에 대해
 대응되는 \vec{x} 가 유일하지 않은 수도 있음). 즉 서로 다른 \vec{x} 가
 하나의 $\vec{\lambda}$ 에 대해 똑같은 L 값을 만들 수도 있음).
 그리고 $\vec{\lambda}$ 와 \vec{x} 의 관계를 명시적 관계식으로 표기하기.
 이것은 $\vec{\lambda}$ 가 주어지면 그 $\vec{\lambda}$ 에 맞는 \vec{x} 를 찾으려는 것과 겹친다.
 그렇게 최소화된 L 값이 주어진 $\vec{\lambda}$ 에 대한 function $\phi(\vec{\lambda})$ 임.

위에 대한 증명 과정(엄밀하지 않으니 참고)

Primal Optimization Problem

Optimum point of
primal opt prob

$$L(\vec{x}, \vec{\lambda}) = f(\vec{x}) - \sum_{j=1}^m \lambda_j \hat{c}_j(\vec{x})$$

\vec{x} -space



$$\min_{\vec{x}} f(\vec{x}) - \sum_{j=1}^m \lambda_j \hat{c}_j(\vec{x})$$

즉, $\vec{\lambda}_j$ 가 \vec{x}^* (constant vector)로 given.
이 constant 값은 primal opt prob의
 \vec{x}^* 에서 구해지는 그것과 같아야 함.

$$\min_{\vec{x}, \lambda} L(\vec{x}, \lambda) = f(\vec{x}) - \sum_{j=1}^m \lambda_j \hat{c}_j(\vec{x})$$

$$\vec{x}^* \neq \vec{x}^{1,*} \neq \vec{x}^{2,*} \neq \vec{x}^{3,*} \neq \dots \neq \vec{x}^{n,*}$$

이로 다른 \vec{x} 가 주어졌을 때 (보통은) 서로 다른 optimum

point이 되는데, optimum point가 서로 같지 않다면 그 중요성이
아니다. "이로 다른 \vec{x} 가 주어짐 = 서로 다른 문제에 등장". 이거
중요한 거임. (이로 다른 문제여도 optimum point는 같은 수 있음)

* 원래 문제 대비해서 우측의 문제 (즉, $\vec{\lambda}$ 가 각각 대로
주어진 문제)들은 주만 최적화하는 걸 주시하심.

즉, 얘네들은 Constraint $\hat{c}_j = 0$ 에 대한 만족을 전혀 고려안함.

\vec{x} -space

$$f(\vec{x}) - \sum_{j=1}^m \lambda'_j \hat{c}_j(\vec{x})$$

$$\min_{\vec{x}} f(\vec{x}) - \sum_{j=1}^m \lambda'_j \hat{c}_j(\vec{x})$$

constant vector where $\lambda' \neq \lambda^*$

\vec{x} -space

$$f(\vec{x}) - \sum_{j=1}^m \lambda''_j \hat{c}_j(\vec{x})$$

$$\min_{\vec{x}} f(\vec{x}) - \sum_{j=1}^m \lambda''_j \hat{c}_j(\vec{x})$$

constant vector where $\lambda'' \neq \lambda^*$

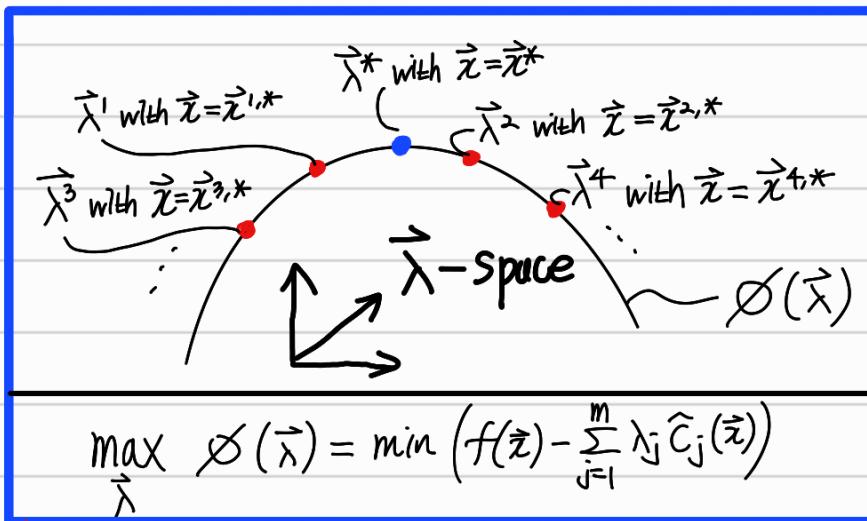
\vec{x} -space

$$f(\vec{x}) - \sum_{j=1}^m \lambda'''_j \hat{c}_j(\vec{x})$$

$$\min_{\vec{x}} f(\vec{x}) - \sum_{j=1}^m \lambda'''_j \hat{c}_j(\vec{x})$$

constant vector where $\lambda''' \neq \lambda^*$

⋮



Dual optimization problem

그림을 보면 \vec{x} 도 λ 의 위치마다 따라 변하는데 왜 \vec{x}, λ -space가 아닌 λ -space인가? 그리고 왜 λ 만 최적화하는 걸로 치는가? → 위 page에서 설명했듯, $\phi(\lambda)$ 함수의 경사에 따라 $\vec{x} \in \vec{x}$ 에

$$\max_{\lambda} \phi(\lambda) = \min_{\vec{x}} \left(f(\vec{x}) - \sum_{j=1}^m \lambda_j \hat{c}_j(\vec{x}) \right)$$

$$= \nabla_{\vec{x}}^2 L(\vec{x}, \lambda)$$

If $\tilde{H}_{\vec{x}}(\vec{x}, \lambda)$ of $L(\vec{x}, \lambda) = f(\vec{x}) - \sum_{j=1}^m \lambda_j \hat{c}_j(\vec{x})$ is P.D (Convex) at nearby \vec{x}^* ,

then $\tilde{H}_{\vec{x}}(\vec{x})$ of $\phi(\lambda)$ is N.D (Concave) at nearby λ^* .

$$= \nabla_{\lambda}^2 \phi(\lambda) \quad \therefore \lambda \neq \lambda^* \text{ 인 모든 } \lambda \text{에서, } \phi(\lambda) \leq \phi(\lambda^*) \text{ 이다.}$$

증명은 생략. 하지만

사실을 생각해보면 직관적 이해가 가능하다. λ -space에서 $\phi(\lambda)$ 위의 모든 point들은 primal opt prob의 Constraint $\hat{c}_j = 0$ 을 만족하지 않는 \vec{x} 및 \vec{x} 들이다.

그리고 얼마든지 \vec{x}^* , λ^* 아래의 L 보다 작은 값을 가질 수 있다 ($\hat{c}_j = 0$ 을 만족하는 \vec{x} !).

따라서 아래와 같은 대소관계가 성립한다.

$$\phi(\lambda) \leq \phi(\lambda^*) \leq L(\vec{x}^*, \lambda^*) \leq L(\vec{x}, \lambda)$$

Optimum point of primal opt prob
 \vec{x}^*

$L(\vec{x}, \lambda)$
 \vec{x}, λ -Space

λ -Space

Dual problem theory
 증명 완료

* Dual theory가 optimization에서의 중요성이, primal prob 푸는 것 대비 어떤 이점이 있는지 등은 “Dual theory의 중요성/의미” 노트 파일 참고.

7-2) Karush - Kuhn - Tucker Conditions (a.k.a KKT)

(Inequality-Constrained opt prblm \rightarrow Equality-Constrained Opt prblm)

이미 “Inequality” constraint $C_j(\vec{x}) \leq 0 ; j=1 \sim p$ 가 포함된 optimization problem을 다룬다.

즉, <Inequality-constrained opt prblm>

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

S.t.

$$C_j(\vec{x}) \geq 0 ; j=1 \sim p$$

는 다른도록 한다. Equality constraint가 추가되면 단지 Lagrange

multiplier theorem을 적용하여 문제를 다루면 되기 때문에, 여기서는

2つ inequality constraints 만 있는 문제를 다루도록 한다.

※ 참고로 대부분의 경우에 $C_j(\vec{x}) \leq 0$ 은 regular form으로 삼으나 여기서는 $C_j(\vec{x}) \geq 0$ 을

표기하였다. ($C_j(\vec{x}) \geq 0$ 은 regular form으로 표기하는 문항도 있음)

※ 주의! $C_j \leq 0$ 로 표기한 경우 $L = f + \sum \lambda_j C_j$, $C_j \geq 0$ 로 표기한 경우 $L = f - \sum \lambda_j C_j$ 를 표기해야 “ $\lambda_j \geq 0$ ”이어야 하는 곳 배운 KKT Condition의 consistency가 깨지게 된다.

By introducing ‘slack variable(이완변수)’ $S_j (j=1 \sim p)$, we convert above inequality

- constrained opt prblm into equality-constrained opt prblm.

* S_j is not simply given!

S_j is introduced

as an additional variables that affect this opt prblm. Thus,

this prblm is dependent on S_j .

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

$$S.t. C_j(\vec{x}) \geq 0 ; j=1 \sim p$$

<Inequality Constrained opt prblm>

Introducing
 $\xleftarrow[S_j]{}$
the slack
variables

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

$$S.t. C_j(\vec{x}) - S_j^2 = 0$$

<Equality-constrained prblm.>

자, 이제 우측과 같이 변환된 “equality-constrained opt prblm”을 다룬다.

이 문제의 optimal point \vec{x}^* 를 찾았다고 가정해보자. 해당 문제는 equality-constrained 문제가 때문에 \vec{x}^* 에서 Lagrange multiplier theorem의 optimality N.C를 만족해야 한다.

$$\rightarrow L(\vec{x}, \vec{\lambda}, \vec{s}) = f(\vec{x}) - \sum_{j=1}^p \lambda_j (C_j(\vec{x}) - S_j^2) \quad \text{... Lagrangian Function}$$

$$\rightarrow \begin{cases} \textcircled{1} \quad \nabla_{\vec{x}} L(\vec{x}^*, \vec{\lambda}^*, \vec{s}^*) = \nabla_{\vec{x}} f(\vec{x}^*) - \sum_{j=1}^p \lambda_j^* (\nabla_{\vec{x}} C_j(\vec{x}^*) - S_j^{*2}) = 0 \\ \textcircled{2} \quad \nabla_{\vec{\lambda}} L(\vec{x}^*, \vec{\lambda}^*, \vec{s}^*) = 0 \rightarrow C_j(\vec{x}^*) - S_j^{*2} = 0 ; j=1 \sim p \end{cases}$$

여기까지는 Lagrange

multiplier theorem으로 부터

이미 알고 있는 것들.

(1st N.C
for optimality)

이제 추가적인 것들을 고려해야 한다(단순한 등호제약문제가 아니라 부등호제약을 등호제약 문제로 바꾸어 당연히 추가되는 부등호제약문제를 등호제약문제로 바꾼 문제의)

local optimizer \vec{x}^* 는 반드시 다음 2가지 Case 중 하나에 속한다. (\therefore 1 & 2 Cases에 대해서만 opt conditions

Case 1) \vec{x}^* 가 feasible design space의 boundary에 걸쳐있지 않고 조사하면 부등호제약문제의 opt

conditions를 모두 만족된다).

내부에 놓여있을 때 i.e. $\vec{C}(\vec{x}^*) > \vec{0}$... 모든 제약조건을 다 만족하지만 (Inactive)

이 경우, \vec{x}^* 가 feasible design space 내부에서 local minimizer이거나 창구에
 $\nabla_{\vec{x}} f(\vec{x}^*) = \vec{0}$ 이다. (\because 제약 조건이 inactive인 경우의 optimum은 unconstrained optimum과 같다)

$$\text{By N.C of } \vec{x}^* \rightarrow \nabla_{\vec{x}} L(\vec{x}^*, \lambda^*) = \nabla_{\vec{x}} f(\vec{x}^*) - \sum_{j=1}^p \lambda_j^* \nabla_{\vec{x}} C_j(\vec{x}^*) = \vec{0}$$

for 등호제약문제

↳ 증명 생략 (자연)

다시 말해

$$C_j(\vec{x}^* + \vec{p}) = C_j(\vec{x}^*) + \nabla_{\vec{x}} C_j(\vec{x}^*) \cdot \vec{p} \geq 0$$

$$\rightarrow \sum_{j=1}^p \lambda_j^* \nabla_{\vec{x}} C_j(\vec{x}^*) = \vec{0}$$

즉, \vec{p} 만큼 움직인 점에서도 여전히 제약 조건을 만족하는 \vec{p} 가 반드시 존재.

\therefore 반드시 $\nabla_{\vec{x}} C_j(\vec{x}^*) \neq \vec{0}$ (\because 움직인 for $j=1 \sim p$)

증여제의 C_j 값 $C_j(\vec{x}^* + \vec{p})$ 은 $C_j(\vec{x}^*)$

와는 달라야 하니까)

↳ 증명 생략 (자연)

$$\rightarrow \lambda_j^* = 0 (\because \nabla_{\vec{x}} C_j(\vec{x}^*) \neq \vec{0})$$

$\therefore C_j$ 가 inactive한 feasible space에서만 만족하는 $C_j(\vec{x}^* + \vec{p}) \geq 0$ 을 만족하는 방향으로의 변화가 가능 (제약에 대한 여유(slack)가 있으니까!)

$$\therefore \text{At } \vec{x}^*, \text{ If } \vec{C}(\vec{x}^*) > \vec{0}, \text{ then } \vec{\lambda}^* = \vec{0} \quad (3)-1$$

(Case 2) \vec{x}^* 가 feasible design space boundary에 놓여있을 때 $\rightarrow C_j(\vec{x}^*) = 0$

$$\text{By N.C of } \vec{x}^* \rightarrow \nabla_{\vec{x}} L(\vec{x}^*, \lambda^*) = \nabla_{\vec{x}} f(\vec{x}^*) - \sum_{j=1}^p \lambda_j^* \nabla_{\vec{x}} C_j(\vec{x}^*) = \vec{0}$$

for 등호제약문제

"일부" 제약 조건이 active -
($j=1 \sim \hat{p}$)
제약 조건에 여유(slack)
가 없음!

$$\rightarrow \nabla_{\vec{x}} f(\vec{x}^*) = \sum_{j=1}^{\hat{p}} \lambda_j^* \nabla_{\vec{x}} C_j(\vec{x}^*) + \sum_{j=\hat{p}+1}^p \lambda_j^* \nabla_{\vec{x}} C_j(\vec{x}^*)$$

Inactive 상태인 제약 조건들에 대한 term. 이 때 \vec{x}^* 에서 더 이상 움직일 방향이 없어야 한다 ($\because \vec{x}^*$ is local optimizer!).

반드시 존재함.

X. 참고로 모든 제약 조건이 active일 때

만족할 수는 없음. 그런 over-constrained 입

따라서 고려한 필요가 없는 Case임.

즉 어느 방향 \vec{p} 로 $\Delta f = \nabla f(\vec{x}^*) \cdot \vec{p} < 0$ & $\Delta C_j = \nabla C_j(\vec{x}^*) \cdot \vec{p} \geq 0$ 을

동시에 만족할 수는 없어야 한다. 이것을 초기 하위 (n -dimensional space) \rightarrow
기하적 관점에서 geometric perspective

해석해보자.

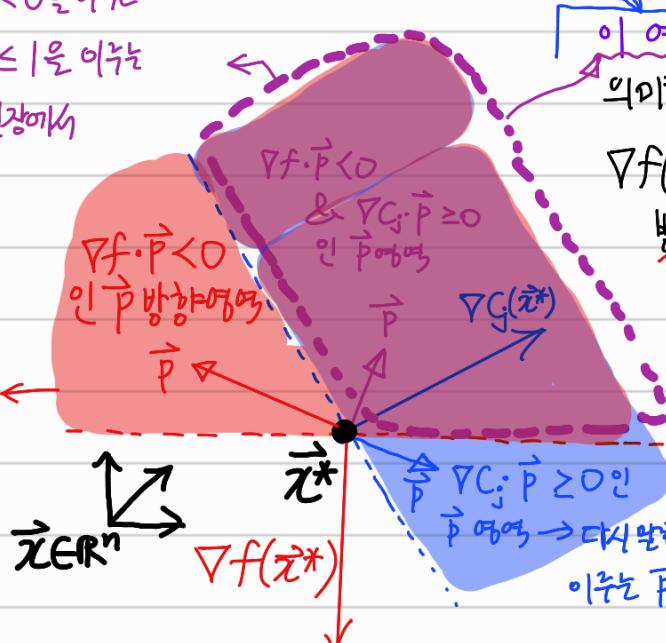
$j=1 \sim \hat{p}$

∇f 와의 각 θ 는 $-1 \leq \cos \theta < 0$ 을 이루고

∇C_j 와의 각 θ 는 $0 \leq \cos \theta \leq 1$ 을 이루는

\vec{p} 접합 (f 입장에서 움직일 수 있는 direction)

움직일 수 있는 direction)



이 예제가 존재하지 않아야 함을 의미한다. 이것을 기하적으로 해석하면

$\nabla f(\vec{x}^*)$ 와 $\nabla C_j(\vec{x}^*)$ 가 같은

방향을 나타내야 함을 의미한다.

(\because 그때야 각각에 빨간색 Red zone이

Blue zone Completely separated됨)

즉 $\nabla f(\vec{x}^*) = \lambda_j^* \nabla C_j(\vec{x}^*)$ 일 때,

반드시 $\lambda_j^* \geq 0$ 이어야 함을 의미한다

이제는 \vec{p} 접합 (C 입장에서 움직일 수 있는 direction)

다시 말해 ∇f 와의 각 θ 가 $0 < \cos \theta \leq 1$ 을

이루는 \vec{p} 접합 (f 입장에서 움직일 수 있는 direction)

있는 direction)

$j=1 \sim \hat{p}$ 가지 모든 경우에 대해 마찬가지이다. 따라서

$\nabla f(\vec{x}^*) = \sum_{j=1}^{\hat{p}} \lambda_j^* \nabla C_j(\vec{x}^*)$ & $\lambda_j^* \geq 0 ; j=1 \sim \hat{p}$ 이어야 한다.

• At \vec{x}^* , If $C_j(\vec{x}^*) = 0$, then $\lambda_j^* \geq 0 ; j=1 \sim \hat{p}$ (3)-2 & (4)

나머지 $C_j (j=\hat{p}+1 \sim p)$ 들은 랭크 1
 λ_j^* 가 0이기 때문에, Lagrangian
function L 의 term | 사라지므로
 ∇f 와 ∇C_j 로 가야하는 압을

\therefore From Case 1) & Case 2), at the optimum point \vec{x}^* of inequality constrained optimization problem, it should be

$$\rightarrow \nabla_{\vec{x}} L(\vec{x}^*, \vec{\lambda}^*) = \vec{0}, C_j(\vec{x}^*) \geq 0, \vec{\lambda}^* \geq 0, \lambda_j^* C_j(\vec{x}^*) = 0; j=1 \sim p$$

(1) (2) (4) (3)

$\min f(\vec{x})$
 $\vec{x} \in \mathbb{R}^n$

s.t. $C_j(\vec{x}) \geq 0 (j=1 \sim p)$

$(\because \nabla_{\vec{x}} L(\vec{x}^*, \vec{\lambda}^*) = C_j(\vec{x}^*) - s_j^2 = 0 \text{ and } s_j^2 \geq 0 \text{ so } C_j(\vec{x}^*) \geq 0)$

"Inequality" Constrained opt prob of 4 가지 optimality conditions.

문제를 풀 때는 보통 ①, ② 먼저 계산해서 후보 $\vec{x}^*, \vec{\lambda}^*$ 를 추려내고 후보들에 ③, ④를 판별식처럼 적용해서 결과적으로 ①~④를 모두 만족하는 $\vec{x}^*, \vec{\lambda}^*$ 를 찾는다.

위 conditions or "equality" constrained opt prob의 optimality conditions를 합치면 KKT conditions가 된다.

→ KKT Conditions.

$$\min f(\vec{x})$$

$\vec{x} \in \mathbb{R}^n$

$$\text{s.t. } \hat{C}_j(\vec{x}) = 0 \quad (j=1 \sim m)$$

$$C_j(\vec{x}) \geq 0 \quad (j=1 \sim p)$$

\therefore Sufficient Condition for Strong local optimum

$H(\vec{x}^*)$ of L must be Positive definite.

$$L(\vec{x}, \vec{\lambda}) = f(\vec{x}) + \sum_{j=1}^m \hat{\lambda}_j \hat{C}_j(\vec{x}) + \sum_{j=1}^p \lambda_j g_j(\vec{x})$$

$$(g_j(\vec{x}) = C_j(\vec{x}) - s_j^2 = 0)$$

$$\nabla_{\vec{x}} L = \nabla f(\vec{x}) + \sum_{j=1}^m \hat{\lambda}_j \nabla \hat{C}_j(\vec{x}) + \sum_{j=1}^p \lambda_j \nabla g_j(\vec{x}) = \vec{0}$$

$$\nabla_{\vec{\lambda}} L = \hat{C}_j(\vec{x}) = 0 \quad (2)$$

$$\nabla_{\vec{\lambda}} L = \nabla g_j(\vec{x}) = 0 \Leftrightarrow C_j(\vec{x}) \geq 0 \quad (3)$$

$$\nabla_{\vec{s}} L = \begin{bmatrix} -2s_1 \lambda_1 \\ \vdots \\ -2s_p \lambda_p \end{bmatrix} = 0 \Leftrightarrow s_j \lambda_j = 0 \rightarrow s_j^2 \lambda_j = 0 \rightarrow C_j \lambda_j = 0 \quad j=1 \sim p$$

$$\lambda_j \geq 0 \quad (4)$$

\therefore KKT Conditions of inequality constraint C_j 와 관련된 Lagrange multiplier λ_j 에 대한 충언.

Active인 inequality constraints $C_j (j=1 \sim \hat{p})$ 에 대해서는 $C_j = 0$ 이거나 $\lambda_j \geq 0$ 이어야 함.

그리야지만 | Lagrangian function L 에서 $-\sum_{j=1}^{\hat{p}} \lambda_j C_j$ term이 증가하는 것에 대한 penalty를 줄 수 있음. 반면 이미 inactive인 나머지

$= C_j \text{ 가 감소}$

inequality eqn을 $C_j (j=\hat{p}+1 \sim p)$ 에 대한 Lagrange multipliers λ_j 들은 굳이 penalty 안 쓰도 됨. 주면 오히려 \vec{x} 업데이트 시 f 의 decrease를 방해할 수 있음. 때문에 $\lambda_j = 0$ 으로 두어야 함.

7-3) Exercises & Available Search direction

ex1) Equality-Constrained Opt Prblm

$$\min f = x_1 + x_2$$

$$\text{s.t. } \hat{c} = 2 - x_1^2 - x_2^2$$

$$\rightarrow L = x_1 + x_2 - \lambda_1 (2 - x_1^2 - x_2^2)$$

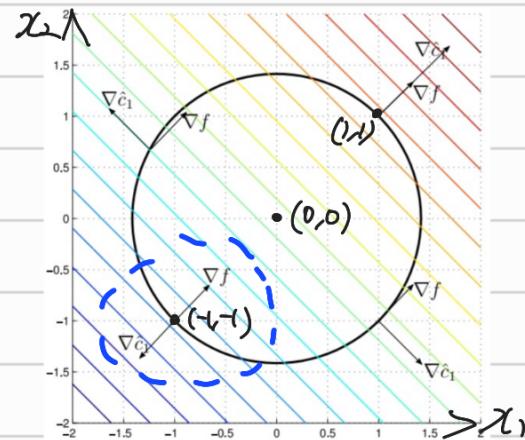


Figure 1: Example 1

$$\begin{aligned} \nabla_x L &= \begin{bmatrix} 1 + 2\lambda_1 x_1 \\ 1 + 2\lambda_1 x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{--- (a)} \\ \nabla_\lambda L &= [2 - x_1^2 - x_2^2] = [0] \rightarrow x_1^2 + x_2^2 = 2. \quad \text{--- (b)} \\ -(1+2\lambda_1 x_1) + (1+2\lambda_1 x_2) &= 0 \end{aligned}$$

3 unknowns
3 eqns.

$$\rightarrow 2\lambda_1(x_2 - x_1) = 0. \rightarrow \text{If } \lambda_1 = 0, \text{ (a) is not satisfied} \\ \therefore \lambda_1 \neq 0, x_1 = x_2$$

$$\text{from (b)} \rightarrow 2x_1^2 = 2 \rightarrow x_1 = 1, x_2 = 1, \lambda_1 = -\frac{1}{2}$$

$$f(1,1) = 2 \rightarrow \text{Max point}$$

$$\boxed{x_1 = -1, x_2 = -1, \lambda_1 = \frac{1}{2}} \\ f(-1,-1) = -2 \rightarrow \text{Min point}$$

ex2) What is a good search direction in Inequality-Constrained Opt Prblm

$$\min f(x_1, x_2) = x_1 + x_2$$

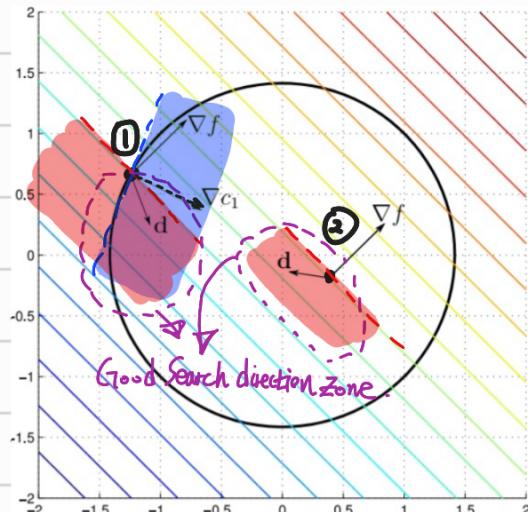
$$\text{s.t. } c_1(x_1, x_2) = 2 - x_1^2 - x_2^2 \geq 0.$$

When \vec{x} is not a local optimizer,
what is a good search direction?

① If \vec{x} is strictly inside of the constrained region,

i.e. $c_1(\vec{x})$ is inactive, which means $c_1(\vec{x}) > 0$,

If \vec{d} is sufficiently small,



$\Delta C_1 \approx C_1(\vec{x}) + \nabla^T C_1(\vec{x}) \cdot \vec{d} \geq 0$ will be still satisfied

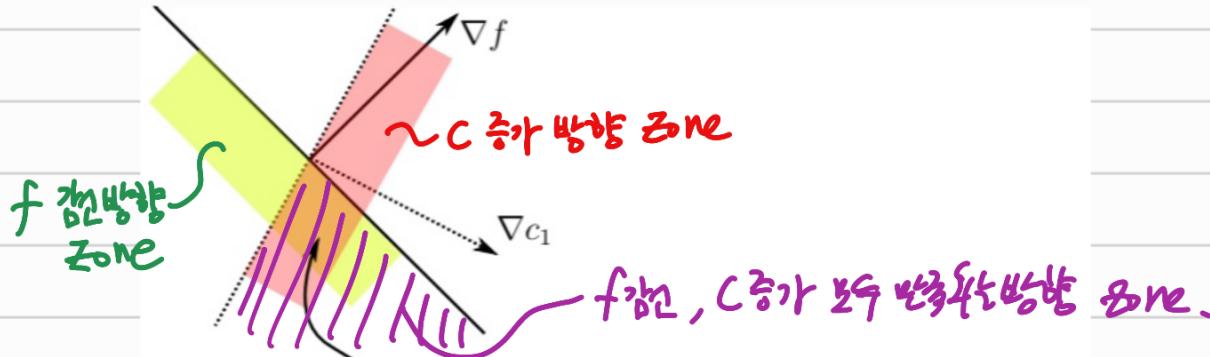
So it's okay to simply pick up \vec{d} that makes f decreased as a search direction.

② If \vec{x} is on the boundary of the constrained region, i.e. $C_1(\vec{x})$ is active, which means $C_1(\vec{x}) = 0$, search direction \vec{d} should satisfy following:

1) $\nabla^T f(\vec{x}) \cdot \vec{d} \leq 0$... for further decrease of f

2) $\nabla^T C_1(\vec{x}) \cdot \vec{d} \geq 0$... for staying in constrained region defined as $C_1(\vec{x}) \geq 0$

$\therefore \vec{d}$ must be in the region like following:



Any d in this region is a good search direction

이제 우리의 관심사는 "그럼 general constrained opt prblm에서 이러한(즉 'f의 감소'와 'constraint 만족'을 동시에 만족하는) search direction p를 어떻게 매 iteration에서 찾을 것인가?"로 간다. 다음 장에서 이를 알아보자.

※ 참고 Constraint qualification (한마디로 제약조건 잘 써놓았는 알임.)

만약 $\hat{C}_1(\vec{x}) = (x_1^2 + x_2^2 - 2)^2 = 0$ 와 같이 제약조건이 솔직히 아니고 생겼어보자.

$(x_1, x_2) = (0, 0)$ 을 중심으로 지나가 $\sqrt{2}$ 인 원 위의 모든 점 중에 최적점을 찾아야 한다.

근데 여기서 나온 $\nabla \hat{C}_1$ 는

$$\nabla \hat{C}_1(\vec{x}) = \begin{bmatrix} 4(x_1^2 + x_2^2 - 2)x_1 \\ 4(x_1^2 + x_2^2 - 2)x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow \text{근데 } \nabla \hat{C}_1(\vec{x}) = 0 \text{이 되는게 } \vec{x} = 0 \text{ 뿐인 경우는.}$$

근데 KKT 조건으로부터 $\nabla f(\vec{x}) = \lambda_1 \nabla \hat{C}_1(\vec{x})$ 인 점을 찾아야 하는데 $\nabla \hat{C}_1(\vec{x}) = 0$ 이니 풀어야 하는 모든 점에서 만족된다 $\nabla f(\vec{x}) = 0$ 인 거야. 그렇기 때문에 minimum point를 찾을 수가 없는 거임. 이런 경우는 constraint 자체가 잘못된 거임. $\hat{C}_1(\vec{x}) = x_1^2 + x_2^2 - 2 = 0$ 처럼 그냥 풀어 써줄 수는 없지만 $\nabla \hat{C}_1(\vec{x}) \neq 0$ 이 되어서 문제를 풀 수 있음.

→ 교훈: 본질적으로 같은 의미를 뜻하는 constraint도 표현 방식이 따라 문제를 풀 수 있게 만들 수도, 풀 수 없게 만들 수도 있으니 주의하여 constraint 속을 세우야됨.

8. Numerical (Iterative) Algorithms for General Nonlinear Constrained Optimization.

→ 이 강의에서 Linear optimization 은 생략한다. 즉, Standard Linear Programming Problem を 사용!!
→ Simplex 법으로 풀.

Nonlinear Constrained optimization 을 다루는 4가지 approaches 를 배우겠다.

① Quadratic Programming (QP)

• Linear Programming (LP) 또는

★ ② Penalty and Augmented Lagrangian Methods.

Sequential Linear Programming (SLP) 기법도

★ ③ Sequential Quadratic Programming. (SQP)

있는데 이 강의에서는 다뤄지 않는다. GIST

④ Interior Point Methods. (IPM)

때 강의노트, Anota 책에 자세히 나오니

→ 강의에서 생략.

SQP만 학습하
야 하므로 큰 도움 있

나중에 필요하다면 참고해라. 여기서 끝고
로 넘기기엔 시간이 부족. (나중에 Sensitivity
analysis, AI, Global Opt., MDO 등에
되는데 여유가 없을 때 -_-::)

8-1) Quadratic Programming

- Quadratic Programming Problem : Objective function이 quadratic function, Constraint function이 linear form인 constrained optimization 을 의미. 즉 수식적으로는 다음과을 의미.

$$\min_{\vec{x}} q(\vec{x}) = \frac{1}{2} \vec{x}^T (\vec{Q} \vec{x} + \vec{C}^T \vec{x}) \rightarrow \text{Objective function이 } \vec{x} \text{에 대한 2차 함수}$$

(상수항은 생략 가능. 최적화 후 단순 더하면 되니까)

$$\begin{aligned} \text{s.t. } \vec{a}_i^T \vec{x} &= \vec{b}_i ; i \in E \\ \vec{a}_i^T \vec{x} &\geq \vec{b}_i ; i \in I \end{aligned} \quad \left. \begin{array}{l} \rightarrow \text{Equality \& Inequality Constraints 가} \\ \text{모두 } \vec{x} \text{에 대한 linear function의 형태} \end{array} \right.$$

<General Quadratic Programming Problem>

→ 여기서 general의 의미: Inequality constraint까지 포함한 QP 문제를 의미. 엄밀한 정의는 아니고

강의에서 우리가 배우는 equality constraint만 포함한 QP 문제에서 구분하고자

Why?

내가 일부러 이렇게 표현하였음.

- Quadratic Programming : Quadratic Programming Problem을 푸는 일련의 과정을 의미. 특징 '기법'이 아니고
그냥 QP 문제를 푸는 과정 자체를 의미하는 추상적 단어에 가까울. 여기서 'Programming'
이라 함은, 이런 '수학 문제를 풀어가는 정식화된 수학적 절차'를 의미하며, 컴퓨터
프로그래밍과는 무관하다. → 이걸 제시한 1940년대에 불여진 용어.

- General QP 문제(Inequality Constraints 포함) 를 풀기 위한 Quadratic Programming에는 여러 method 존재

- Interior Point Methods. (IPM)
- Active Set Methods. (ASM)

- Augmented Lagrangian Methods.(ALM)
- Conjugate Gradient Methods.(CGM)
- Gradient Projection Methods.(GPM)
- Extensions of the Simplex Algorithm.(LP)
- ⋮

이와 GIST 강의에서도 general QP는 skip됨. 여기 강의상 속도가
있어서 행렬eqn 풀 때에 더 초점을
맞출었음.

→ 그러나 general QP 문제를 푸는 것은, 실제로 위와 같이 무수히 많은 접근법 중

일부를 선택하여 다뤄야 하기 때문에 (하나하나 method가 내용이 상당하기에) 시간적으로 어렵다.

따라서 이 section에서는 General QP 문제가 아닌, "QP problem with Equality Constraints"만을

대상으로 하여, 이 문제에서의 optimum point \vec{x}^* 를 찾기 위한 단 하나의 final equation을 derive하는데 초점을 맞춘다. 그 derive된 식을 푸는 방법은 여러 가지가 있지만 그들을 이용하여 final equation을 직접 풀어 \vec{x}^* 를 구해내는 과정에 대한 설명을 생략한다.

이
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
18

$$\tilde{A} \vec{x}^* = \vec{b}$$

$$\rightarrow \tilde{A}(\vec{x} + \vec{p}) = \vec{b}$$

$$\rightarrow \tilde{A} \vec{p} = -\tilde{A} \vec{x} + \vec{b}$$

$$\rightarrow \text{Let } \tilde{A} \vec{x} - \vec{b} = \vec{h}$$

$$\rightarrow [\tilde{A} \quad \tilde{0}] \begin{bmatrix} \vec{p} \\ \lambda^* \end{bmatrix} = -\vec{h}$$

$$\tilde{G} \vec{x}^* - \lambda^* \tilde{A}^T = -\vec{c}$$

$$\rightarrow \tilde{G}(\vec{x} + \vec{p}) - \lambda^* \tilde{A}^T = -\vec{c}$$

$$\rightarrow \tilde{G} \vec{p} - \lambda^* \tilde{A}^T = -(\tilde{G} \vec{x} + \vec{c})$$

$$\rightarrow \text{Let } \tilde{G} \vec{x} + \vec{c} = \vec{g}$$

$$\rightarrow \tilde{G} \vec{p} - \lambda^* \tilde{A}^T = -\vec{g}$$

$$\rightarrow \begin{bmatrix} \tilde{G} & -\tilde{A}^T \end{bmatrix} \begin{bmatrix} \vec{p} \\ \lambda^* \end{bmatrix} = -\vec{g}$$

4

$$\text{KKT matrix} \quad \begin{bmatrix} \tilde{G} & -\tilde{A}^T \\ \tilde{A} & \tilde{0} \end{bmatrix} \begin{bmatrix} \vec{p} \\ \lambda^* \end{bmatrix} = -\begin{bmatrix} \vec{g} \\ \vec{b} \end{bmatrix}$$

... "KKT System"

→ 굳이 이렇게 \vec{x}^* 가 아니라 \vec{x} 에서 \vec{x}^* 로 가기 위한 방법과
거의인 \vec{p} 를 미지수로 다시 시도하여 문제를 푸는 이유는,
nonlinear constrained optimization 문제의 대략에서 QP 문제를

풀 때나 유사하다. General Nonlinear Constrained Optimization에서는 일반적으로 한 번의 eqn solving으로 \vec{x}^* 를 찾을 수 있고 반복적인 iteration을 통해 \vec{x}^* 를 찾아야 하기 때문이다. 이때는, 위 KKT System을 풀면 \vec{p} (혹은 \vec{x}^*)와 \vec{x}^* 를 구할 수 있다.

QP의 대안

KKT System을 푸는 방법은 2가지로 크게 나눌 수 있음.

(강 $\tilde{A} \vec{x} = \vec{b}$ 형식 풀기)

일반적 least-square 문제

3D 간접할 수 있음.

(i.e. Unconstrained quadratic function optimization)

Direct Solution

Factoring methods

(\vec{x} 를 찾는 방법, 정부호 \uparrow
but 계산량/시간 오래
 \downarrow)

LU 분해 (High)

LDLT 분해 (대칭, indefinite)

Cholesky 분해 (positive definite)

Schur-complement method

: \vec{x} 는 \vec{p} 를 소거해 \vec{x} 를 작은 차원으로 바꾸어 풀기
(계산량 줄임 가능)

Null-space method

: Constraints를 만족시키는 Subspace에 대한 해를 찾기
(구조 조건 적용 때 유용)

:

Iterative methods

Conjugate Gradient

★ 제일 강력.

GMRES (Generalized Minimal RESidual Method)

: least-square prblm (= QP 문제)을 푸는 방법 중 하나이며

Krylov subspace iterative method / 기반으로 문제를 풀어간다.
시스템이 비정방 (non-square matrix)일 때도 사용 가능.

QMR (Quasi-Minimal Residual Method) : light version of GMRES

LQR (Least-Squares QR method) : Least-square prblm

(= QP 문제)을 푸는 Krylov subspace 방법.稠密 matrix)가 sparse
인 non-square 일 때 효과적. Implicit QR 분해 과정도 포함된다.

:

8-2) Penalty and Augmented Lagrangian Methods.

↳ 알고리즘은 아니고 일종의 문제변형 framework. 각 iteration마다 그 다음 차례에

- Penalty methods & Augmented methods 구하는 난 기준 unconstrained opt 알고리즘 사용

Constraints가 있는 기준의 opt probm은 obj function을, constraints function을 포함하는 penalty function을 augments한 새로운 obj function으로 바꾸어 unconstrained opt probm을 새로이 정의하여 거기서 optimum을 찾는 methods를 일컫는다. 아래와 같이 표현할 수 있다.

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

$$\text{s.t. } \hat{C}_i(\vec{x}) = 0$$

$$C_i(\vec{x}) \geq 0$$

Constrained Opt Prblm

Augmented objective function with penalty function.

$$\min_{\vec{x} \in \mathbb{R}^n} \hat{f}(\vec{x}) = f(\vec{x}) + \phi(\vec{C}(\vec{x}), \vec{\hat{C}}(\vec{x}))$$

Penalty function

Unconstrained Opt probm

여기서는 다음 3가지 methods를 다룬 것이다.

- Quadratic Penalty Methods.
- NonSmooth Exact Penalty Methods.
- Augmented Lagrangian Methods. ☆ 제일 유용.

8-2-1) Quadratic Penalty Method.

Quadratic Penalty method는 기존 opt probm이 가지는 constraint를 제거하여 obj factor를 늘리고 objective function이 더하여 새로운 objective function을 가지는 unconstrained optimization을 푸는 method이다.

< Equality Constrained Optimization의 대안 Quadratic Penalty Method >

Equality Constrained Opt Prblm → Unconstrained Opt Prblm

$$\min_{\vec{x} \in \mathbb{R}^n} f(\vec{x}) ; \text{smooth \& differentiable}$$

$$\text{s.t. } \hat{C}_i(\vec{x}) = 0 ; i=1, \dots, m$$

$$\min_{\vec{x} \in \mathbb{R}^n} Q(\vec{x}) = f(\vec{x}) + \frac{\mu}{2} \sum_{i=1}^m \hat{C}_i^2(\vec{x})$$

Quadratic penalty function
Penalty Parameter; $\mu > 0$

이렇게 함으로써 제약조건을 위배하면 obj func이 커지는 효과(penalty 효과)를 얻을 수 있음. 이렇게 새롭게 된 새 opt probm의 obj func $Q(\vec{x})$ 를 Quadratic Penalty function이라고 한다. 여기서 constraints의 제곱 sum에 곱해지는 factor μ 는 penalty parameter라고 부르며 $\mu > 0$ 으로 선택된다. μ 가 크면 constraint violation에 대한 weight이

구조가 있어 그것을 Constraint violation을 작게 만들도록 (constraint를 만족시키는 방향으로 움직이도록)

\vec{x} update 방향을 유도할 것이다. 또한 새로 만들어진 $Q(\vec{x})$ 또한 smooth & differentiable 이기 때문에 그가 앞의 어떤 unconstrained optimization algorithm으로 optimum point를 찾을 수 있을 것이다.

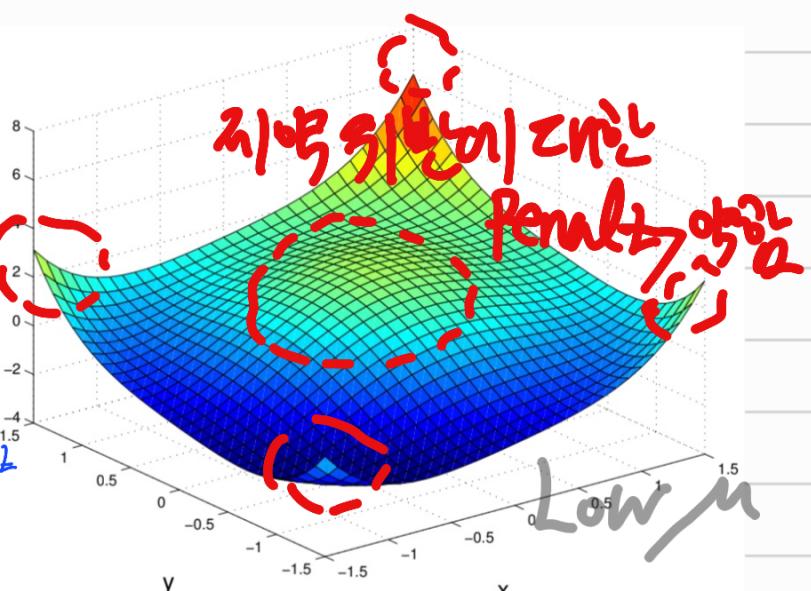
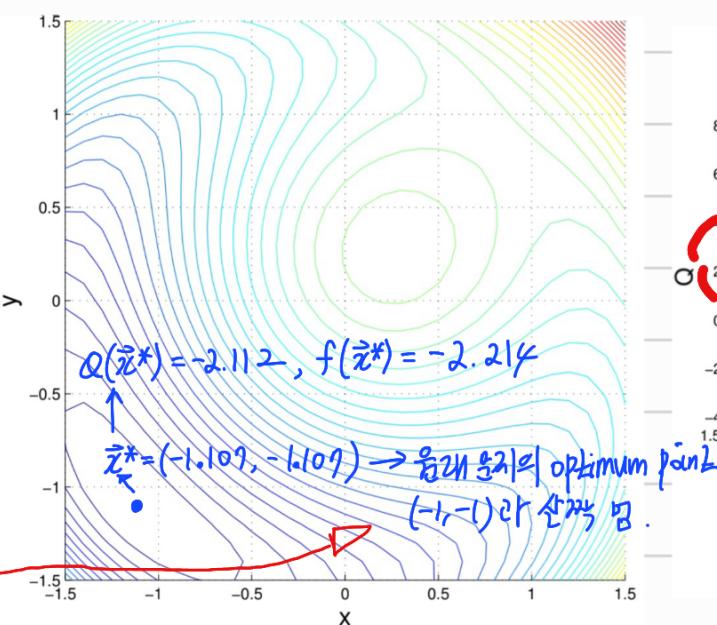
ex 1) Equality Constrained Opt Prblm

$$\min_{\vec{x} \in \mathbb{R}^2} f(\vec{x}) = x_1 + x_2$$

$$\text{s.t. } x_1^2 + x_2^2 - 2 = 0$$

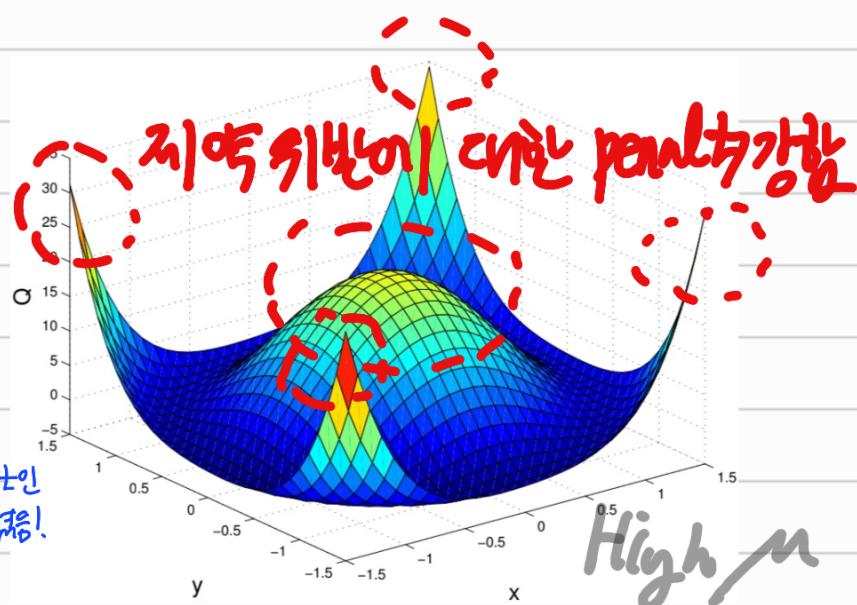
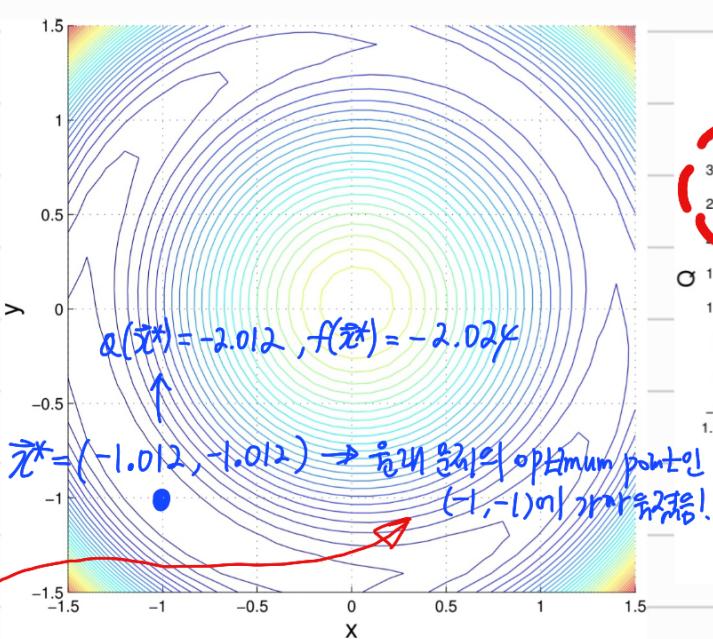
Quadratic Penalty Method (Unconstrained Opt Prblm)

$$\min_{\vec{x} \in \mathbb{R}^2} Q(\vec{x}) = x_1 + x_2 + \frac{M}{2} (x_1^2 + x_2^2 - 2)^2$$



Constraint weight가 비교적 낮을 때 $M=1$ 일 때.

$$\text{i.e., } Q(\vec{x}) = x_1 + x_2 + \frac{1}{2} (x_1^2 + x_2^2 - 2)^2 \quad |+x_1=0$$



Constraint weight가 비교적 높을 때 $M=10$ 일 때.

$$\text{i.e., } Q(\vec{x}) = x_1 + x_2 + 5 (x_1^2 + x_2^2 - 2)^2$$

\therefore Penalty parameter μ 의 값을 커울수록 constraint violation에 대한 penalty (weight)가 커져서 제약조건을 더 잘 지키게 된다.

$$Q(\vec{x}) = f(\vec{x}) + \frac{1}{2} \mu \sum_i (\hat{C}_i(\vec{x}))^2$$

... Big ...
... Small ...

\rightarrow 즉 $f(x)$ 와 \hat{C}_i 에 걸리는 상대적 weight는 μ 의 크기에 따라 결정

그럼에도 불구하고, Quadratic Penalty Method를 사용하여 찾았던 optimum point $\vec{x}^* = (-1.012, -1.012)$ 가 제약조건을 충족하지는 못시키는 것을 확인할 수 있다. $\rightarrow \underline{x_1^{*2} + x_2^{*2} - 2 = 0.04829 \neq 0}$

Penalty method의 약점은 (수학적으로 엄밀하지 않음)을 확인할 수 있는 부분이다.

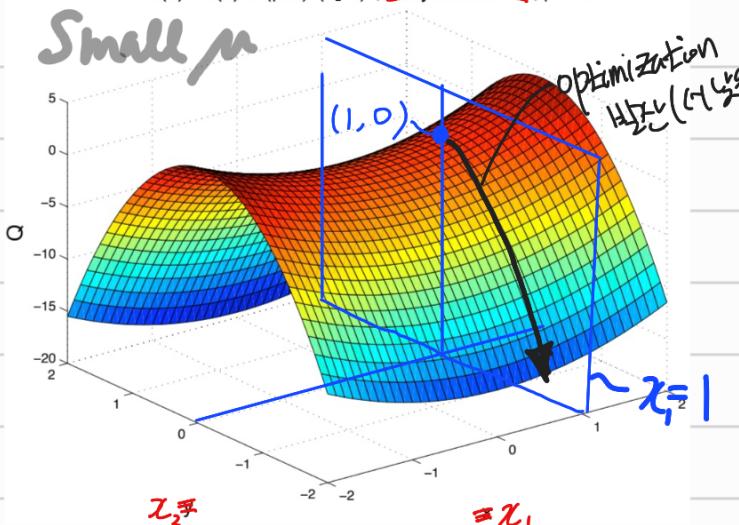
하지만 scheme이 매우 간단하여 μ 만 조금 손보면 비교적 간편하게 optimum point의 가까운 지점을 찾을 수 있다는 장점이 있다.

ex 2) Equality Constrained Opt Prblm Quadratic Penalty Method (Unconstrained Opt Prblm)

$$\min_{\vec{x} \in \mathbb{R}^2} f(\vec{x}) = -5x_1^2 + x_2^2 \quad \rightarrow \min_{\vec{x} \in \mathbb{R}^2} Q(\vec{x}) = -5x_1^2 + x_2^2 + \frac{\mu}{2}(x_1 - 1)^2$$

$$\text{s.t. } x_1 = 1$$

$$Q(x) = f(x) + (\mu/2)*(x_1 - 1)^2, \text{ subject to } x_1 = 1; \mu = 1$$



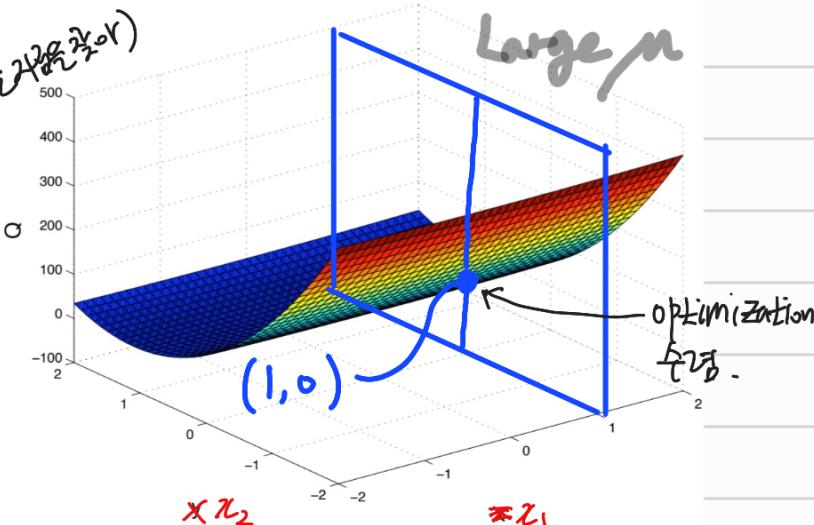
$\hookrightarrow \mu = 1$ 로 setting 했을 때.

원래 $f(\vec{x})$ 의 optimum point는 $(1, 0)$ 이지만 $Q(\vec{x})$ 함수 개성이 아래로 빨간색에 여겨, 이런 이형 함수를 대상으로 약속을 나면 빨간색이다. $\mu < 10$ 인

μ 를 선택시 $Q(\vec{x})$ 가 (-) 방향으로 unbound 개성이 있어서 빨간색을 벗어날 수 있음.

$\therefore \mu$ 를 너무 작게 선택하면 $Q(\vec{x})$ 함수에 대한 약속이 빨간색을 벗어날 수도 있다.

$$Q(x) = f(x) + (\mu/2)*(x_1 - 1)^2, \text{ subject to } x_1 = 1; \mu = 100$$



$\rightarrow \mu = 100$ 으로 setting 했을 때.

간단히 $Q(\vec{x})$ 함수의 (-) 방향을 막기로 한 것을 볼 수 있다.
따라서 $f(\vec{x})$ 의 optimum point인 $(1, 0)$ 으로 수렴할 수 있게 된다.

< Inequality Constraints 가 포함된 General Optimization or 대입 Quadratic Penalty Method >

General Constrained Opt Prblm $\xrightarrow[\text{Quadratic Penalty Method}]{}$ Unconstrained Opt Prblm

$\min f(\vec{x})$; smooth & differentiable
 $\vec{x} \in \mathbb{R}^n$

s.t. $\hat{C}_i(\vec{x}) = 0$; $i = 1, \dots, m$

$\underline{\hat{C}_i(\vec{x})} \geq 0$; $i = 1, \dots, p$

$$\min_{\vec{x} \in \mathbb{R}^n} Q(\vec{x}) = f(\vec{x}) + \frac{M}{2} \sum_{i=1}^m \hat{C}_i^2(\vec{x}) + \frac{M}{2} \sum_{i=1}^p [C_i(\vec{x})^-]^2$$

where $[C_i(\vec{x})]^- = \max(-C_i(\vec{x}), 0)$... Penalty Switch of $C_i(\vec{x})$

$[C_i(\vec{x})]^-$ 의 역할은, 현재 점 \vec{x} 가 어떤 inequality constraint 위반 여부를 확인하여 위반하지 않았으면 (즉 $C_i(\vec{x}) \geq 0$ 이면) 해당 constraint C_i 에 대한 penalty term을 0으로 만들어서 아니면 그 constraint가 Q 에 영향을 끼치지 못하게 하여 obj func f 를 낮추고 equality constraints \hat{C}_i 를 자가는 방향으로 가중을 두어 \vec{x} 를 업데이트하거나

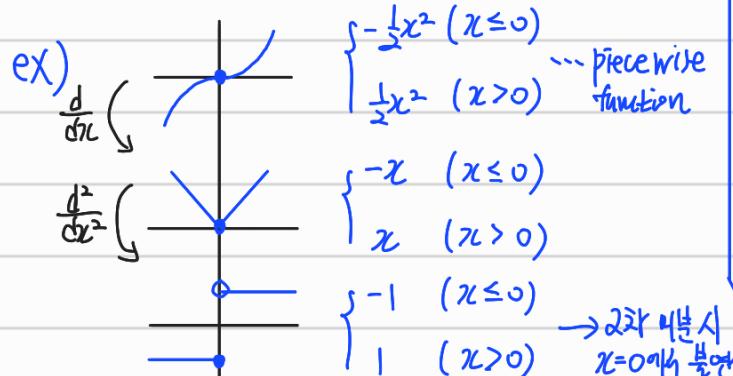
위반을 했다면 (즉 $C_i(\vec{x}) < 0$ 이면) 해당 constraint C_i 에 대한 penalty term을 살려서 해당 constraint의 violation 정도를 낮추는 방향으로 \vec{x} 가 update되게끔 한다. 그래서 $[C_i(\vec{x})]^-$ 는 부등호의 약근간 $C_i(\vec{x})$ 의 penalty 효과를 끊어 끊어 할 수 있게 해주는 일종의 penalty switch인 것이다.

* $[C_i(\vec{x})]^-$ 항의 존재로 인해 $Q(\vec{x})$ 는 더 이상 2번 맵을 가능한 함수가 아님. 따라서 2차 미분 정보를 필요로 하는 Newton's method는 inequality constraint가 포함된

Proof) $\{[C_i(\vec{x})]^- \}^2 = \begin{cases} 0 & \text{If } -C_i(\vec{x}) \leq 0 \\ C_i(\vec{x}) & \text{If } -C_i(\vec{x}) > 0 \end{cases}$... Piecewise function. Quadratic Penalty Method에는 사용할 수 없다. 대신

$\nabla_{\vec{x}} (\{[C_i(\vec{x})]^- \}^2) = \begin{cases} \vec{0} & \text{If } -C_i(\vec{x}) \leq 0 \\ \left[2C_i \cdot \frac{\partial C_i}{\partial x_j} \dots \right] & \text{If } -C_i(\vec{x}) > 0 \end{cases}$ Steepest descent / CGM / Quasi-Newton's method 최점 Hessian을 직접적으로 쓰지 않는 method를 써야 한다.

$\nabla_{\vec{x}}^2 (\{[C_i(\vec{x})]^- \}^2) = \begin{cases} \vec{0} & \text{If } -C_i(\vec{x}) \leq 0 \\ \left[\frac{\partial^2 C_i}{\partial x_k \partial x_j} \dots \right] & \text{If } -C_i(\vec{x}) > 0 \end{cases}$ Hessian matrix C_i의 존재로 인해 $C_i(\vec{x}) = 0$ 인 지점에서 1차 derivative가 $\vec{0}$ 로 연속.



사용으로 Matlab TO 99 line code! Optimality Criteria 시전에 비슷한 Switch이나 있음. 이걸 통한 기준... * 참고

$[C_i(\vec{x})]^- = \max(-C_i(\vec{x}), 0)$

$= \begin{cases} 0 & \text{If } -C_i \leq 0 \\ -C_i & \text{If } -C_i > 0 \end{cases}$ ↑ 동일

$-[C_i(\vec{x})]^+ = -\min(C_i(\vec{x}), 0)$

$= \begin{cases} 0 & \text{If } C_i \geq 0 \\ -C_i & \text{If } C_i < 0 \end{cases}$



Quadratic Penalty Method에 대한 Algorithm Scheme.

실제 Quadratic Penalty method를 implement할 때는 앞서 배운 구조와는 다르게 매 k 번째 iteration마다 quadratic penalty function $Q_k(\vec{x}_k, \mu_k)$ 을 update하여 Q_k 에 대한 local optimum \vec{x}_k^* 을 찾는 방식으로 원래의 constrained optimization problem의 다른 optimum point를 찾아나간다. $Q_k(\vec{x}_k, \mu_k)$ 는 μ_k 를 매 iteration마다 조금씩 증가시킴으로써 update된다. ↗ 각 iteration마다 optimizable 함수의 개수가 조금씩 늘어나게 된다.

μ_k 는 iteration마다 전해온 변수로 증가시키는 이유는, 처음에는 낮은 μ_k 로 시작해 $f(\vec{x})$ 의 weight를 $C_i(\vec{x})$ 의 weight 대비 상대적으로 커워 $f(\vec{x})$ 가 감소하는 point를 찾고, 이걸 어느정도 $f(\vec{x})$ 값이 충분히 낮아졌으면 μ_k 를 커서 $C_i(\vec{x})$ 에 대한 weight를 늘려 constraint를 만족하는 방향으로 point를 찾아나가기 위함이다.

Iteration의 종료는 → 내부 iteration 종료 : $\nabla Q_k(\vec{x}_k, \mu_k) \approx 0$

외부 " " : BE Constraints 충분히 만족시.

For finding optimum point of Q_k (이전 step의 방법과 동일한 method를 사용하는 Step)

내부 Iteration

외부 Iteration

Algorithm 1 Quadratic Penalty Algorithm

```

1: procedure
2: Given  $\mu_0 > 0$ , a non-negative sequence  $\tau_k$  with  $\tau_k \rightarrow 0$ , and a starting point  $\mathbf{x}_0^s$ 
3: for  $k = 0, 1, 2, \dots$  do
4:   Find an approximate minimizer  $\mathbf{x}_k$  of  $Q(\mathbf{x}_k; \mu_k)$ , starting at  $\mathbf{x}_k^s$ , and terminating when
5:    $\|\nabla_{\mathbf{x}} Q(\mathbf{x}_k; \mu_k)\| < \tau_k$ ; → 내부 iteration 종료 기준 ( $\mathbf{x}_k$ 는 외부 iteration 진행됨에 따라 점점 갱신된다)
6:   if (Final Convergence Test Satisfied) then
7:     Stop with approximate solution  $\mathbf{x}_k$  → 외부 iteration 종료 기준.
8:   end if
9:   Choose new penalty parameter,  $\mu_{k+1} > \mu_k$ 
10:  Choose new starting point,  $\mathbf{x}_{k+1}^s$ ;
11: end for
12: end procedure
    
```

$$\mu_{k+1} = 5/\mu_k$$

$$\begin{aligned} &\text{L} \rightarrow \|\hat{C}_i(\vec{x}_k)\| < \varepsilon_c \rightarrow \text{충분히 저렴은} \\ &\text{&} \quad \hat{C}_j(\vec{x}_k) \geq 0 \end{aligned}$$

For
optimization
process.

For
updating
 Q_k

8-2-2) Non Smooth Exact Penalty Methods. (노이거)

<General Constrained Optimization problem에 대한 Non-smooth Exact Penalty Method>

General Constrained Opt Prblm $\xrightarrow{\text{Non-Smooth Exact}} \text{Unconstrained Opt Prblm}$

$\min_{\vec{x} \in \mathbb{R}^n} f(\vec{x})$; smooth & differentiable

s.t. $\hat{C}_i(\vec{x}) = 0$; $i = 1, \dots, m$

$C_i(\vec{x}) \geq 0$; $i = 1, \dots, p$

일부 원래 non-smooth/non-differentiable 함

$$\min_{\vec{x} \in \mathbb{R}^n} \phi(\vec{x}) = f(\vec{x}) + \mu \sum_{i=1}^m |\hat{C}_i(\vec{x})| + \mu \sum_{i=1}^p [C_i(\vec{x})]^-$$

$$\text{where } [C_i(\vec{x})]^- = \max(-C_i(\vec{x}), 0) \dots \text{Penalty Switch of } C_i(\vec{x})$$

→ 앞선 8-2-1의 Quadratic Penalty Method에서는 L_2 -norm 형식으로 $C_i(\vec{x})$ 함수의 제곱을 더하여 penalty function Q 를 구성했었는데, Non-smooth Exact Penalty Method에서는 L_1 -norm 형식, 즉 $C_i(\vec{x})$ 함수의 absolute value를 더하여 penalty function ϕ 를 구성한다.

• Non-Smooth Exact Penalty Method의 특징.

1) Constraint 조건이 L_1 -norm, 즉 absolute를 쓰우기 때문에 일부 $\hat{C}_i(\vec{x})$ 은 non-smooth, 즉 non-differentiable 하기 된다. \rightarrow gradient-based unconstrained optimization 알고리즘 (steepest descent, conjugate gradient, ...)을 못 쓴다. 당연히 Hessian 기반의 Newton 기법 method도 못 쓰다 그걸 어렵게 하니? \rightarrow 사실상 다른 계열의 알고리즘 (Active-set, Subgradient, Bundle, ...)을 적용해야지만 Non-Smooth Exact Penalty Method를 실제로 수행시킬 수 있음. \rightarrow 가능하기가 같은 method.

2) 이를적으로 $M \rightarrow \infty$ 여야지만 원래의 constrained optimization problem의 exact한 optimum point가 도달할 수 있는 Quadratic Penalty Method와 다르게,

Non-Smooth Exact Penalty Method는 충분히 큰 finite한 M 을 가지고도 원래 최적화 문제의 exact한 optimum point가 도달 가능하다.

$\begin{array}{l} \text{Quadratic} \\ \text{Penalty} \\ \text{Method} \\ \text{의 penalty term} \end{array}$ $\rightarrow M \left\ \hat{C}_i(\vec{x}) \right\ ^2 \xrightarrow{\text{미분}} 2M \hat{C}_i \nabla \hat{C}_i$	$\begin{array}{l} \text{Non-smooth} \\ \text{Exact} \\ \text{Penalty method} \\ \text{의 penalty term} \end{array}$ $\rightarrow M \left \hat{C}_i(\vec{x}) \right \xrightarrow{\text{미분}} M \nabla \hat{C}_i$	저마다 $\hat{C}_i(\vec{x})$ 자체가 미분 가능하지 않아 $\left\ \hat{C}_i(\vec{x}) \right\ ^2$ 항의 gradient 자체가 작아지게 만든다. 대체로 ∇f 와 $\nabla \hat{C}_i$ 중 $\nabla \hat{C}_i$ 에 대한 weight(M)을 키우자 보면 안되는 (왜냐하면 $C_i(\vec{x})$ 가 저마다 미분을 허용하지 않아서 penalty effect가 작아지기 때문에) 상황이 된다.
---	---	---

\hookrightarrow 반면 Non-smooth Exact Penalty Method에서는 gradient의 외부값이 그대로 gradient를 사용할 수 있으나 굳이 penalty parameter M 을 제한해도 충분할 필요가 없어. 또한 이 특성으로 인해 Quadratic Penalty Method 보다 상당히 빠르게 \vec{x}^* 에 도달할 수 있다.

Non-Smooth Exact Penalty Method의 알고리즘 scheme은 사실상 Quadratic Penalty Method의 그것과 거의 동일하다. $Q_k(\vec{x}_k, \mu_k)$ 의 형태만 L_1 -norm penalty function으로 서주면 된다.

Algorithm 1 Quadratic Penalty Algorithm (Non-smooth Exact Penalty Algorithm)

```

1: procedure
2:   Given  $\mu_0 > 0$ , a non-negative sequence  $\tau_k$  with  $\tau_k \rightarrow 0$ , and a starting point  $\mathbf{x}_o^s$ 
3:   for  $k = 0, 1, 2, \dots$  do
4:     Find an approximate minimizer  $\mathbf{x}_k$  of  $Q(\mathbf{x}_k; \mu_k)$ , starting at  $\mathbf{x}_k^s$ , and terminating when
5:      $\|\nabla Q(\mathbf{x}_k; \mu_k)\| < \tau_k$ ;  $\rightarrow$  내부 iteration 종료 기준 ( $\mathbf{x}_k$ 는 외부 iteration 진행됨에 따라 점점 갑소시킨다)
6:     if (Final Convergence Test Satisfied) then
7:       Stop with approximate solution  $\mathbf{x}_k$ 
8:     end if
9:     Choose new penalty parameter,  $\mu_{k+1} > \mu_k$ ;
10:    Choose new starting point,  $\mathbf{x}_{k+1}^s$ ;
11:  end for
12: end procedure
  
```

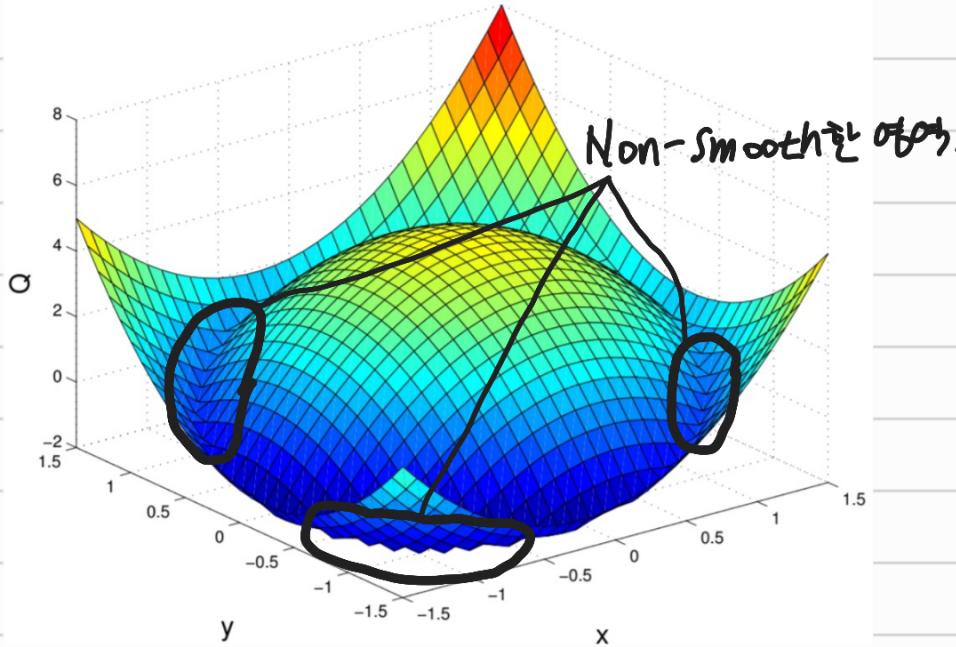
For finding
optimum point of Q_k
 ↓
 내부 Iteration

↓
 2번 Iteration

↓
 2번 Iteration

↓
 For
Updating
 Q_k

$\left(\begin{array}{l} \|\hat{C}_i(\vec{x}_k)\| < \epsilon_c \rightarrow$ 충분히 저마다 $\hat{C}_i(\vec{x}_k)$ 가
 $\hat{C}_i(\vec{x}_k) \geq 0 \end{array} \right)$ 및
 optimization 종료.



$$\phi(\vec{x}) = x_1 + x_2 + \mu |x_1^2 + x_2^2 - 2| ; \mu = 2$$

\times "Quadratic Penalty Method / Non-smooth Exact Penalty Method" 를
 μ 가 너무 크거나 작으면 ill-conditioned 한 Hessian 을 가지게 됨.

What is ill-Conditioned Hessian?

$$\tilde{H} = \begin{bmatrix} \nabla^2 Q_{11} & \nabla^2 Q_{12} & \cdots \\ \nabla^2 Q_{21} & \nabla^2 Q_{22} & \cdots \\ \vdots & & \ddots \end{bmatrix} \rightarrow |\text{Largest component} - \text{smallest component}| \text{ 가 } \\ \text{너무크면 ill-conditioned 이다.} \text{ 향.} \\ \text{이를 나타내는 특수치로서} \quad \text{Eigenvalue of } \tilde{H} \\ \text{Condition number} = \frac{\lambda_{\max}}{\lambda_{\min}} \text{ 가 } \infty.$$

Condition number \uparrow : Ill-conditioned matrix \tilde{H} .

$\rightarrow \tilde{H}^{-1}$ 의 계산이 아주 조심스러워도 많이 오류될 수 있음. 이로 인한 numerical instability 가능할 수 있음. \rightarrow Search direction 계산 불안정.
 \rightarrow slow convergence.

8-2-3) Augmented Lagrangian Method (ALM) \times 핫한 주제이지만 미안한데, 여기서부터는 등점법의 대각방식을 이용(?)하거나 단지 λ로, 부분점법의 " " 이용(λ)까지와 달리 고려해야

$$\begin{array}{ll} \min_{\vec{x} \in \mathbb{R}^n} f(\vec{x}) & \text{원래 등점법 문제} \\ \text{s.t. } \hat{C}_i(\vec{x}) = 0 ; i=1 \sim m & \end{array} \xrightarrow{\hspace{1cm}} \begin{array}{l} \min_{\vec{x} \in \mathbb{R}^n} L_A(\vec{x}, \lambda, \mu) \\ = f(\vec{x}) - \sum_{i=1}^m \lambda_i \hat{C}_i(\vec{x}) + \frac{\mu}{2} \sum_{i=1}^m \hat{C}_i^2(\vec{x}) \end{array} \begin{array}{l} \text{증가된 바이어스 문제} \\ \text{Augmented Lagrangian Function} \end{array}$$

where λ_i, μ 는 우리가 설정한 parameter

↓ Lagrangian Function.

$$L(\vec{x}, \lambda) = f(\vec{x}) - \sum_{i=1}^m \lambda_i \hat{C}_i$$

↓ Optimality Condition for Lagrangian Multiplier Theorem

At optimal point (\vec{x}^*, λ^*) : f 의 optimal point

$$\nabla_{\vec{x}} L(\vec{x}^*, \lambda^*) = \nabla f(\vec{x}^*) - \sum_{i=1}^m \lambda_i^* \nabla \hat{C}_i(\vec{x}^*) = \vec{0} \quad \dots (a)$$

$$\nabla_{\lambda} L(\vec{x}^*, \lambda^*) = -\vec{C}(\vec{x}^*) = \vec{0} \quad \dots (d)$$

1st order Optimality NC

$$\nabla_{\vec{x}} L_A(\vec{x}^*) = \vec{0}; \quad \text{기본적으로 } \vec{x}^* \text{는 } L_A \text{의 optimal point of } L_A.$$

$$\nabla_{\vec{x}} L_A(\vec{x}^*) = \nabla f(\vec{x}^*) - \sum_{i=1}^m ((\lambda_i - \mu \hat{C}_i(\vec{x}^*)) \nabla \hat{C}_i(\vec{x}^*)) = \vec{0}$$

$$\boxed{\text{Let } \bar{\lambda}_i = \lambda_i - \mu \hat{C}_i(\vec{x}^*)}$$

$$\nabla_{\vec{x}} L_A(\vec{x}^*) = \nabla f(\vec{x}^*) - \sum_{i=1}^m \bar{\lambda}_i \nabla \hat{C}_i(\vec{x}^*) = \vec{0} \quad \dots (b)$$

$$\text{이 때, } \hat{C}_i(\vec{x}^*) = -\frac{1}{\mu} (\bar{\lambda}_i - \lambda_i) \quad \dots (c)$$

• ALM strategy의 원리 (와 유사한 점은 같은 문제에 대한 접근법; Nocedal 23주 (자세히 나옴))

만약 $\vec{x}^* = \vec{x}^*$ 라면, 즉 여기서 찾은 \vec{x}^* 가 원래의 제약 문제의 전자 optimum point \vec{x}^* 라면 여기서의 $\bar{\lambda}_i$ 또한 $\bar{\lambda}_i = \lambda_i^*$ 일 것이다. 왜냐하면 (a) = (b) 여야 하기 때문이다. 자 여기서 한 번 사용해보자.

우리가 LA함수를 만들 때 임의로 선택해야 하는 값인 μ 와 λ_i 를 다음과 같이 $\bar{\lambda}_i = \lambda_i - \mu \hat{C}_i(\vec{x}^*) = \lambda_i^*$ 가 되도록, 즉 LA의 \vec{x}^* 이 \vec{x}^* 이 되고 $\bar{\lambda}_i = \lambda_i^*$ 가 되도록 설정했다고 하자. 이 때 $\vec{x}^* = \vec{x}^*$ 이기 때문에 $\hat{C}_i(\vec{x}^*) = 0$ 이 되고 따라서 $\bar{\lambda}_i = \lambda_i = \lambda_i^*$ 이 되게 된다. 즉 아래 층계 λ_i 를 잘 설정한다면 같은 원래의 제약 문제에서의 λ_i^* 와 같은 값을 $\bar{\lambda}_i$ 로 설정하는 걸 뜻한다. 한편 놓고 $\hat{C}_i(\vec{x}^*) = -\frac{1}{\mu} (\bar{\lambda}_i - \lambda_i)$ 이며 $\bar{\lambda}_i = \lambda_i = \lambda_i^*$ 이면 당연히 $\hat{C}_i(\vec{x}^*) = 0$ 이 된다. 여기서 중요 한 것은 $\bar{\lambda}_i$ 와 λ_i 가 같아지니 μ 의 값과 무관하게 $\hat{C}_i(\vec{x}^*) = 0$ 이 된다는 것이다. (c)

→ 이 말인즉슨 μ 가 어떤 값이든 상관없이 $\hat{C}_i(\vec{x}^*) = 0$ 이나, 기존의 penalty method (Quadratic Penalty Method, Non-Smooth exact penalty method)처럼 μ 를 아주 큰 값으로 설정할 필요가 없단 얘기다.

→ 한마디로, ALM 사용 시 λ_i ($i=1 \sim m$)의 값을 원래 제약 문제의 전자 optimum point에서 Lagrange 송수 λ_i^* 에 매우 첨예하게 설정하면 penalty parameter μ 를 비교적 작게 설정해도 $\hat{C}_i(\vec{x}) \approx 0$ 이 되어 제약 조건이 만족이 쉬워짐.

→ 이를 위해 얻는 이점: 굳이 큰 μ 를 설정하지 않아도 constraint violation이 크게 일어나지 않을 수 있으니, 작은 μ 를 설정할 수 있고 따라서 ill-conditioned matrix를 피할 수 있음.

→ 근데 처음부터 아래 층계 λ_i 를 λ_i^* 로 설정하는 어려움으로 다음 식으로 λ_i 를 update해나간다.

$$\lambda_i^{k+1} = \lambda_i^k - \mu_k \hat{C}_i(\vec{x}_k); \quad i=1 \sim m$$

즉, 다음 iteration에서의 LA함수의 penalty function에 쓰일 λ_i^{k+1} 는 이전 iteration에서 마치

원래 제약 문제에서의 Optimum Lagrange 송수를 묘사하는 개념으로 쓰인 $\bar{\lambda}_i^k = \lambda_i^k - \mu_k \hat{C}_i(\vec{x}_k)$ 로 update된다.

(" " " ") λ_i 로 갱신해 두기가 바라는 것이다 (대문)

* 설사 초기 optimization 후 최종적으로 converge된 λ_i 가 λ_i^* 와 정확히 일치하지 않더라도 그 차이가 충분히 작다면 Quadratic Penalty Method와 비교하여 충분히 작은 penalty parameter μ 로도 충분히 정확한 $\vec{x}^* \approx \vec{x}_k$ 를 찾을 수 있다.

$$\min_{\vec{x} \in \mathbb{R}^n} L_A(\vec{x}, \lambda, \mu)$$

$$= f(\vec{x}) - \sum_{i=1}^m \lambda_i \hat{C}_i(\vec{x}) + \frac{\mu}{2} \sum_{i=1}^m \hat{C}_i^2(\vec{x})$$

for k in $[0, 1, \dots, 100]$

$$\min_{\vec{x} \in \mathbb{R}^n} L_A^{(k)}(\vec{x}_k, \lambda_k, \mu_k)$$

$$= f(\vec{x}_k) - \sum_{i=1}^m \lambda_i^{(k)} \hat{C}_i(\vec{x}_k)$$

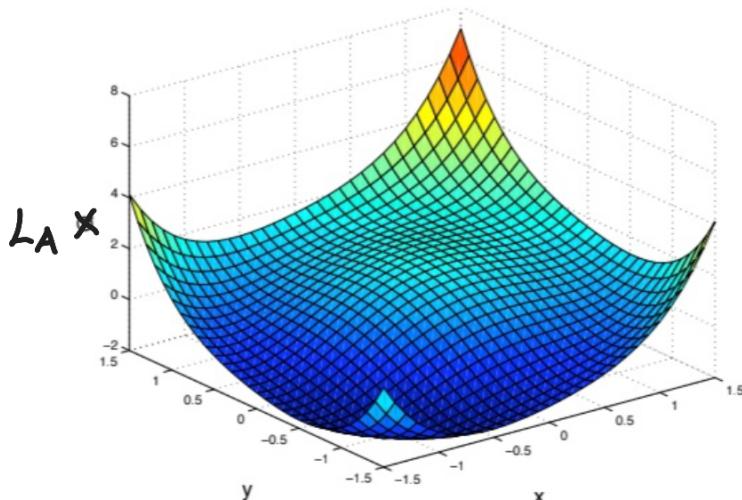
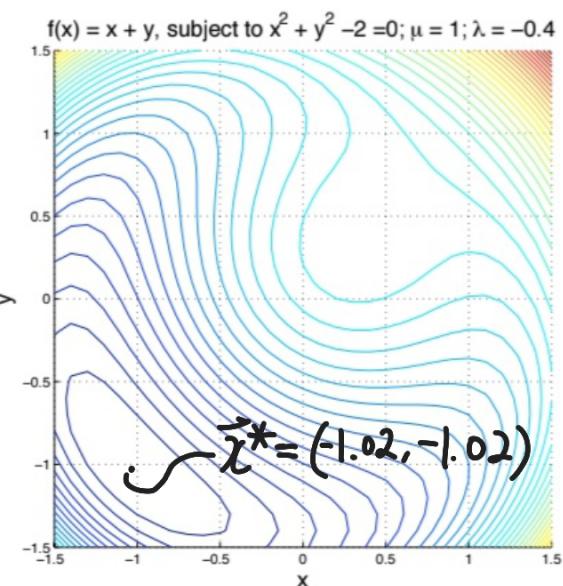
$$+ \frac{\mu}{2} \sum_{i=1}^m \hat{C}_i^2(\vec{x}_k)$$

Algorithm 2 Augmented Lagrangian Method

```

1: procedure
2:   Given  $\mu_0 > 0$ , tolerance  $\tau_0 > 0$ , starting points  $\mathbf{x}_0^s$  and  $\lambda^0$ .
3:   for  $k = 0, 1, 2, \dots$  do
4:     Find an approximate minimizer  $\mathbf{x}_k$  of  $\mathcal{L}_A(\mathbf{x}_k, \lambda^k; \mu_k)$ , starting at  $\mathbf{x}_k^s$ , and terminating
5:     when  $\|\nabla_{\mathbf{x}} \mathcal{L}_A(\mathbf{x}_k, \lambda^k; \mu_k)\| \leq \tau_k$ ;
6:     if (Final Convergence Test Satisfied) then
7:       Stop with approximate solution  $\mathbf{x}_k$ 
8:     end if
9:     Update  $\lambda_i^{k+1} = \lambda_i^k - \mu_k c_i(\mathbf{x}_k)$ ,  $i = 1, \dots, p$ 
10:    Choose new penalty parameter,  $\mu_{k+1} > \mu_k$ ;
11:    Choose new starting point,  $\mathbf{x}_{k+1}^s = \mathbf{x}_k$ ;
12:    Select tolerance  $\tau_{k+1}$ ;
13:  end for
14: end procedure

```



(b) Three-Dimensional view of the Modified Equation

(a) Contour of Modified Equation

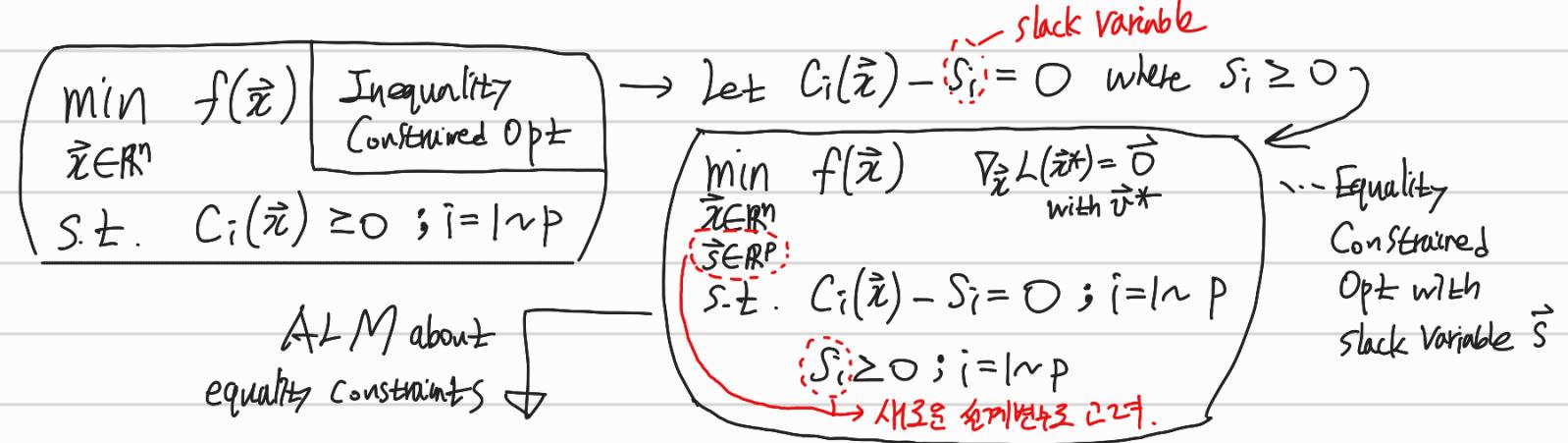
Figure 10: Augmented Lagrangian Method, $\mu = 1$, $\lambda = 0.4$

$\hookrightarrow \mu = 1$ 일 때 λ 는 λ^0 로 초기화 시켜야 하는 optimum point $\vec{x}^* = (-1, -1)$ 에 상당히 가깝게 optimum point $(-1.02, -1.02)$ 를 가지기 때문에 LA 함수가 잘 구동이 되었음.

\because ALM은 결코 Inexact하게 \vec{x}^* 를 찾을 수 없어 없음. 즉 결코에 찾았다는 \vec{x}^* 는 초기 세이프 투리의 \vec{x}^* 와 같은 하지 일치하는 어려움. 이런 고차 항수가 term으로 이어짐을 추가되는 penalty method의 근본적 한계임.

Inequality Constraints 및 general Constrained Opt ALM

(Nocedal & Wright - Numerical Optimization 17.4 (page 514) 참고)



$$\begin{array}{ll} \min_{\substack{\vec{x} \in \mathbb{R}^n \\ \vec{s} \in \mathbb{R}^p}} & L_A = f(\vec{x}) - \sum_i v_i (C_i(\vec{x}) - s_i) + \frac{\rho}{2} \sum_i (C_i(\vec{x}) - s_i)^2 \\ \text{s.t.} & s_i \geq 0 ; i=1 \sim p \end{array}$$

우리가 알기로 접하는 수.(등호지역 ALM이 적용되는 것)

... Inequality Constrained Opt with Slack Variable \vec{s}

$\rightarrow \nabla_{\vec{s}} L_A = \vec{0} \quad (\because \vec{s} \text{ 또는 } \vec{v} \text{의 } 1^{\text{st}} \text{ order NC for optimality } \vec{s} \text{은 } \vec{0})$

$$\rightarrow v_i - \rho(C_i(\vec{x}) - s_i) = 0 \rightarrow s_i = C_i(\vec{x}) - \frac{v_i}{\rho} \rightarrow 2C_i \leq s_i \geq 0 \text{ 이어야 한다 (by 정의)} \quad \downarrow$$

$$\rightarrow \therefore s_i = \max\left(C_i(\vec{x}) - \frac{v_i}{\rho}, 0\right) \quad \downarrow$$

$$\begin{aligned} \rightarrow L_A = & \left[f(\vec{x}) - \sum_i \frac{v_i^2}{\rho} + \frac{\rho}{2} \sum_i \frac{v_i^2}{\rho} \right] = f(\vec{x}) - \frac{1}{2\rho} \sum_i v_i^2 \quad \cdots C_i(\vec{x}) - \frac{v_i}{\rho} \geq 0 \quad \uparrow \\ & \quad \downarrow \text{C}_i(\vec{x}) \text{에 대한 penalty term 있음} \\ & f(\vec{x}) - \sum_i v_i C_i(\vec{x}) + \frac{\rho}{2} \sum_i C_i^2(\vec{x}) \quad \cdots C_i(\vec{x}) - \frac{v_i}{\rho} < 0 \quad \downarrow \\ & \quad \downarrow \text{C}_i(\vec{x}) \text{에 대한 penalty term 있음.} \end{aligned}$$

$\rightarrow \text{Penalty Switch On } \vec{v} \text{ iff } \nabla_{\vec{x}} L_A = \vec{0} \text{ 조건을 사용해보자.}$

$$\nabla_{\vec{x}} L_A = \nabla_{\vec{x}} f(\vec{x}^*) - \sum_i (v_i - \rho C_i(\vec{x}^*)) \nabla_{\vec{x}} C_i(\vec{x}^*) = \vec{0}$$

Let \vec{v}_i

$$\rightarrow \nabla_{\vec{x}} L_A = \nabla_{\vec{x}} f(\vec{x}^*) - \sum_i \vec{v}_i \nabla_{\vec{x}} C_i(\vec{x}^*) = \vec{0} \quad \cdots \text{만약 } \vec{x}^* = \vec{x}^k \text{이면 } \vec{v}^* = \vec{v}^k \text{ 일 거임.}$$

“각각 등호지역 ALM이 적용 가능한지로 $\vec{v} = \vec{v}^*$ 인 \vec{v} 만 고른다면, (이를적으로는) ρ 를 충분히 고르면 언제 $C_i(\vec{x}^*) = 0$ 을 항상 만족할 수 있거” 됨. “각각 등호지역 ALM과 동일한 원칙로 다음과 같다.” | 때 iteration에서의 v_i^{k+1} 를 update할 수 있음. 또한 때 iteration에서의 L_A^k 는 아래와 같이 쓸 수 있음

$$\rightarrow \begin{cases} v_i^{k+1} = \max(\vec{v}_i^k, 0) = \max(v_i^k - \rho C_i(\vec{x}^k), 0) \dots \because \text{부등호제약의 나그랑지승수는 항상 } 0 \text{ 이상이어야 함} \\ L_A^k = f(\vec{x}) - \sum_i v_i^k (C_i(\vec{x}) - \max(C_i(\vec{x}) - \frac{v_i^k}{\rho}, 0)) + \frac{\rho}{2} \sum_i (C_i(\vec{x}) - \max(C_i(\vec{x}) - \frac{v_i^k}{\rho}, 0))^2 \end{cases}$$

$$L_A^k = f(\vec{x}) - \sum_i v_i^k \left(C_i(\vec{x}) + \left[C_i(\vec{x}) - \frac{v_i^k}{\rho} \right]^- \right) + \frac{\rho}{2} \sum_i \left(C_i(\vec{x}) + \left[C_i(\vec{x}) - \frac{v_i^k}{\rho} \right]^- \right)^2$$

8-3) Sequential Quadratic Programming (SQP)

→ non-linear Constrained Optimization을 푸는 가장 powerful하고 빠른 method

자 이전에는 Penalty Method를 사용해 원래의 constrained opt prob을 unconstrained opt prob으로 바꾸어 푸는 방식이 아니라, Constrained 형식 그대로 문제를 풀어나가는 방법을 배운다. 여러 방법이 있겠지만 가장 대중적이고 간단한 method 중 하나로 알려진 Sequential Quadratic Programming, 즉 SQP를 배우보자. 일단 equality-constrained non-linear optimization에 대한 SQP부터 알아본다. 그다음으로 Equality-Constrained SQP 알아보기 전에, 일단 Iterative Newton's Method로 equality-Constrained Non-linear Opt를 푸는 approach를 살펴본다.

8-3-1) Equality Constrained Non-linear Optimization

• "Iterative Newton's Method" for Equality Constrained Non-linear Optimization

• 매 번째 iteration point인 $\vec{x}_k, \vec{\lambda}_k$ 에서 현재 비선형 제약 문제의 Lagrangian Function L 을 Quadratic Function으로 근사시키고, 근사된 함수의 minimum($\nabla L = \vec{0}$)을 만족하는 $\vec{x}_k^*, \vec{\lambda}_k^*$ 을 찾아 해당 점을 다음 iteration point로 삼는 과정을 반복함으로써 (using Newton-method) 현재 고지약 문제의 KKT Condition($\nabla L = \vec{0}$)을 만족하는 global $\vec{x}^*, \vec{\lambda}^*$ 를 찾으나 한다.

<수식 증거 통해 Iterative Newton's Method for Equality-Constrained Non-linear opt 탐구>

비례식 증거

원래의
등호식
최적화 문제

$$\begin{aligned} & \min_{\vec{x} \in \mathbb{R}^n} f(\vec{x}) \\ \text{s.t. } & \vec{C}(\vec{x}) = \vec{0} \\ & \Downarrow M \geq N. \end{aligned}$$

$$\rightarrow \text{Lagrangian Function : } L(\vec{x}, \vec{\lambda}) = f(\vec{x}) - \vec{\lambda}^T \cdot \vec{C}(\vec{x})$$

$$\rightarrow \text{KKT Condition : } \begin{aligned} \nabla_{\vec{x}}^T L(\vec{x}^*, \vec{\lambda}^*) &= \nabla_{\vec{x}}^T f(\vec{x}^*) - (\nabla \vec{C}(\vec{x}^*))^T \cdot \vec{\lambda}^* = \vec{0} \\ \nabla_{\vec{\lambda}}^T L(\vec{x}^*, \vec{\lambda}^*) &= -\vec{C}(\vec{x}^*) = \vec{0} \iff \nabla_{\vec{\lambda}}^T L(\vec{x}^*, \vec{\lambda}^*) = \vec{C}(\vec{x}^*) = \vec{0} \quad \nabla L = \vec{0} \end{aligned}$$

(증거에 따라 이기 표준 표현식임. 별도로 설명하지 않으면 계산하는 원리만)

$$\rightarrow \text{이 KKT Condition 만족하는 점이 제약최적화 문제의 최적점 } \vec{x}^*, \vec{\lambda}^*!$$

→ 고지약($\nabla L = \vec{0}$)을 한 방에 풀기는 어려워 (\because high nonlinearity), Newton's method로 $\vec{x}^*, \vec{\lambda}^*$ 을 iterative하게 찾아나가자.
(즉, 매 iteration point $\vec{x}_k, \vec{\lambda}_k$ 에서 L 을 2차 함수 \hat{L}_k 로 근사 후 $\nabla \hat{L}_k = \vec{0}$ 인점을 찾아내면서 $\vec{x}^*, \vec{\lambda}^*$ 까지 수렴)

$$\text{Taylor Expansion} \rightarrow L(\vec{x}_k + \vec{p}_{\vec{x}}, \vec{\lambda}_k + \vec{p}_{\vec{\lambda}}) = L(\vec{x}_k, \vec{\lambda}_k) + \left[\begin{matrix} \nabla_{\vec{x}}^T L(\vec{x}_k, \vec{\lambda}_k) \\ \nabla_{\vec{\lambda}}^T L(\vec{x}_k, \vec{\lambda}_k) \end{matrix} \right]^T \left[\begin{matrix} \vec{p}_{\vec{x}} \\ \vec{p}_{\vec{\lambda}} \end{matrix} \right] + \frac{1}{2} \left[\begin{matrix} \vec{p}_{\vec{x}} \\ \vec{p}_{\vec{\lambda}} \end{matrix} \right]^T \left[\begin{matrix} \nabla_{\vec{x}\vec{x}}^2 L(\vec{x}_k, \vec{\lambda}_k) & \nabla_{\vec{x}\vec{\lambda}}^2 L(\vec{x}_k, \vec{\lambda}_k) \\ \nabla_{\vec{\lambda}\vec{x}}^2 L(\vec{x}_k, \vec{\lambda}_k) & \nabla_{\vec{\lambda}\vec{\lambda}}^2 L(\vec{x}_k, \vec{\lambda}_k) \end{matrix} \right] \left[\begin{matrix} \vec{p}_{\vec{x}} \\ \vec{p}_{\vec{\lambda}} \end{matrix} \right] + H.O.T.$$

Newton's method

$$\text{Let } \hat{L}_k(\vec{x}_k + \vec{p}_{\vec{x}}, \vec{\lambda}_k + \vec{p}_{\vec{\lambda}}) = \left[\begin{matrix} \nabla_{\vec{x}}^T L(\vec{x}_k, \vec{\lambda}_k) \\ \nabla_{\vec{\lambda}}^T L(\vec{x}_k, \vec{\lambda}_k) \end{matrix} \right]^T \left[\begin{matrix} \vec{p}_{\vec{x}} \\ \vec{p}_{\vec{\lambda}} \end{matrix} \right] + \frac{1}{2} \left[\begin{matrix} \vec{p}_{\vec{x}} \\ \vec{p}_{\vec{\lambda}} \end{matrix} \right]^T \left[\begin{matrix} \nabla_{\vec{x}\vec{x}}^2 L(\vec{x}_k, \vec{\lambda}_k) & \nabla_{\vec{x}\vec{\lambda}}^2 L(\vec{x}_k, \vec{\lambda}_k) \\ \nabla_{\vec{\lambda}\vec{x}}^2 L(\vec{x}_k, \vec{\lambda}_k) & \nabla_{\vec{\lambda}\vec{\lambda}}^2 L(\vec{x}_k, \vec{\lambda}_k) \end{matrix} \right] \left[\begin{matrix} \vec{p}_{\vec{x}} \\ \vec{p}_{\vec{\lambda}} \end{matrix} \right]$$

$\rightarrow \vec{x}_k, \vec{\lambda}_k$ 에서 Quadratic Function으로 근사된 Lagrangian function
($L(\vec{x}_k, \vec{\lambda}_k)$ 항은 scalar이므로 굳이 반영 안해도 됨)

$$\rightarrow \min_{\vec{p}_{\vec{x}} \in \mathbb{R}^n, \vec{p}_{\vec{\lambda}} \in \mathbb{R}^m} \hat{L}_k(\vec{x}_k + \vec{p}_{\vec{x}}, \vec{\lambda}_k + \vec{p}_{\vec{\lambda}})$$

... 원래 등호식 문제의 KKT 조건 $\nabla L = \vec{0}$ (비례식 증거)을

Iterative Newton's Method로 approach

(즉 2차 함수 비제약 최적화 @ every k-th iteration)

$$\rightarrow \begin{bmatrix} \nabla_{\vec{x}}^T \vec{L} \\ \nabla_{\vec{\lambda}}^T \vec{L} \end{bmatrix} = \begin{bmatrix} \nabla_{\vec{x}}^T L(\vec{x}_k, \vec{\lambda}_k) \\ \nabla_{\vec{\lambda}}^T L(\vec{x}_k, \vec{\lambda}_k) \end{bmatrix} + \begin{bmatrix} \nabla_{\vec{x}\vec{x}}^2 L(\vec{x}_k, \vec{\lambda}_k) & \nabla_{\vec{x}\vec{\lambda}}^2 L(\vec{x}_k, \vec{\lambda}_k) \\ \nabla_{\vec{\lambda}\vec{x}}^2 L(\vec{x}_k, \vec{\lambda}_k) & \nabla_{\vec{\lambda}\vec{\lambda}}^2 L(\vec{x}_k, \vec{\lambda}_k) \end{bmatrix} \begin{bmatrix} \vec{P}_{\vec{x}}^k \\ \vec{P}_{\vec{\lambda}}^k \end{bmatrix} = \begin{bmatrix} \vec{0} \\ \vec{0} \end{bmatrix}$$

...不受约束 Unconstrained Quadratic function을 minimize 시키는 $\vec{P}_{\vec{x}}^k, \vec{P}_{\vec{\lambda}}^k$ 에 대한 eqn (1st Order NC)

$$\rightarrow \begin{array}{c} n \\ m \end{array} \begin{bmatrix} \nabla_{\vec{x}\vec{x}}^2 L(\vec{x}_k, \vec{\lambda}_k) & \nabla_{\vec{x}\vec{\lambda}}^2 L(\vec{x}_k, \vec{\lambda}_k) \\ \nabla_{\vec{\lambda}\vec{x}}^2 L(\vec{x}_k, \vec{\lambda}_k) & \nabla_{\vec{\lambda}\vec{\lambda}}^2 L(\vec{x}_k, \vec{\lambda}_k) \end{bmatrix} \begin{bmatrix} \vec{P}_{\vec{x}}^k \\ \vec{P}_{\vec{\lambda}}^k \end{bmatrix} = - \begin{bmatrix} \nabla_{\vec{x}}^T L(\vec{x}_k, \vec{\lambda}_k) \\ \nabla_{\vec{\lambda}}^T L(\vec{x}_k, \vec{\lambda}_k) \end{bmatrix}$$

KKT matrix

$$\rightarrow \begin{array}{c} n \\ m \end{array} \begin{bmatrix} n \times n & \xi \\ \nabla_{\vec{x}}^2 f(\vec{x}_k) - \lambda_i^k \nabla_{\vec{x}}^2 \hat{C}_{ijk}(\vec{x}_k) & - \left\{ \nabla_{\vec{x}}^T \hat{C}(\vec{x}_k) \right\}^T \\ \nabla_{\vec{x}}^2 \hat{C}(\vec{x}_k) & m \times n \end{bmatrix} \begin{bmatrix} \vec{P}_{\vec{x}}^k \\ \vec{P}_{\vec{\lambda}}^k \end{bmatrix} = - \begin{bmatrix} \nabla_{\vec{x}} f(\vec{x}_k) - \left\{ \nabla_{\vec{x}}^T \hat{C}(\vec{x}_k) \right\}^T \cdot \vec{\lambda}_k \\ \hat{C}(\vec{x}_k) \end{bmatrix}$$

Newton's method

$$\nabla_{\vec{x}} L = 0 \Rightarrow$$

$\hat{C}(\vec{x}) = \vec{0}$ 이거나 $\hat{C}(\vec{x}) = \vec{0}$ 으로 인수에 따라 달라짐.

\therefore 예)의 $\frac{\partial^2 L}{\partial \vec{x}^2} \leq \hat{C}(\vec{x})$ 에 대해 $\nabla_{\vec{x}}$ 를 한 거임.

\rightarrow If we get $\vec{P}_{\vec{x}}^k, \vec{P}_{\vec{\lambda}}^k$ by solving above system eqn, \vec{x}_{k+1} & $\vec{\lambda}_{k+1}$ are to be updated as

$$\rightarrow \begin{bmatrix} \vec{x}_{k+1} \\ \vec{\lambda}_{k+1} \end{bmatrix} = \begin{bmatrix} \vec{x}_k \\ \vec{\lambda}_k \end{bmatrix} + \begin{bmatrix} \vec{P}_{\vec{x}}^k \\ \vec{P}_{\vec{\lambda}}^k \end{bmatrix} \rightarrow \text{1회 } \vec{x}_{k+1}, \vec{\lambda}_{k+1} \text{ QP subproblem 만들고 이것끼리 반복 until converging to global } \vec{x}^*, \vec{\lambda}^*.$$

This is the iterative process using Newton's method for finding $\vec{x}^*, \vec{\lambda}^*$ of equality-constrained Non-linear opt probm.

The Newton Iteration is well defined when the KKT matrix is non-singular.

The matrix is non-singular if $\begin{array}{l} \text{1. The constraint Jacobian } \nabla \hat{C}(\vec{x}) \text{ has full row rank } m. \\ \text{2. The matrix } \nabla_{\vec{x}\vec{x}}^2 L(\vec{x}, \vec{\lambda}) \text{ is Positive-Definite} \end{array}$

(1,1) 성분 of KKT matrix (ξ)

$\because \nabla \hat{C}(\vec{x})$ has full row rank? \rightarrow 각 등式(약관)이 이루는 M개의 균연($\hat{C}_i(\vec{x}) = 0$)이 법선벡터($\nabla \hat{C}_i(\vec{x}) = \vec{0}$)끼리 모두 서로 독립임 의미. 즉 충분다는 저약관이 없음을 의미.

$\therefore \vec{x}_0, \vec{\lambda}_0$ 가 실제 문제의 진짜 optimum point인 $\vec{x}^*, \vec{\lambda}^*$ 에 어느 정도 이상 가깝게 초기화되면 $\nabla_{\vec{x}\vec{x}}^2 L$ 은 항상 P-D이고 equality-constrained non-linear opt Newton's method 기반 SQP를 이용해 성공적으로 풀린다.

Sequential Quadratic Programming (SQP) for Equality Constrained Non-linear Optimization

자, 이제 앞서 알아보았던 Iterative Newton's Method for Equality-Constrained Non-linear Optimization을 동일한 의미(식)를
내포하는 다른 형식의 approach, 즉 Sequential Quadratic Programming (SQP) for Equality-Constrained Non-linear Optimization
로 바꾸어보자.

$$\begin{array}{l} \min_{\vec{x} \in \mathbb{R}^n} f(\vec{x}) \\ \text{등호제약} \\ \text{문제} \\ \text{s.t. } \vec{C}(\vec{x}) = \vec{0} \end{array} \rightarrow \begin{array}{l} \text{Lagrangian Function: } L(\vec{x}, \vec{\lambda}) = f(\vec{x}) - \vec{\lambda}^T \cdot \vec{C}(\vec{x}) \\ \nabla L(\vec{x}^*, \vec{\lambda}^*) = \vec{0} \end{array} \quad \text{비제약 문제}$$

Newton's Method

$$\text{Let } \hat{L}_k(\vec{x}_k + \vec{p}_x^k, \vec{\lambda}_k + \vec{p}_{\lambda}^k) = \begin{bmatrix} \nabla_{\vec{x}}^T L(\vec{x}_k, \vec{\lambda}_k) \\ \nabla_{\vec{\lambda}}^T L(\vec{x}_k, \vec{\lambda}_k) \end{bmatrix}^T \begin{bmatrix} \vec{p}_x^k \\ \vec{p}_{\lambda}^k \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \vec{p}_x^k \\ \vec{p}_{\lambda}^k \end{bmatrix}^T \begin{bmatrix} \nabla_{\vec{x}\vec{x}}^2 L(\vec{x}_k, \vec{\lambda}_k) & \nabla_{\vec{x}\vec{\lambda}}^2 L(\vec{x}_k, \vec{\lambda}_k) \\ \nabla_{\vec{\lambda}\vec{x}}^2 L(\vec{x}_k, \vec{\lambda}_k) & \nabla_{\vec{\lambda}\vec{\lambda}}^2 L(\vec{x}_k, \vec{\lambda}_k) \end{bmatrix} \begin{bmatrix} \vec{p}_x^k \\ \vec{p}_{\lambda}^k \end{bmatrix}$$

$$\min_{\vec{p}_x^k \in \mathbb{R}^n, \vec{p}_{\lambda}^k \in \mathbb{R}^m} \hat{L}_k(\vec{x}_k + \vec{p}_x^k, \vec{\lambda}_k + \vec{p}_{\lambda}^k)$$

(이것은 필요)

... 초기 등호제약 문제의 KKT 조건 $\nabla L = \vec{0}$ (비제약 문제) $\frac{1}{2}$
뉴턴반복법으로 approach
 (즉 고차항수 비제약 최적화 @ every k-th iteration)

$$\rightarrow \begin{bmatrix} \nabla_{\vec{x}\vec{x}}^2 L(\vec{x}_k, \vec{\lambda}_k) & \nabla_{\vec{x}\vec{\lambda}}^2 L(\vec{x}_k, \vec{\lambda}_k) \\ \nabla_{\vec{\lambda}\vec{x}}^2 L(\vec{x}_k, \vec{\lambda}_k) & \nabla_{\vec{\lambda}\vec{\lambda}}^2 L(\vec{x}_k, \vec{\lambda}_k) \end{bmatrix} \begin{bmatrix} \vec{p}_x^k \\ \vec{p}_{\lambda}^k \end{bmatrix} = - \begin{bmatrix} \nabla_{\vec{x}}^T L(\vec{x}_k, \vec{\lambda}_k) \\ \nabla_{\vec{\lambda}}^T L(\vec{x}_k, \vec{\lambda}_k) \end{bmatrix} \quad \text{... 1st Order NC for Optimality}$$

$$\Rightarrow \nabla \hat{L}_k = \vec{0}$$

$$\rightarrow \begin{bmatrix} \nabla_{\vec{x}}^2 f(\vec{x}_k) - \lambda_i^k \nabla_{\vec{x}}^2 \hat{C}_{ijk}(\vec{x}_k) - \{\nabla_{\vec{x}}^T \vec{C}(\vec{x}_k)\}^T \\ \nabla_{\vec{x}}^T \vec{C}(\vec{x}_k) \end{bmatrix} \begin{bmatrix} \vec{p}_x^k \\ \vec{p}_{\lambda}^k \end{bmatrix} = - \begin{bmatrix} \nabla_{\vec{x}} f(\vec{x}_k) - \{\nabla_{\vec{x}}^T \vec{C}(\vec{x}_k)\}^T \cdot \vec{\lambda}_k \\ \vec{C}(\vec{x}_k) \end{bmatrix}$$

여기까지가 앞에서 알아보았던 방식

: 원래의 제약 문제를 비제약 문제(목적함수가 고차항수로 변환 후 Newton's Method로 조작)로 바꾸기.
 이걸이 Newton's Method가 푸는
 고차항수 비제약 최적화 문제를 Quadratic Programming (QP)
 문제 (즉, $\min Q$ 문제)로 바꿔버림
 s.t. Linear function

등호제약 문제 \rightarrow 비제약 문제 \rightarrow 뉴턴반복법

$$\left\{ \nabla_{\vec{x}}^2 f(\vec{x}_k) - \lambda_i^k \nabla_{\vec{x}}^2 \hat{C}_{ijk}(\vec{x}_k) \right\} \vec{p}_x^k - \{\nabla_{\vec{x}}^T \vec{C}(\vec{x}_k)\}^T \vec{p}_{\lambda}^k = -\nabla_{\vec{x}} f(\vec{x}_k) + \{\nabla_{\vec{x}}^T \vec{C}(\vec{x}_k)\}^T \vec{\lambda}_k$$

$$\rightarrow \left\{ \nabla_{\vec{x}}^2 f(\vec{x}_k) - \lambda_i^k \nabla_{\vec{x}}^2 \hat{C}_{ijk}(\vec{x}_k) \right\} \vec{p}_x^k - \{\nabla_{\vec{x}}^T \vec{C}(\vec{x}_k)\}^T (\vec{\lambda}_k + \vec{p}_{\lambda}^k) = -\nabla_{\vec{x}} f(\vec{x}_k)$$

$\hookrightarrow \text{Let } \vec{P}_k \quad \hookrightarrow \text{Let } \vec{\lambda}_k$

$$\rightarrow \left\{ \nabla_{\vec{x}}^2 f(\vec{x}_k) - \lambda_i^k \nabla_{\vec{x}}^2 \hat{C}_{ijk}(\vec{x}_k) \right\} \vec{p}_x^k + \nabla_{\vec{x}} f(\vec{x}_k) - \{\nabla_{\vec{x}}^T \vec{C}(\vec{x}_k)\}^T \vec{\lambda}_k = \vec{0}$$

$$\{\nabla_{\vec{x}}^T \vec{C}(\vec{x}_k)\}^T (\vec{P}_k + \vec{C}(\vec{x}_k)) = \vec{0}$$

\hookrightarrow 이걸 QP 문제의 제약 조건 만족시켜 $\nabla_{\vec{x}}^T \vec{P}_k = \vec{0}$ 조건이라 생각해보자.
 \vec{P}_k 가 선제변수고 $\vec{\lambda}_k$ 가 Lagrange 변수!

\hookrightarrow 이걸 QP 문제의 제약 조건 만족시켜 $\nabla_{\vec{x}}^T \vec{P}_k = \vec{0}$ 이걸 사용해보자.

$\hookrightarrow \vec{P}_k$ 가 선제변수고 $\vec{\lambda}_k$ 가 Lagrange 변수!

* QP가 뉴턴법과는 8-1 내용 참고

↓ 실제 변수 \vec{P}_k 에 대해 좌변 적분 = 어떤 QP 문제의 Lagrangian 합수

$$\hat{\mathcal{L}}_k^{QP} = \frac{1}{2} \vec{P}_k^T \left(\nabla_{\vec{x}}^2 f(\vec{x}_k) - \lambda_i^k \nabla_{\vec{x}}^2 \hat{C}_{ijk}(\vec{x}_k) \right) \vec{P}_k + (\nabla_{\vec{x}} f(\vec{x}_k))^T \vec{P}_k - \vec{\lambda}_k^T \nabla_{\vec{x}} \hat{C}(\vec{x}_k) \cdot \vec{P}_k + (\vec{P}_k \text{와 무관한 적분 상수항})$$

Lagrange 승수 $\vec{\lambda}_k$ 에 대해 좌변 적분 = 어떤 QP 문제의 Lagrangian 합수

$$\hat{\mathcal{L}}_k^{QP} = - \vec{\lambda}_k^T \left((\nabla_{\vec{x}} \hat{C}(\vec{x}_k))^T \cdot \vec{P}_k + \hat{C}(\vec{x}_k) \right) + (\vec{\lambda}_k \text{와 무관한 적분상수항}) \quad \text{← 어떤 QP 문제의 Lagrangian 합수로 등장}$$

$$\therefore \hat{\mathcal{L}}_k^{QP} = \frac{1}{2} \vec{P}_k^T \left(\nabla_{\vec{x}}^2 f(\vec{x}_k) - \lambda_i^k \nabla_{\vec{x}}^2 \hat{C}_{ijk}(\vec{x}_k) \right) \vec{P}_k + (\nabla_{\vec{x}} f(\vec{x}_k))^T \vec{P}_k - \vec{\lambda}_k^T \left((\nabla_{\vec{x}} \hat{C}(\vec{x}_k))^T \cdot \vec{P}_k + \hat{C}(\vec{x}_k) \right)$$

이 Lagrangian Functional
대응되는 QP
이 QP 문제에 대응되는
Lagrangian Function

QP Subproblem

$$\underset{\vec{P}_k \in \mathbb{R}^n}{\text{Min}} \quad \frac{1}{2} \vec{P}_k^T \left(\nabla_{\vec{x}}^2 f(\vec{x}_k) - \lambda_i^k \nabla_{\vec{x}}^2 \hat{C}_{ijk}(\vec{x}_k) \right) \vec{P}_k + (\nabla_{\vec{x}} f(\vec{x}_k))^T \cdot \vec{P}_k$$

↳ 3차 Tensor

$$\text{s.t. } (\nabla_{\vec{x}} \hat{C}(\vec{x}_k))^T \cdot \vec{P}_k + \hat{C}(\vec{x}_k) = \vec{0} \quad \dots \text{Linear form}$$

$\vec{\lambda}_k = \vec{\lambda}_k + \vec{P}_k^k = \vec{\lambda}_{k+1}$
 $\therefore \vec{\lambda}_k = \vec{\lambda}_{k+1}$ from 뉴턴반복법
 i.e. QP 문제의 Lagrange 승수
 = 원래의 최적화 문제에 따른 Newton's method에서 찾고자 하는 그 다음
 iteration에서의 Lagrange st.

→ 원래의 동적 최적화 문제의 KKT 조건 $\nabla L = \vec{0}$ (비지역 문제)에 대한

Iterative Newton's Method의 변형에 대한

Sequential Quadratic Programming (SQP)으로의 변형

뉴턴반복법 → SQP

최종적으로 변형된 QP를 매 iteration에서 풀어서 (Newton's method로 푸는 식과 동일) \vec{P}_k , $\vec{\lambda}_k$ 를 구하면, 그 다음 design variables \vec{x}_{k+1} 및 Lagrange 승수 $\vec{\lambda}_{k+1}$ 는 다음과 같다.
 ↳ 비지역 문제 (뉴턴반복법) 관점에서 보면!

$$\begin{matrix} n \\ m \end{matrix} \begin{bmatrix} \nabla_{\vec{x}}^2 f(\vec{x}_k) - \lambda_i^k \nabla_{\vec{x}}^2 \hat{C}_{ijk}(\vec{x}_k) - \{\nabla_{\vec{x}} \hat{C}(\vec{x}_k)\}^T \\ \nabla_{\vec{x}} \hat{C}(\vec{x}_k) \end{bmatrix} \begin{bmatrix} \vec{P}_k \\ \vec{\lambda}_k \end{bmatrix} = - \begin{bmatrix} \nabla_{\vec{x}} f(\vec{x}_k) \\ \hat{C}(\vec{x}_k) \end{bmatrix} \rightarrow \begin{array}{l} \text{이걸 한 방에 풀어서} \\ \begin{bmatrix} \vec{P}_k \\ \vec{\lambda}_k \end{bmatrix} \text{를 동시에 해야 함.} \end{array}$$

High Cost!!

$$\rightarrow \vec{x}_{k+1} = \vec{x}_k + \vec{P}_k$$

↳ 상기 QP subproblem의 KKT식.
 ↳ §-1 (QP에 대한 KKT식) 참고

$$\rightarrow \vec{\lambda}_{k+1} = \vec{\lambda}_k$$

동적 최적화 → 비지역 문제 → 뉴턴반복법 → SQP

어차피 Newton's method(unconstrained 형식)로 푸는 거나 QP method(constrained 형식)로 푸는 거나 결국엔 Newton's method 쓰는 거랑 동일한 건데 굳이 왜 QP method를 기준으로 표현해서 x_{k+1} , λ_{k+1} 을 구하는 건지 모르겠네.. 뭐 SQP에 대해 더 배우다보면 왜 굳이 이렇게 표현해서 푸는 지 알게 되겠지. 일단 지금은 굳이..?라는 생각임.
 어쨌든 기술적인 프로세스 자체는 이해가 감 ○○

$$\boxed{\text{線性}} \quad \hat{C}_1(\vec{n}) = \vec{0} \quad f(\vec{n})$$

\vec{x}^* coupled with $\vec{\lambda}^*$ which makes

$$\nabla f(\vec{x}^*) = \sum_i \lambda_i^* \nabla \hat{C}_i(\vec{x}^*)$$

Diagram illustrating the iterative process of the Sequential Quadratic Programming (SQP) algorithm. It shows a sequence of quadratic approximations $L_k(\vec{x}, \lambda)$ centered at \vec{x}_k^*, λ_k^* . The diagram highlights the 'Subproblem' (QP Subproblem) at each step, which involves solving a quadratic function coupled with linear constraints. A red circle labeled 'Subproblem' encloses the current quadratic approximation and its associated linear terms. Another red circle labeled 'QP Subproblem' encloses the specific QP subproblem equation. The overall process is labeled '수렴 과정' (Convergence Process).

\langle SQP (or Iterative Newton's Method) for Equality Constrained Non-linear Opt \rangle

→ SQP Algorithm for Equality Constrained Optimization.

Algorithm 18.1 (Local SQP Algorithm).

Choose an initial pair (x_0, λ_0) ; → 초기값 초기화 → 초기 계산이 cost에 영향

for $k = 0, 1, 2, \dots$

Evaluate $f_k, \nabla f_k, W_k = W(x_k, \lambda_k), c_k$, and A_k ;

Solve (18.8) to obtain p_k and λ_k ;

$x_{k+1} \leftarrow x_k + p_k; \lambda_{k+1} \leftarrow \lambda_k$ → QP Subproblem

if convergence test satisfied

STOP with approximate solution (x_{k+1}, λ_{k+1}) ;

end (for).

$$n \begin{bmatrix} \nabla_{\vec{x}}^2 f(\vec{z}_k) - \lambda_i^k \nabla_{\vec{x}}^2 \hat{C}_{ijk}(\vec{z}_k) \\ \vec{w}_k \end{bmatrix} - \underbrace{\begin{bmatrix} \nabla_{\vec{x}} \hat{C}(\vec{z}_k) \end{bmatrix}^T}_{\sim A_k} \begin{bmatrix} \vec{p}_k \\ \vec{l}_k \end{bmatrix} = - \begin{bmatrix} \nabla_{\vec{x}} f(\vec{z}_k) \\ \vec{r}_k \end{bmatrix} \rightarrow \text{이걸 한 끝에 끌어가} \\ m \begin{bmatrix} \nabla_{\vec{x}} \hat{C}(\vec{z}_k) \\ \sim A_k \end{bmatrix} \begin{bmatrix} \vec{p}_k \\ \vec{l}_k \end{bmatrix} = \begin{bmatrix} \vec{C}(\vec{z}_k) \\ C_k \end{bmatrix} \rightarrow \text{ } \begin{bmatrix} \vec{p}_k \\ \vec{l}_k \end{bmatrix} \text{를 동시에 해야함.}$$

8-3-2) General Constrained SQP (with Equality + Inequality Constraints)

Inequality Constrained 가 추가된 general Constrained optimization의 경우, QP Subproblem이 다음과
 $\frac{g_L^T}{\lambda} + \frac{1}{2} x^T P x$ 구성된다. (8-3-1 라우지) → 단/불등式 등호 - $\vec{x} = \vec{\lambda} \cup \vec{v}$ / $\vec{\tilde{x}} = \vec{\tilde{\lambda}} \cup \vec{\tilde{v}}$

$$\begin{aligned}
 & \min_{\vec{x} \in \mathbb{R}^n} f(\vec{x}) \\
 & \text{s.t. } \widehat{C}_i(\vec{x}) = 0 ; i=1 \sim m \\
 & \quad C_i(\vec{x}) \geq 0 ; i=1 \sim p
 \end{aligned}
 \quad \xrightarrow{\quad \vec{P}_k \in \mathbb{R}^n \quad} \quad
 \begin{aligned}
 & \min_{\vec{P}_k \in \mathbb{R}^n} \frac{1}{2} \vec{P}_k^T \left(\nabla_{\vec{x}}^2 f(\vec{x}_k) - \sum_{i=1}^m \nabla_{\vec{x}}^2 \widehat{C}_{i,j}(\vec{x}_k) \right) \vec{P}_k + \left(\nabla_{\vec{x}} f(\vec{x}_k) \right)^T \vec{P}_k \\
 & \quad \text{S.t. } (\nabla_{\vec{x}} \widehat{C}_i(\vec{x}_k))^T \vec{P}_k + \widehat{C}_i(\vec{x}_k) = 0 ; i=1 \sim m \\
 & \quad \quad \quad \text{S.t. } (\nabla_{\vec{x}} C_i(\vec{x}_k))^T \cdot \vec{P}_k + C_i(\vec{x}_k) \geq 0 ; i=1 \sim p
 \end{aligned}$$

QP Subproblem at every k-th outer iteration

• QP Subproblem의 Inequality Constraint 및 결국 Equality Constraint와 같은 방식으로 표현됨을 볼 수 있다.
 그래서 Constrained Optimal 문제는 KKT 조건 중 하나인 $\nabla_x L = 0$ 이 두 Constraints에 대한 eqn을 똑같은 방식으로 나타내기 때문이다. 뿐만 아니라 추후에 Active-set method를 통해서 Active(위반)에 가까운 부등호제약들을 마치 등호제약처럼 다룰 때에 같은 form으로 나타내지는 것이라 보도 흔들 것이다. 특히 부등호제약이 왜 그렇게 선형화되는 표현되는지 엄밀하게 유도해보고 싶으면, 원래의 부등호제약이 slack variables 추가하여 등호제약처럼 간주한 뒤 일정 배수인 과정(8-3-1에서의)을 거치고 slack variables를 최종적으로 삭제해버리면 볼 수 있을 듯하다. 근데 너무 깊고 소모적인 거 같으니 생략. 어쨌든 자, 그럼 이제 이 QP Subproblem에서 \vec{p}_k, \vec{l}_k 를 어떻게 구하는지 알아보자.

8-3-2-1) SQP for general Constrained optimization 알고리즘 유도를 위한 준비

일단 본격적으로 시작하기에 앞서서 서두를 깔아보자.

1) 부등호제약조건 SQP를 등호제약조건 SQP처럼 다루기(Active Set Method)

일단 우리의 최종목표는 부등호 제약까지 고려한 general constrained optimization을 SQP로 푸는 거다. 부등호까지 고려한 SQP를 수행하는 방식은 여러가지(Active-Set Method(ASM), Interior Point Method(IPM), Augmented Lagrangian Method(ALM), Gradient Projection Method(GPM), ...)가 있지만 여기서는 앞서 한 번 이미 배운 ALM을 통해 SQP를 수행할 것이다. 참고로 ASM은 결과적으로는 부등호제약조건을 마치 등호제약조건처럼 간주하여 SQP를 진행하는 것이다. 따라서 Active-Set Method의 Framework 안에서 SQP를 수행할 거라면 최종적으로는 원래의 주어진 부등호제약문제에 대한 QP subproblem을 마치 등호제약문제에 대한 QP subproblem을 다루는 것으로 이해해도 좋을 것이다. 시바 사실 ASM으로 SQP 구현할랬는데 난이도가 너무 높다 생각보다.. 지금까지 배운 것중에 제일 어려운 듯;; 그래도 나중에 시간 나면 한 번 구현해보자. Nocedal 책 16.3~16.5절까지의 내용 참고하면 된다.

2) Quasi-Newton Methods를 통해 ALM의 Augmented KKT Systems를 PD화하여 풀기

따라서 우리가 반복적으로 풀고자 하는 등호제약문제에 대한 QP는 ALM을 통해 비제약 문제로 바꿀 수 있는데, ALM의 OC로부터 나오는 Augmented KKT Systems를 풀으로써 step을 얻고 그 다음 iteration point로 갈 수가 있다. 이 때 KKT Systems의 (1,1) 성분에 위치한 행렬이 PD(Convex)가 아닐 수도 있기 때문에(이는 search direction을 descent 방향이 아니게 만들 수 있고 따라서 global convergence를 방해하고 divergence를 가능하게 함) 이를 handling하기 위한 approach가 다음과 같이 크게 2가지가 있다.

A) Line search SQP methods : (1, 1) 성분 행렬에 어떤 조작(Quasi-Newton approximation / Hessian of augmented Lagrangian function / 기타 등등)을 가해서 무조건 PD(convex)로 만들어 search direction pk를 descent direction으로 보장하는 것.

B) Trust region SQP methods : QP subproblem에 별도의 constraint를 추가해서 search direction pk를 “신뢰 가능한 영역” 내로 제한

여기서는 A) Line Search SQP Methods를 택해서 배울 거고 특히 앞서 배웠던 Quasi-Newton approximation을 써서 QP 부문제의 의 W_k 를 PD화하는 방식을 택할 거다(그리고 Trust region SQP methods는 공수가 많이 들어서 별로임).

3) 빠른 QP 문제 solving을 위한 "Hot Start" 전략 / infeasible QP 문제 생성에 대한 대처법

- $k+1$ -th QP를 빠르게 풀기 위해 k -th QP에서 찾아낸 p^* , k 를 $p(0)$, $k+1$ 로 활용(이거 왜 적용 안했냐. 코드 수정 그거).
- 그리고 가끔 Infeasible QP 문제가 생성되기도 하는데, 이를 해결하기 위해 SNOPT와 같은 알고리즘에서는 infeasible QP와 유사한 auxiliary QP subproblem을 세워서 이걸 풀으로써 feasible solution을 얻기도 함. 근데 그냥 초반에 시도하는 x_0 를 feasible point로 잘 잡으면 infeasible QP가 웬만하면 안 생길 거임. 그럼에도 생기면 저런 method를 써야겠지.

이렇게 “ALM(for handling general constraints) + Quasi-Newton approximation(for stabilizing W_k of Qk)” 조합의 line search approach로 search direction pk를 찾을 거고 그 다음 step length는 merit function이라고 하는 것을 이용해서 찾을 거다. 상세한 과정은 쭉 이어지는 과정들을 통해 확인할 수 있을 것이다. 자 그럼 시작해보자.

8-3-2-2) SQP with ALM ??.

Original Constrained Opt probm - - - - -

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

$$\text{s.t. } \hat{C}_i(\vec{x}) = 0 ; i=1 \sim m$$

$$C_i(\vec{x}) \geq 0 ; i=1 \sim p$$

$$\vec{x}_0, \vec{\lambda}_0$$

\vec{x}^* ... 최종 optimum point

Sequential Quadratic Programming (SQP)

$k=0 \sim \text{Converge}$

$$\min_{\vec{P}_k \in \mathbb{R}^n} Q_k(\vec{P}_k) = \frac{1}{2} \vec{P}_k^T \left(\nabla_{\vec{x}}^2 f(\vec{x}_k) - \sum_i^k \nabla_{\vec{x}}^2 \tilde{C}_{ij,k}(\vec{x}_k) \right) \vec{P}_k + (\nabla_{\vec{x}} f(\vec{x}_k))^T \vec{P}_k$$

부등호 제약/등호 제약 다 포함.
 $\tilde{C}_{ij,k}$: BFGS2 계산 & P.D.E.P.

$$\text{s.t. } (\nabla_{\vec{x}} \hat{C}_i(\vec{x}_k))^T \vec{P}_k + \hat{C}_i(\vec{x}_k) = 0 ; i=1 \sim m$$

$$\tilde{C}_i(\vec{P}_k) (\nabla_{\vec{x}} C_i(\vec{x}_k))^T \vec{P}_k + C_i(\vec{x}_k) \geq 0 ; i=1 \sim p$$

Outer loop

iStep length
based on
(Merit Function)

Augmented Lagrangian
Method (ALM)

$$= \vec{P}_{k-1}^* \quad (\text{Hot start 가능})$$

$\vec{P}_{k0}, Q_k,$
 $\tilde{C}_i, \vec{C}_i, \vec{\lambda}_k, \vec{v}_k$ for $\vec{\lambda}_{kj}^{(0)}, \vec{v}_{kj}^{(0)}$ (as initial guess)

$$\min_{\vec{P}_{kj} \in \mathbb{R}^n} L_{A,kj}(\vec{P}_{kj}) = \frac{1}{2} \vec{P}_{kj}^T \left(\nabla_{\vec{x}}^2 f(\vec{x}_k) - \sum_i^k \nabla_{\vec{x}}^2 \tilde{C}_{ij,k}(\vec{x}_k) \right) \vec{P}_{kj} + (\nabla f(\vec{x}_k))^T \vec{P}_{kj}$$

$j=0 \sim \text{Converge}$

Intermediate
loop

$$- \sum_i^m \vec{\lambda}_{kji} \left\{ (\nabla \hat{C}_i(\vec{x}_k))^T \vec{P}_{kj} + \hat{C}_i(\vec{x}_k) \right\} + \frac{M_{kj}}{2} \sum_i^m \left\{ (\nabla \hat{C}_i(\vec{x}_k))^T \vec{P}_{kj} + \hat{C}_i(\vec{x}_k) \right\}^2$$

$$- \sum_i^p \vec{v}_{kji} \left\{ (\nabla C_i(\vec{x}_k))^T \vec{P}_{kj} + C_i(\vec{x}_k) - \max \left((\nabla C_i(\vec{x}_k))^T \vec{P}_{kj} + C_i(\vec{x}_k) - \frac{\vec{v}_{kji}}{p_{kj}}, 0 \right) \right\}$$

$$+ \frac{p_{kj}}{2} \sum_i^p \left\{ (\nabla C_i(\vec{x}_k))^T \vec{P}_{kj} + C_i(\vec{x}_k) - \max \left((\nabla C_i(\vec{x}_k))^T \vec{P}_{kj} + C_i(\vec{x}_k) - \frac{\vec{v}_{kji}}{p_{kj}}, 0 \right) \right\}^2$$

Unconstrained Opt



SQP / CGM / QNM + Step length algorithm

Inner loop

$\vec{P}_{kj}^*, \vec{\lambda}_{kj}^*, \vec{v}_{kj}^*$

8-3-2-3) Damped - BFGS $\nabla_{\bar{x}} L_k = \tilde{W}_k$ for SQP

$$\min_{\tilde{P}_k \in \mathbb{R}^n} Q_k(\tilde{P}_k) = \frac{1}{2} \tilde{P}_k^T \left(\nabla_{\bar{x}}^2 f(\bar{x}_k) - \sum_i^K \lambda_i \nabla_{\bar{x}}^2 \tilde{C}_i(\bar{x}_k) \right) \tilde{P}_k + (\nabla_{\bar{x}} f(\bar{x}_k))^T \tilde{P}_k$$

부등호의 역/등호의 역 다 포함.
[W_k]

$$\text{s.t. } \begin{aligned} & (\nabla_{\bar{x}} \tilde{C}_i(\bar{x}_k))^T \tilde{P}_k + \tilde{C}_i(\bar{x}_k) = \vec{0} ; i=1 \sim m \\ & \tilde{C}_i(\bar{x}_k) (\nabla_{\bar{x}} C_i(\bar{x}_k))^T \cdot \tilde{P}_k + C_i(\bar{x}_k) \geq \vec{0} ; i=1 \sim p \end{aligned}$$

↓ 흡약해석 표현하면

$$\min_{\tilde{P}_k \in \mathbb{R}^n} Q_k(\tilde{P}_k) = \frac{1}{2} \tilde{P}_k^T [W_k](\bar{x}_k, \bar{\lambda}_k, \bar{v}_k) \tilde{P}_k + (\nabla_{\bar{x}} f(\bar{x}_k))^T \tilde{P}_k$$

$$\text{s.t. } \tilde{C}_i(\tilde{P}_k) = \vec{0} ; i=1 \sim m$$

$$\tilde{C}_i(\tilde{P}_k) = \vec{0} ; i=1 \sim p$$

→ where $[W_k] = (\nabla_{\bar{x}}^2 L(\bar{x}_k)) = (\nabla_{\bar{x}\bar{x}}^2 f(\bar{x}_k)) - \sum_i \lambda_i (\nabla_{\bar{x}\bar{x}}^2 \tilde{C}_i(\bar{x}_k))_{ij,k} - \sum_j v_i (\nabla_{\bar{x}\bar{x}}^2 C_i(\bar{x}_k))_{ij,k}$

↳ $L(\bar{x}) = f(\bar{x}) - \sum_i \lambda_i \tilde{C}_i(\bar{x}) - \sum_j v_i C_i(\bar{x})$... Lagrangian function

→ 이거 싱으로 구하려면 너무 비쌈 (Hessian 포함). BFGS 기법으로 비교적 비교적 코리에이션 문제

사기 구하기 (gradient(∇L) 만 이용하기!)

unmodified

Original BFGS method again from Quasi-Newton's Method for Unconstrained Optimization

$$\frac{B_{k+1}}{SS} = \frac{B_k}{SS} + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \frac{B_k \mathbf{s}_k \mathbf{s}_k^T B_k^T}{\mathbf{s}_k^T B_k \mathbf{s}_k} \quad \text{where } \mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$$

$\frac{H(\bar{x}_{k+1})}{||} \quad \frac{H(\bar{x}_k)}{||}$
 $\nabla_{\bar{x}\bar{x}}^2 f(\bar{x}_{k+1}) \quad \nabla_{\bar{x}\bar{x}}^2 f(\bar{x}_k)$

$\frac{\tilde{s}_k^T \tilde{y}_k > 0}{\text{이거 거짓일 때는}} \quad \leftarrow \begin{array}{l} \text{항상 PositiveDefinite 가ass하고} \\ \text{여기 BFGS 쓸까?} \end{array}$

$$\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$$

Unconstrained Optimization

$$\frac{\tilde{B}_{k+1}}{SS} = \frac{\tilde{B}_k}{SS} - \frac{\tilde{B}_k \tilde{s}_k \tilde{s}_k^T \tilde{B}_k^T}{\tilde{s}_k^T \tilde{B}_k \tilde{s}_k} + \frac{\tilde{y}_k \tilde{y}_k^T}{\tilde{y}_k^T \tilde{s}_k} \quad \text{where } \begin{cases} \tilde{s}_k = \bar{x}_{k+1} - \bar{x}_k \\ \tilde{y}_k = \nabla_{\bar{x}} L(\bar{x}_{k+1}) - \nabla_{\bar{x}} L(\bar{x}_k) \end{cases}$$

$\nabla_{\bar{x}\bar{x}}^2 L(\bar{x}_{k+1}) \quad \nabla_{\bar{x}\bar{x}}^2 L(\bar{x}_k)$

Constrained Optimization | 우리가 관심 있는 경우.

③ 그림 원래
BFGS 기법
그냥 쓰는법
한국 사용 가능
보기. 그럼
자주 아래처럼
 $\nabla^2 L_k$ 을
구하기 된다.
이때 잠자리
을지가 있나?
(• 히네우리)

Let's say that we use this original unmodified form of BFGS method in Constrained opt

이때 물리적으로 optimal point \bar{x}^* 과 방향에서 $\nabla_{\bar{x}\bar{x}} L$ 가 positive definite 면 위의 BFGS로 사용할

B_k 의 Convex한 Curvature를 반영해서 \bar{x}^* 로 빠르게 수렴을 할거다. 그러나 만약 $\nabla_{\bar{x}\bar{x}} L$ 중 하나라도 음수가 있다

Negative-definite이나 indefinite라면 (이는 $\tilde{s}_k^T \tilde{y}_k < 0$ 을 뜻한다) B_k 가 P.D가 되지 않을 수도 있기 때문에 하강 방향이

이제가 상승 방향으로 Search direction을 내용을 수도 있고 이로 인해 \bar{x}^* 로 수렴 못하고 발산할 수도 있거나 된다.

$f(\vec{x})$... convex (It makes sense to assume it)

 $\vec{x}^* \rightarrow \nabla_{\vec{x}\vec{x}}^2 f(\vec{x}^*) \succ 0$ (convex)
 \downarrow
 $\vec{s}_k^T \vec{y}_k > 0$ (always if convex every \vec{x}_k)
 \downarrow
 $\nabla_{\vec{x}\vec{x}}^2 f(\vec{x}_k) \approx \tilde{B}_k \succ 0$ (always if convex every \vec{x}_k)

Unconstrained Opt

→ BFGS always works well

$L(\vec{x})$... Not ensuring convex \rightarrow because of existence of constraints within $L(\vec{x})$

 $\vec{x}^* \rightarrow \nabla_{\vec{x}\vec{x}}^2 L(\vec{x}^*) \succ 0 \rightarrow \vec{s}_k^T \vec{y}_k > 0$ (It could be) (It could be)
 \downarrow
 $\nabla_{\vec{x}\vec{x}}^2 L(\vec{x}_k) \approx \tilde{B}_k \succ 0$ (It could be)

Constrained Opt

→ BFGS does not likely work well.

→ (1) Constrained opt와 BFGS를 소리고 다음과 같은 조건을 만날 수도 있음. ↗ 첫번째 경우 (별로)

If $\vec{s}_k^T \vec{y}_k \geq \theta \vec{s}_k^T \tilde{B}_k \vec{s}_k$ → BFGS updating $\rightarrow \tilde{B}_{k+1} = \tilde{B}_k - \frac{\tilde{B}_k \vec{s}_k \vec{s}_k^T \tilde{B}_k^T}{\vec{s}_k^T \tilde{B}_k \vec{s}_k} + \frac{\vec{y}_k \vec{y}_k^T}{\vec{y}_k^T \vec{s}_k}$

↓ 실제 경사 = 예측 경사보다 클 때 (Convex 가능성이 대)

where $\theta = 0.0$ (or 0.00 or 0.000 / ...)

④

If $\vec{s}_k^T \vec{y}_k < \theta \vec{s}_k^T \tilde{B}_k \vec{s}_k$ → skip BFGS updating $\rightarrow \tilde{B}_{k+1} = \tilde{B}_k$

↓ 실제 경사 = 예측 경사보다 작을 때 (Convex 가능성이 소)

Update or Skip Scheme

②) 이건 intuitive/heuristic 한 방법이니 항상 잘 작동하지는 않음. 따라서 3번이 advanced한 BFGS updating은 더 좋지 않음.

⑤ ↗ 두 번째 경우 (better) → Core idea: transform \vec{y}_k into (always) positive \vec{r}_k

Damped BFGS updating for SQP

$$\tilde{B}_{k+1} = \tilde{B}_k - \frac{\tilde{B}_k \vec{s}_k \vec{s}_k^T \tilde{B}_k^T}{\vec{s}_k^T \tilde{B}_k \vec{s}_k} + \frac{\vec{r}_k \vec{r}_k^T}{\vec{s}_k^T \vec{r}_k}$$

where

$\vec{s}_k^T \tilde{B}_k \vec{s}_k$: Approximate curvature b/w \vec{x}_k, \vec{x}_{k+1}

$\tilde{B}_k \vec{s}_k$: Approximate gradient difference b/w \vec{x}_k, \vec{x}_{k+1}

modified \vec{y}_k $\vec{s}_k^T \vec{y}_k$: Real curvature b/w \vec{x}_k, \vec{x}_{k+1}

\vec{r}_k : Real gradient difference b/w \vec{x}_k, \vec{x}_{k+1}

$\vec{s}_k = \vec{x}_{k+1} - \vec{x}_k$... step b/w \vec{x}_k, \vec{x}_{k+1}

$$\vec{y}_k = \nabla_{\vec{x}} L(\vec{x}_{k+1}, \vec{\lambda}_{k+1}) - \nabla_{\vec{x}} L(\vec{x}_k, \vec{\lambda}_k) \dots$$

$$\theta_k = \begin{cases} 1 & \text{if } \vec{s}_k^T \vec{y}_k \geq 0.2 \vec{s}_k^T \tilde{B}_k \vec{s}_k \\ 0.8 \vec{s}_k^T \tilde{B}_k \vec{s}_k & \end{cases}$$

$$0.8 \leq \frac{\vec{s}_k^T \tilde{B}_k \vec{s}_k - \vec{s}_k^T \vec{y}_k}{\vec{s}_k^T \tilde{B}_k \vec{s}_k} < 1$$

$$\frac{0.8}{1 - \frac{\vec{s}_k^T \vec{y}_k}{\vec{s}_k^T \tilde{B}_k \vec{s}_k}} < 1$$

$$\frac{\vec{s}_k^T \vec{y}_k}{\vec{s}_k^T \tilde{B}_k \vec{s}_k} < 0.2$$

$$\alpha = 0.2$$

$$\beta = 0.8$$

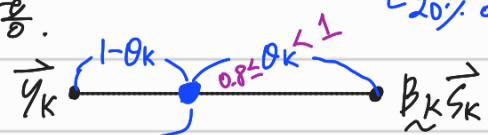
α, β 조정하면 좋으면

$\alpha + \beta = 1$ 만족하도록 조정해야 함

$$\vec{r}_k = \theta_k \vec{y}_k + (1 - \theta_k) \underline{B}_k \vec{s}_k$$

$\Rightarrow (\theta_k = 1) \rightarrow \vec{y}_k$: 실수로 끝이 극단을 보다 보면 실수로 끝 \vec{y}_k 를 측정해 Convex한 \underline{B}_{k+1} 을 만들 수 있을 거라고 판단하여 그동 \vec{y}_k 를 쓴다.

$(\theta_k \neq 1) \rightarrow \vec{r}_k$: 실수로 끝이 극단을 보다 작으면 실수로 끝 \vec{y}_k 가 충분히 Convex하지 않다고 보고, \vec{y}_k 에다가 극단(P-D여야 Convex인) $\underline{B}_k \vec{s}_k$ 를 실수로 만드는 애를 \underline{B}_{k+1} update에 사용.



Unlike the unconstrained opt, there is no strong assumptions or ensurance that \vec{y}_k (gradient difference b/w \vec{r}_k , \vec{r}_{k+1}) is always positive wr.t. step \vec{s}_k in Constrained opt. Therefore instead of simply using \vec{y}_k , we use \vec{r}_k to make a \underline{B}_{k+1} , which is artificially modified version of \vec{y}_k to have a partly positive property of $\underline{B}_k \vec{s}_k$. just in case that \vec{y}_k is judged not to be positive!

We can ensure P-D of \underline{B}_{k+1} by adopting above method (Damped BFGS)

↓
proof)

- If $\theta_k = 0$, It trivially holds $\therefore \vec{r}_k = \underline{B}_k \vec{s}_k \rightarrow \underline{B}_{k+1} = \underline{B}_k \rightarrow \underline{B}_{k+1}$ becomes P-D
- If $0 < \theta_k < 1$,

$$\vec{r}_k = \left(\frac{0.8 \vec{s}_k^T \underline{B}_k \vec{s}_k}{\vec{s}_k^T \underline{B}_k \vec{s}_k - \vec{s}_k^T \vec{y}_k} \right) \vec{y}_k + \left(1 - \frac{0.8 \vec{s}_k^T \underline{B}_k \vec{s}_k}{\vec{s}_k^T \underline{B}_k \vec{s}_k - \vec{s}_k^T \vec{y}_k} \right) \underline{B}_k \vec{s}_k \quad (\underline{B}_0 \text{ is chosen } \vec{I}, \text{ so it holds true})$$

$$\vec{s}_k^T \vec{r}_k = \left(\frac{0.8 \vec{s}_k^T \underline{B}_k \vec{s}_k}{\vec{s}_k^T \underline{B}_k \vec{s}_k - \vec{s}_k^T \vec{y}_k} \right) \vec{s}_k^T \vec{y}_k + \left(1 - \frac{0.8 \vec{s}_k^T \underline{B}_k \vec{s}_k}{\vec{s}_k^T \underline{B}_k \vec{s}_k - \vec{s}_k^T \vec{y}_k} \right) \vec{s}_k^T \underline{B}_k \vec{s}_k$$

$$= \left(\frac{0.8}{1 - \alpha} \right) \alpha \vec{s}_k^T \underline{B}_k \vec{s}_k + \left(1 - \frac{0.8}{1 - \alpha} \right) \vec{s}_k^T \underline{B}_k \vec{s}_k$$

$$= \left(\frac{0.8\alpha + 1 - \alpha - 0.8}{1 - \alpha} \right) \vec{s}_k^T \underline{B}_k \vec{s}_k = \frac{0.2(1 - \alpha)}{1 - \alpha} \vec{s}_k^T \underline{B}_k \vec{s}_k = 0.2 \vec{s}_k^T \underline{B}_k \vec{s}_k$$

$$\rightarrow \vec{s}_k^T \vec{r}_k = 0.2 \vec{s}_k^T \underline{B}_k \vec{s}_k > 0 \quad (\text{if } \underline{B}_k \text{ is P-D} \rightarrow \text{it holds true } \because \underline{B}_0 \text{ is chosen } \vec{I})$$

$\rightarrow \therefore \underline{B}_{k+1}$ becomes P-D because \vec{r}_k is positively oriented (which means $\vec{s}_k^T \vec{r}_k > 0$)

• If $\theta_k = 1 \rightarrow$ It means that \vec{y}_k is positively oriented so we just simply use \vec{y}_k to make P-D \tilde{B}_{k+1} (same as unmodified BFGS)

$\therefore \{\tilde{B}_k\}$ made by 'damped BFGS' is always P-D

-X Damped-BFGS에 대한 고찰.

when $\theta = 0 \rightarrow \tilde{B}_{k+1} = \tilde{B}_k$

when $\theta = 1 \rightarrow \tilde{B}_{k+1} = \text{Approx matrix made by Unmodified BFGS } (\because \vec{y}_k \text{를 그대로 사용})$

when $0 < \theta < 1 \rightarrow \tilde{B}_{k+1} = \text{interpolation of unmodified BFGS and Current } \tilde{B}_k$ *

→ 즉 \vec{y}_k 의 성질에 따라 경우에 따라 damped BFGS와 unmodified BFGS를 쓰기 위해 \tilde{B}_{k+1} 를 만드는 거임.



$$\theta_k = \begin{cases} 0, & \text{then } \tilde{B}_{k+1} = \tilde{B}_k. \\ (0, 1), & \text{then } \tilde{B}_{k+1} \text{ is an interpolate between the two methods.} \\ 1, & \text{then } \mathbf{r}_k = \mathbf{y}_k \text{ and the original BFGS is employed.} \end{cases}$$

✓ 결론 요약

θ_k 기준의 0.2를 0.5로 높이면

- "Hessian 근사 B_{k+1} 가 더 "convex/positive definite"해집니다 (안정성↑)"
- "그러나 새로운 곡률 정보 y_k 의 반영이 줄어, "보수적이고 느린" 업데이트가 됩니다."

즉,

- 수치적 불안정이 잦은 비선형 문제 $\rightarrow c \uparrow$ (예: 0.3~0.5)
- smooth convex 문제 $\rightarrow c \downarrow$ (예: 0.1~0.2)

으로 조정할 수 있고,

대부분의 SQP 구현은 "0.2" 근처가 가장 균형 잡힌 경험적 값이에요.

8-3-2-4) Step length via Merit function

l₁ merit function = $f(\text{obj func})$ + $\mu_k \sum_i^m |\hat{C}_i(\vec{x}_k + \alpha_k \vec{p}_k)| + \sum_i^p \max(-C_i(\vec{x}_k + \alpha_k \vec{p}_k), 0)$

이하 $f(x)$ 만족할 때만 decrease되며 (\leq) violation \leq decrease \Rightarrow 같은 α_k 를 찾도록 문제를 세팅.
이는 α_k constrained optimal step length을 merit function를 사용하여 구하는 대의 경계.

$$\phi_1(\vec{x}_k + \alpha_k \vec{p}_k, \mu_k) = f(\vec{x}_k + \alpha_k \vec{p}_k) + \mu_k \left[\sum_i^m |\hat{C}_i(\vec{x}_k + \alpha_k \vec{p}_k)| + \sum_i^p \max(-C_i(\vec{x}_k + \alpha_k \vec{p}_k), 0) \right]$$

By mathematics, directional derivative of ϕ_1 w.r.t. \vec{p}_k , which is $D_{\vec{p}_k} \phi_1$ at \vec{x}_k , always satisfy below

$$D_{\vec{p}_k} \phi_1(\vec{x}_k) \leq -\vec{p}_k^T W_k \vec{p}_k - (\mu - \|\vec{x}_{k+1}\|_\infty) \sum_i^m |\hat{C}_i(\vec{x}_k + \alpha_k \vec{p}_k)|$$

Gradient of l₁ merit function at $\alpha_k=0$

It means $D_{\vec{p}_k} \phi_1(\vec{x}_k) < 0$!!! \Rightarrow l₁ merit function ϕ_1 의 \vec{x}_k 에서 \vec{p}_k 방향으로의 기울기는 항상 음수!!
(i.e. $\alpha_k=0$)

Therefore we can use step length search algorithm using ϕ_1 , which is very similar to Wolfe-condition.

That is, $\xrightarrow{\text{즉 위 조건이 성립해야만}} \text{Wolfe condition 사용가능.}$

If $\phi(\vec{x}_{k+1}, \mu_k) \leq \phi(\vec{x}_k, \mu_k) + \eta \alpha_k D_{\vec{p}_k}(\phi(\vec{x}_k, \mu_k)) \Rightarrow \alpha_k = \alpha_k^* !!$

$= \vec{x}_k + \alpha_k \vec{p}_k$ Linear function to be compared. $\eta \in (0, 1)$

\Rightarrow We quit finding α_k if we found α_k satisfying above Condition \Rightarrow Same with Armijo Condition!

The thing is, how to choose μ_k ? There are several ways to choose it, but let's pick one of them as following.

$\xrightarrow{\text{penalty parameter for constraint.}}$
 $\xrightarrow{\text{It also plays as a relative weight to objective function } f.}$ $\xrightarrow{\text{thus }} \mu_k$ 인정.

$$\mu_k \geq \frac{\nabla f_k^T \vec{p}_k + \frac{\sigma}{2} \vec{p}_k^T \nabla_{\vec{x}_k}^2 L_k \vec{p}_k}{(1-p) \left(\sum_i |\hat{C}_i(\vec{x}_k)| + \sum_i |C_i(\vec{x}_k)| \right)}$$

where $\sigma = \begin{cases} 1, & \text{if } \vec{p}_k^T \nabla_{\vec{x}_k}^2 L_k \vec{p}_k > 0 \\ 0, & \text{otherwise.} \end{cases}$

$\xrightarrow{\text{Wk by Damped BFGS}} 0 < p < 1$... Heuristic! $\xrightarrow{\text{잡아야 함.}}$
 $\xrightarrow{\text{0.1~0.2 recommended by chetGPT.}}$

Just pick any μ_k that satisfies above condition at k-th iteration and apply it to the l₁ merit function $\phi_1(\vec{x}_{k+1}, \mu_k) !!$

하지만 위와 같이 ϕ_1 함수를 접으면 f 와 \hat{C}_i, C_i 의 모든 모두 줄일 수 있는 방향으로 step length를 update할 수 있다.

자! 이렇게 \vec{x}_{k+1} 에 SQP의 모든 과정을 다 적용해보자. 이를 코드로 표현.

$$(\vec{p}_k, \alpha_k \text{ given} \rightarrow \vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{p}_k)$$

$\xrightarrow{\text{By ALM}} \text{By l}_1\text{-merit function } \phi_1(\vec{x}_k, \mu_k(\sigma, p, \vec{x}_k, \vec{p}_k))$

$D_{\vec{p}_k} \phi_1(\vec{x}_k) < 0$ 이면 Step length을 통해 decrease를 이룰 수 있음. \therefore 저 값을 잘 계산하는 매우 중요!

$$\therefore D_{\vec{p}_k} \phi_1(\vec{x}_k) = \left. \frac{d\phi_1}{d\alpha_k} \right|_{\alpha_k=0} \text{이므로.}$$

$$\phi_1(\vec{x}_k + \alpha_k \vec{p}_k, \mu) = f(\vec{x}_k + \alpha_k \vec{p}_k) + \mu_k \left[\sum_i^m |\hat{C}_i(\vec{x}_k + \alpha_k \vec{p}_k)| + \sum_i^P \{\max(-c_i(\vec{x}_k + \alpha_k \vec{p}_k), 0)\} \right]$$

Let $\vec{y}_k(\alpha_k) = \vec{x}_k + \alpha_k \vec{p}_k$

\vec{y}_k variable α_k constant.

$$\phi_1(\vec{y}_k) = f(\vec{y}_k) + \mu_k \left[\sum_i^m |\hat{C}_i(\vec{y}_k)| + \sum_i^P \{\max(-c_i(\vec{y}_k), 0)\} \right]$$

$$\frac{d\phi_1}{d\alpha_k} = \left(\frac{\partial \phi_1}{\partial y_1} dy_1 + \dots + \frac{\partial \phi_1}{\partial y_n} dy_n \right) / d\alpha_k = \frac{\partial \phi_1}{\partial y_1} \frac{dy_1}{d\alpha_k} + \frac{\partial \phi_1}{\partial y_2} \frac{dy_2}{d\alpha_k} + \dots + \frac{\partial \phi_1}{\partial y_n} \frac{dy_n}{d\alpha_k}$$

\vec{p}_1 \vec{p}_2 \vec{p}_n

$$= \begin{bmatrix} \frac{\partial \phi_1}{\partial y_1} & \dots & \frac{\partial \phi_1}{\partial y_n} \end{bmatrix} \cdot \begin{bmatrix} p_1 & p_2 & \dots & p_n \end{bmatrix} = \nabla_{\vec{y}} \phi_1(\vec{y}_k) \cdot \vec{p}_k$$

$$= \left(\nabla_{\vec{y}} f(\vec{y}_k) + \mu_k \left[\sum_i^m |\nabla \hat{C}_i(\vec{y}_k)| + \sum_i^P \{\nabla \{\max(-c_i(\vec{y}_k), 0)\}\} \right] \right) \cdot \vec{p}_k$$

$$\left. \frac{d\phi_1}{d\alpha_k} \right|_{\alpha_k=0} = \left(\nabla f(\vec{x}_k) + \mu_k \left[\sum_i^m |\nabla \hat{C}_i(\vec{x}_k)| + \sum_i^P \nabla \{\max(-c_i(\vec{x}_k), 0)\} \right] \right) \cdot \vec{p}_k$$

It means
 $\vec{y}_k = \vec{x}_k$

$$\text{If } \hat{C}_i(\vec{x}_k) < 0 \rightarrow \nabla(-\hat{C}_i(\vec{x}_k)) \quad \text{If } c_i(\vec{x}_k) < 0 \rightarrow \nabla(-c_i(\vec{x}_k))$$

$$\hat{C}_i(\vec{x}_k) \geq 0 \rightarrow \nabla \hat{C}_i(\vec{x}_k)$$

$$c_i(\vec{x}_k) \geq 0 \rightarrow \vec{0} \text{ ... size } n \times 1$$

참고로 아래에 대해 풀었습니다.
양수만 해석해 이용합니다
기호

이 경우들(그것이 분명히)
소 양수만 가지거나 같은
없다. 도함수와 원함수는
다른 차이입니다.

아래로 마찬가지로 적용되는 것입니다.

$\Rightarrow 25.12.03$ wed 9) $\left. \frac{d\phi}{d\alpha_k} \right|_{\alpha_k=0}$ 시 \vec{p}_k module-opt.py \vec{p}_k script의
StepLength - merit module-opt-AD.py

스케일이 작은 문제(constrained rosenbrock 문제, breguet range 문제)는 내가 짠 sqp 코드가 답을 잘 찾는다.

그런데

스케일이 큰 문제(설계변수가 많거나, 제약조건이 많거나 한)를 풀면 내가 짠 sqp 코드는 답을 찾지 못한다. alm4sqp에서의 grad_L이 감소하지 않고 오히려 커지며 이것이 전체 sqp 루프에서 x의 최적화를 저지하는 것으로 보인다. 이것을 해결하기 위해서 제약조건에 스케일링(정규화) 적용, alm4sqp의 convergence criteria 기준 grad_LA -> grad_L로 수정, grad_L이 증가하기 전 이른 iteration에서 alm4sqp 종료 유도(종료 tolerance를 일부러 크게 부여함으로써) 등 다양한 것을 시도해봤지만 결국에 sqp 전체 루프에서 실제 최적해 x^* (scipy에서는 잘 찾는..-_-^)로 수렴이 불가했다. 아 참고로 QP 부문제 자체가 infeasible한게 아닌가 해서 feasibility 검사도 했지만 매 k-th iteration에서 QPK 부문제는 feasible한 것으로 판명됐다(GPT 도움으로 scipy의 LP solver 이용한 함수 따로 만들어서 확인함).

module_opt_AD_assignment5.py 파일로 assignment_5_mySQPtest.ipynb 문제에서 여러 시도를 했으나 어쨌든 문제 풀이는 불가했다.

문제 발생과 그것에 대한 조치 시도 등 기록을 남기는 것이 중요하기에 이렇게 기록을 남긴다.

