CONSTRAINED NONLINEAR PROGRAMMING

We now turn to methods for general constrained nonlinear programming. These may be broadly classified into two categories:

1. TRANSFORMATION METHODS: In this approach the constrained nonlinear program is transformed into an unconstrained problem (or a series of unconstrained problems) and solved using one (or some variant) of the algorithms we have already seen.

Transformation methods may further be classified as

(1) Exterior Penalty Function Methods

(2) Interior Penalty (or Barrier) Function Methods

(3) Augmented Lagrangian (or Multiplier) Methods

2. PRIMAL METHODS: These are methods that work directly on the original problem.

We shall look at each of these in turn; we first start with transformation methods.

Exterior Penalty Function Methods

These methods generate a sequence of infeasible points whose limit is an optimal solution to the original problem. The original constrained problem is transformed into an unconstrained one (or a series of unconstrained problems).The constraints are incorporated into the objective by means of a "penalty parameter" which penalizes any constraint violation. Consider the problem

Min $f(\boldsymbol{x})$

st $g_j(\boldsymbol{x}) \le 0, j=1,2,\ldots,m,$ $h_j(\boldsymbol{x}) = 0, j=1,2,\ldots,p,$ $\boldsymbol{x} \in \mathbb{R}^n$

Define a penalty function p as follows:

$$p(x) = \sum_{j=1}^{m} \left[\text{Max}\{0, g_j(\boldsymbol{x})\}\right]^{\alpha} + \sum_{j=1}^{p} \left|h_j(\boldsymbol{x})\right|^{\alpha}$$

where α is some positive integer. It is easy to see that

If $g_j(\boldsymbol{x}) \le 0$ then Max $\{0, g_j(\boldsymbol{x})\} = 0$

If $g_j(\boldsymbol{x}) > 0$ then Max $\{0, g_j(\boldsymbol{x})\} = g_j(\boldsymbol{x})$ (*violation*)

Now define the auxiliary function as

$\mathcal{A}(\boldsymbol{x}) = f(\boldsymbol{x}) + \mu p(\boldsymbol{x})$,where $\mu >> 0$.

Thus if a constraint is violated, $p(\boldsymbol{x})>0$ and a "penalty" of $\mu p(\boldsymbol{x})$ is incurred.

The (UNCONSTRAINED) problem now is:

Minimize $\mathcal{P}(\boldsymbol{x}) = f(\boldsymbol{x}) + \mu p(\boldsymbol{x})$

st $\boldsymbol{x} \in \mathbb{R}^n$

EXAMPLE:

Minimize $f(x)=x$, st $g(x)=2-x \leq 0$, $x \in \mathbb{R}$ (Note that minimum is at $x^* = 2$)

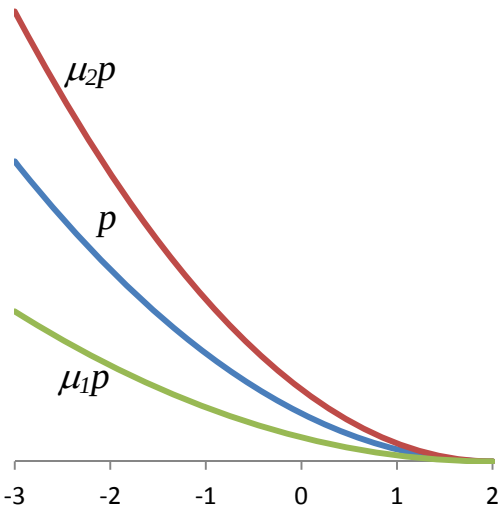
Let $p(x)= \text{Max}[\{0, g(x)\}]^2$,

$$\text{i.e.,} \quad p(x) = \begin{cases} (2-x)^2 & \text{if } x < 2 \\ 0 & \text{if } x \geq 2 \end{cases}$$

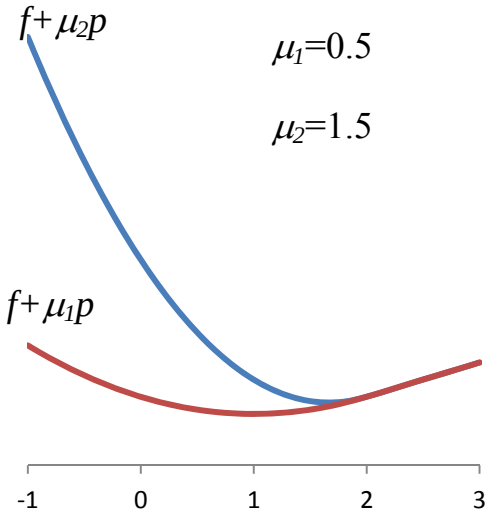
If $p(x)=0$, then the optimal solution to Minimize $f + \mu p$ is at $x^* = -\infty$ and this is infeasible; so $f + \mu p = x + \mu(2-x)^2$

Minimize $f + \mu p$ has optimum solution $x^* = 2 - \frac{1}{2\mu}$.

$Note$: As $\mu \to \infty, \left(2 - \frac{1}{2\mu}\right) \to 2 \ (= x^*)$



Penalty Function



Auxiliary Function

In general:

1. We convert the constrained problem to an unconstrained one by using the penalty function.
2. The solution to the unconstrained problem can be made arbitrarily close to that of the original one by choosing sufficiently large μ.

In practice, if μ is very large, too much emphasis is placed on feasibility. Often, algorithms for unconstrained optimization would stop prematurely because the step sizes required for improvement become very small.

Usually, we solve a sequence of problems with successively increasing μ values; the optimal point from one iteration becomes the starting point for the next problem.

PROCEDURE: Choose the following initially:

1. a tolerance ε,
2. an "increase" factor β,
3. a starting point $\boldsymbol{x}^{\mathbf{1}}$, and
4. an initial μ_1.

At Iteration i

1. Solve the problem Minimize $f(\boldsymbol{x})+\mu_i p(\boldsymbol{x})$; st $\boldsymbol{x}\in\mathbb{X}$ (usually $\mathbb{R}^n$). Use $\boldsymbol{x}^{\boldsymbol{i}}$ as the starting point and let the optimal solution be $\boldsymbol{x}^{\boldsymbol{i+1}}$
2. If $\mu_i p(\boldsymbol{x}^{\boldsymbol{i+1}}) < \varepsilon$ then STOP; else let $\mu_{i+1}=(\mu_i)\beta$ and start iteration $(i+1)$

EXAMPLE: Minimize $f(\boldsymbol{x}) = x_1^2 + 2x_2^2$

st $g(\boldsymbol{x}) = 1 - x_1 - x_2 \leq 0$; $\boldsymbol{x} \in \mathbb{R}^2$

Define the penalty function

$$p(x) = \begin{cases} (1 - x_1 - x_2)^2 & \text{if } g(\boldsymbol{x}) > 0 \\ 0 & \text{if } g(\boldsymbol{x}) \leq 0 \end{cases}$$

The unconstrained problem is

Minimize $x_1^2 + 2x_2^2 + \mu p(\boldsymbol{x})$

If $p(\boldsymbol{x})=0$, then the optimal solution is $\boldsymbol{x}^* = (0,0)$ INFEASIBLE!

$\therefore\ p(\boldsymbol{x}) = (1 - x_1 - x_2)^2,\ \Rightarrow\ f(\boldsymbol{x}) = x_1^2 + 2x_2^2 + \mu\,[(1 - x_1 - x_2)^2]$, and the necessary conditions for the optimal solution ($\nabla \boldsymbol{f}(\boldsymbol{x})=\boldsymbol{0}$) yield the following:

$$\frac{\partial f}{\partial x_1} = 2x_1 + 2\mu(1 - x_1 - x_2)(-1) = 0, \frac{\partial f}{\partial x_2} = 4x_2 + 2\mu(1 - x_1 - x_{2)}(-1) = 0$$

Thus $x_1^* = \dfrac{2\mu}{(2 + 3\mu)}$ and $x_2^* = \dfrac{\mu}{(2 + 3\mu)}$

Starting with $\mu = 0.1$, $\beta = 10$ and $\boldsymbol{x}^1 = (0,0)$ and using a tolerance of 0.005 (say), we have the following:

Iter. (i)	μ_i	$\boldsymbol{x}^{i+1}$	$g(\boldsymbol{x}^{i+1})$	$\mu_i p(\boldsymbol{x}^{i+1})$
1	0.1	(0.087, 0.043)	0.87	0.0757
2	1.0	(0.4, 0.2)	0.40	0.16
3	10	(0.625, 0.3125)	0.0625	0.039
4	100	(0.6622, 0.3311)	0.0067	0.00449
5	1000	(0.666, 0.333)	0.001	0.001

Thus the optimal solution is $\boldsymbol{x}^* = (2/3, 1/3)$.

INTERIOR PENALTY (BARRIER) FUNCTION METHODS

These methods also transform the original problem into an unconstrained one; however the barrier functions prevent the current solution from ever leaving the feasible region. These require that the interior of the feasible sets be nonempty, which is impossible if equality constraints are present. Therefore, they are used with problems having only inequality constraints.

A barrier function B is one that is continuous and nonnegative over the interior of $\{\boldsymbol{x} \mid g(\boldsymbol{x}) \leq 0\}$, i.e., over the set $\{\boldsymbol{x} \mid g(\boldsymbol{x}) < 0\}$, and approaches ∞ as the boundary is approached from the interior.

Let $\phi(y) \geq 0$ if $y<0$ and $\lim_{y \to 0^-} \phi(y) = \infty$

Then $B(\boldsymbol{x}) = \sum_{j=1}^{m} \phi[g_j(x)]$

Usually,

$B(\boldsymbol{x}) = -\sum_{j=1}^{m} \frac{1}{g_j(x)}$ or $B(\boldsymbol{x}) = -\sum_{j=1}^{m} \log(-g_j(\boldsymbol{x}))$

In both cases, note that $\lim_{g_j(x) \to 0^-} B(\boldsymbol{x}) = \infty$

The auxiliary function is now

$$f(\boldsymbol{x}) + \mu B(\boldsymbol{x})$$

where μ is a SMALL positive number.

Q. Why should μ be SMALL?

A. Ideally we would like $B(\boldsymbol{x})=0$ if $g_j(\boldsymbol{x})<0$ and $B(\boldsymbol{x})=\infty$ if $g_j(\boldsymbol{x})=0$, so that we never leave the region $\{\boldsymbol{x} \mid g(\boldsymbol{x}) \leq 0\}$. However, $B(\boldsymbol{x})$ is now discontinuous. This causes serious computational problems during the unconstrained optimization.

Similar to exterior penalty functions we don't just choose one small value for rather we start with some μ_1 and generate a sequence of points.

PROCEDURE: Initially choose a tolerance ε, a "decrease" factor β, an interior starting point $\boldsymbol{x}^1$, and an initial μ_1.

At Iteration i

1. Solve the problem Minimize $f(\boldsymbol{x})+\mu_i B(\boldsymbol{x})$,st $\boldsymbol{x} \in \mathbb{X}$ (usually $\mathbb{R}^n$). Use $\boldsymbol{x}^i$ as the starting point and let the optimal solution be $\boldsymbol{x}^{i+1}$
2. If $\mu_i B(\boldsymbol{x}^{i+1}) < \varepsilon$ then STOP; else let $\mu_{i+1}=(\mu_i)\beta$ and start iteration $(i+1)$

Consider the previous example once again…

EXAMPLE: Minimize $f(\boldsymbol{x}) = x_1^2 + 2x_2^2$

st $g(\boldsymbol{x}) = 1 - x_1 - x_2 \leq 0$; $\boldsymbol{x} \in \mathbb{R}^2$

Define the barrier function

$$B(x) = -\log\big(-g(\boldsymbol{x})\big) = -\log(x_1 + x_2 - 1)$$

The unconstrained problem is

Minimize $x_1^2 + 2x_2^2 + \mu B(\boldsymbol{x}) = x_1^2 + 2x_2^2 - \mu \log(x_1 + x_2 - 1)$

The necessary conditions for the optimal solution ($\nabla \boldsymbol{f}(\boldsymbol{x})=\boldsymbol{0}$) yield the following:

$$\frac{\partial f}{\partial x_1} = 2x_1 - \mu/(x_1 + x_2 - 1) = 0, \frac{\partial f}{\partial x_2} = 4x_2 - \mu/(x_1 + x_2 - 1) = 0$$

Solving, we get $x_1 = \dfrac{1 \pm \sqrt{1 + 3\mu}}{3}$ and $x_2 = \dfrac{1 \pm \sqrt{1 + 3\mu}}{6}$

Since the negative signs lead to infeasibility, we $x_1^* = \frac{1+\sqrt{1+3\mu}}{3}$ and $x_2^* = \frac{1+\sqrt{1+3\mu}}{6}$

Starting with μ=1, β=0.1 and $\boldsymbol{x}^1$=(0,0) and using a tolerance of 0.005 (say), we have the following:

Iter. (i)	μ_i	$\boldsymbol{x}^{i+1}$	$g(\boldsymbol{x}^{i+1})$	$\mu_i B(\boldsymbol{x}^{i+1})$
1	1	(1.0, 0.5)	-0.5	0.693
2	0.1	(0.714, 0.357)	-0.071	0.265
3	0.01	(0.672, 0.336)	-0.008	0.048
4	0.001	(0.6672, 0.3336)	-0.0008	0.0070
5	0.0001	(0.6666, 0.3333)	-0.0001	0.0009

Thus the optimal solution is $\boldsymbol{x}^*$= (2/3, 1/3).

Penalty Function Methods and Lagrange Multipliers

Consider the penalty function approach to the problem below

Minimize $f(\boldsymbol{x})$

st $\quad g_j(\boldsymbol{x}) \le 0;\ j = 1,2,\ldots,m$

$\quad h_j(\boldsymbol{x}) = 0;\ j = m+1,m+2,\ldots,l \qquad \boldsymbol{x}^{\boldsymbol{k}} \in \mathbb{R}^n.$

Suppose we use the usual interior penalty function described earlier, with p=2.

The auxiliary function that we minimize is then given by

$$\mathcal{P}(\boldsymbol{x}) = f(\boldsymbol{x}) + \mu p(\boldsymbol{x}) = f(\boldsymbol{x}) + \mu\left\{\sum_j [\text{Max}(0, g_j(\boldsymbol{x}))]^2 + \sum_j [h_j(\boldsymbol{x})]^2\right\}$$

$$= f(\boldsymbol{x}) + \left\{\sum_j \mu[\text{Max}(0, g_j(\boldsymbol{x}))]^2 + \sum_j \mu[h_j(\boldsymbol{x})]^2\right\}$$

The necessary condition for this to have a minimum is that

$$\nabla\mathcal{P}(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) + \mu\nabla p(\boldsymbol{x}) = 0, \qquad \text{i.e.,}$$

$$\nabla f(\boldsymbol{x}) + \sum_j 2\mu[\text{Max}(0, g_j(\boldsymbol{x})]\nabla g_j(\boldsymbol{x}) + \sum_j 2\mu[h_j(\boldsymbol{x})]\nabla h_j(\boldsymbol{x}) = 0 \qquad (\mathbf{1})$$

Suppose that the solution to (**1**) for a fixed μ (say μ_k>0) is given by $\boldsymbol{x}^{\boldsymbol{k}}$.

Let us also designate

$$2\mu_k[\text{Max}\{0, g_j(\boldsymbol{x}^{\boldsymbol{k}})\}] = \lambda_j(\mu_k); \qquad j=1,2,\ldots,m \qquad (\mathbf{2})$$

$$2\mu_k[h_j(\boldsymbol{x}^{\boldsymbol{k}})] = \lambda_j(\mu_k); \qquad j=m+1,\ldots,l \qquad (\mathbf{3})$$

so that for $\mu=\mu_k$ we may rewrite (**1**) as

$$\nabla f(\boldsymbol{x}) + \sum_j \lambda_j(\mu_k)\, \nabla g_j(\boldsymbol{x}) + \sum_j \lambda_j(\mu_k)\, \nabla h_j(\boldsymbol{x}) = 0 \qquad (\mathbf{4})$$

Now consider the Lagrangian for the original problem:

$$L(\boldsymbol{x},\boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_j \lambda_j g_j(\boldsymbol{x}) + \sum_j \lambda_j h_j(\boldsymbol{x})$$

The usual K-K-T necessary conditions yield

$$\nabla \boldsymbol{f}(\boldsymbol{x}) + \sum_{j=1..m} \{\lambda_j \nabla \boldsymbol{g_j}(\boldsymbol{x})\} + \sum_{j=m+1..l} \{\lambda_j \nabla \boldsymbol{h_j}(\boldsymbol{x})\} = 0 \qquad (\mathbf{5})$$

plus the original constraints, complementary slackness for the inequality constraints and $\lambda_j \geq 0$ for $j=1,\ldots,m$.

Comparing (**4**) and (**5**) we can see that when we minimize the auxiliary function using $\mu=\mu_k$, the $\lambda_j(\mu_k)$ values given by (**2**) and (**3**) estimate the Lagrange multipliers in (**5**). In fact it may be shown that as the penalty function method proceeds and $\mu_k \to \infty$ and the $\boldsymbol{x}^{\boldsymbol{k}} \to \boldsymbol{x}^{*}$, the optimum solution, the values of $\lambda_j(\mu_k) \to \lambda_j^{*}$, the optimum Lagrange multiplier value for constraint j.

Consider our example on page 153 once again. For this problem the Lagrangian is given by $L(\boldsymbol{x},\lambda) = x_1^2 + 2x_2^2 + \lambda(1-x_1-x_2)$.

The K-K-T conditions yield

$$\partial L/\partial x_1 = 2x_1-\lambda = 0; \qquad \partial L/\partial x_2 = 4x_2-\lambda = 0; \qquad \lambda(1-x_1-x_2)=0$$

Solving these results in $x_1^{*}=2/3$; $x_2^{*}=1/3$; $\lambda^{*}=4/3$; ($\lambda=0$ yields an infeasible solution).

Recall that for fixed μ_k the optimum value of $\boldsymbol{x_k}$ was $[2\mu_k/(2+3\mu_k) \quad \mu_k/(2+3\mu_k)]^T$. As we saw, when $\mu_k \to \infty$, these converge to the optimum solution of $\boldsymbol{x}^* = (2/3,1/3)$.

Now, suppose we use (**2**) to define $\lambda(\mu) = 2\mu[\text{Max}(0, g_j(\boldsymbol{x}^k)]$

$$= \quad 2\mu[1 - \{2\mu/(2+3\mu)\} - \{\mu/(2+3\mu)\}] \qquad (\text{since } g_j(\boldsymbol{x}^k)>0 \text{ if } \mu>0)$$

$$= \quad 2\mu[1 - \{3\mu/(2+3\mu)\}] = 4\mu/(2+3\mu)$$

Then it is readily seen that $\text{Lim}_{\mu\to\infty}\lambda(\mu) = \text{Lim}_{\mu\to\infty}4\mu/(2+3\mu) = 4/3 = \lambda^*$

Similar statements can also be made for the barrier (interior penalty) function approach, e.g., if we use the log-barrier function we have

$$\mathcal{P}(\boldsymbol{x}) = f(\boldsymbol{x}) + \mu\, p(\boldsymbol{x}) = f(\boldsymbol{x}) + \mu\textstyle\sum_j \text{-log}\big(\text{-}g_j(\boldsymbol{x})\big),$$

so that

$$\nabla\mathcal{P}(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) - \textstyle\sum_j \{\mu/g_j(\boldsymbol{x})\}\nabla g_j(\boldsymbol{x}) = 0 \qquad (\mathbf{6})$$

If (like before) for a fixed μ_k we denote the solution by $\boldsymbol{x}^k$ and define $-\mu_k/g_j(\boldsymbol{x}^k) = \lambda_j(\mu_k)$, then from (**6**) and (**5**) we see that $\lambda_j(\mu_k)$ approximates λ_j. Furthermore, as $\mu_k \to 0$ and the $\boldsymbol{x}^k \to \boldsymbol{x}^*$, it can be shown that $\lambda_j(\mu_k) \to \lambda_j^*$

For our example (page 156) $-\mu_k/g_j(\boldsymbol{x}^k) = \mu_k/\left(1 - \frac{1+\sqrt{1+3\mu}}{3} - \frac{1+\sqrt{1+3\mu}}{6}\right) =$

$-2\mu_k/(1-\sqrt{1+3\mu_k}) \to 4/3$, as $\mu_k \to 0$.

Penalty and Barrier function methods have been referred to as “SEQUENTIAL UNCONSTRAINED MINIMIZATION TECHNIQUES” (SUMT) and studied in detail by Fiacco and McCormick. While they are attractive in the simplicity of the principle on which they are based, they also possess several undesirable properties.

When the parameters μ are very large in value, the penalty functions tend to be ill-behaved near the boundary of the constraint set where the optimum points usually lie. Another problem is the choice of appropriate μ_1 and β values. The rate at which μ_i change (i.e. the β values) can seriously affect the computational effort to find a solution. Also as μ increases, the Hessian of the unconstrained function becomes ill-conditioned.

Some of these problems are addressed by the so-called "multiplier" methods. Here there is no need for μ to go to infinity, and the unconstrained function is better conditioned with no singularities. Furthermore, they also have faster rates of convergence than SUMT.

Ill-Conditioning of the Hessian Matrix

Consider the Hessian matrix of the Auxiliary function $\mathcal{P}(\boldsymbol{x})=x_1^2+2x_2^2+\mu(1-x_1-x_2)^2$:

$\mathbf{H} = \begin{bmatrix} 2+2\mu & 2\mu \\ 2\mu & 4+2\mu \end{bmatrix}$. Suppose we want to find its eigenvalues by solving

$|\mathbf{H}-\lambda\mathbf{I}| = (2+2\mu-\lambda)*(4+2\mu-\lambda) - 4\mu^2$

$= \lambda^2 - (6+4\mu)\lambda + (8+12\mu) = 0$

This quadratic equation yields

$\lambda = (3+2\mu) \pm \sqrt{4\mu^2+1}$

Taking the ratio of the largest and the smallest eigenvalue yields

$\frac{(3+2\mu)+\sqrt{4\mu^2+1}}{(3+2\mu)-\sqrt{4\mu^2+1}}$. It should be clear that as $\mu\to\infty$, the limit of the preceding ratio also goes to ∞. This indicates that as the iterations proceed and we start to increase the value of μ, the Hessian of the unconstrained function that we are minimizing becomes increasingly ill-conditioned. This is a common situation and is especially problematic if we are using a method for the unconstrained optimization that requires the use of the Hessian.

MULTIPLIER METHODS

Consider the problem: Min $f(\boldsymbol{x})$, st $g_j(\boldsymbol{x})\leq 0$, $j=1,2,...,m$.

In multiplier methods the auxiliary function is given by

$$\mathcal{P}(\boldsymbol{x}) = f(\boldsymbol{x}) + \tfrac{1}{2}\sum_{j=1}^{m}\mu_j\left[\text{Max}\{0, g(x)+\theta_j^i\}\right]^2$$

where $\mu_j>0$ and $\theta_j^i \geq 0$ are parameters associated with the j^{th} constraint. This is also often referred to as the *AUGMENTED LAGRANGIAN* function

Note that if $\theta_j^i=0$ and $\mu_j=\mu$, this reduces to the usual exterior penalty function.

The basic idea behind multiplier methods is as follows:

Let $\boldsymbol{x^i}$ be the minimum of the auxiliary function $\mathcal{P}(\boldsymbol{x})$ at some iteration i. From the optimality conditions we have

$$\boldsymbol{\nabla \mathcal{P}^i(x^i) = \nabla f(x^i) + \sum_{j=1}^{m}\{\mu_j[Max\{0, g_j(x^i)+\theta_j^i\}]\nabla g_j(x^i)\}}$$

Now, $Max\{0, g_j+\theta_j^i\} = \theta_j^i + Max(-\theta_j^i, g_j)$. Therefore $\boldsymbol{\nabla \mathcal{P}^i(x^i) = 0}$ implies

$$\boldsymbol{\nabla f(x^i) + \sum_{j=1}^{m}\{\mu_j\theta_j^i\nabla g_j(x^i)\} + \sum_{j=1}^{m}\{\mu_j Max\{-\theta_j^i, g_j(x^i)\}\nabla g_j(x^i)\} = 0}$$

Let us assume for a moment that θ_j^i are chosen at each iteration i so that they satisfy the following:

$$\boxed{Max(-\theta_j^i, g_j) = 0} \qquad (1)$$

It is easily verified that (1) is equivalent to requiring that

$$\theta_j^i \geq 0, g_j(\boldsymbol{x^i}) \leq 0 \quad \text{and} \quad \theta_j^i g_j(\boldsymbol{x^i}) = 0 \qquad (2)$$

Therefore as long as (2) is satisfied, we have for the point $\boldsymbol{x^i}$ that minimize $\mathcal{P}(\boldsymbol{x})$

$$\boldsymbol{\nabla \mathcal{P}^i(x^i) = \nabla f(x^i) + \sum_{j=1}^{m}\{\mu_j \theta_j^i \nabla g_j(x^i)\} = 0} \tag{3}$$

If we let $\mu_j\theta_j^i = \lambda_j^i$ and use the fact that μ_j>0, then (2) reduces to

$$\lambda_j^i \geq 0, g_j(\boldsymbol{x^i}) \leq 0 \qquad \text{and} \qquad \lambda_j^i \cdot g_j(\boldsymbol{x^i}) = 0, \forall j \tag{$\mathcal{A}$}$$

and (3) reduces to

$$\boldsymbol{\nabla f(x^i) + \sum_{j=1}^{m} \lambda_j^i \nabla g_j(x^i) = 0}. \tag{$\mathcal{B}$}$$

It is readily seen that ($\mathcal{A}$) and ($\mathcal{B}$) are merely the KARUSH-KUHN-TUCKER necessary conditions for $\boldsymbol{x^i}$ to be a solution to the original problem below!!

Min $f(\boldsymbol{x})$, st $g_j(\boldsymbol{x})\leq 0$, j=1,2,...,m.

We may therefore conclude that

$$\boldsymbol{x^i = x^*} \text{ and } \mu_j\theta_j^i = \mu_j\theta_j^* = \lambda_j^*$$

Here $\boldsymbol{x^*}$ is the solution to the problem, and λ_j^* is the optimum Lagrange multiplier associated with constraint j.

From the previous discussion it should be clear that at each iteration θ_j should be chosen in such a way that $Max(-\theta_j^i, g_j) \rightarrow 0$, which results in $\boldsymbol{x^i \rightarrow x^*}$. Note that $\boldsymbol{x^i}$ is obtained here by <u>minimizing</u> the auxiliary function with respect to $\boldsymbol{x}$.

ALGORITHM: A general algorithmic approach for the multiplier method may now be stated as follows:

STEP 0: Set i=0, choose vectors $\boldsymbol{x}^{\boldsymbol{i}}$, $\boldsymbol{\mu}$ and $\boldsymbol{\theta}^{\boldsymbol{i}}$.

STEP 1: Set i=i+1.

STEP 2: Starting at $\boldsymbol{x}^{\boldsymbol{i}\text{-}1}$, Minimize $\mathcal{P}(\boldsymbol{x})$ to find $\boldsymbol{x}^{\boldsymbol{i}}$.

STEP 3: Check convergence criteria and go to Step 4 only if the criteria are not satisfied.

STEP 4: Modify $\boldsymbol{\theta}^{\boldsymbol{i}}$ based on satisfying (1). Also modify $\boldsymbol{\mu}$ if necessary and go to Step 2.

(Usually $\boldsymbol{\theta}^{0}$ is set to $\mathbf{0}$…)

One example of a formula for changing $\boldsymbol{\theta}^{\boldsymbol{i}}$ is the following:

$$\boldsymbol{\theta}^{\boldsymbol{i}+1} = \boldsymbol{\theta}^{\boldsymbol{i}} + \left[\boldsymbol{max}\left(g_j\left(\boldsymbol{x}^{\boldsymbol{i}}\right), -\theta_j^i\right)\right]$$

PRIMAL METHODS

By a primal method we mean one that works directly on the original constrained problem:

Min $f(\boldsymbol{x})$

st $g_j(\boldsymbol{x}) \leq 0$ $j=1,2,...,m$.

Almost all methods are based on the following general strategy:

Let $\boldsymbol{x^i}$ be a design at iteration i. A new design $\boldsymbol{x^{i+1}}$ is found by the expression $\boldsymbol{x^{i+1}} = \boldsymbol{x^i} + \alpha_i \boldsymbol{d^i}$, where α_i is a step size parameter and $\boldsymbol{d^i}$ is a search direction (vector). The direction vector $\boldsymbol{d^i}$ is typically determined by an expression of the form:

$$\boldsymbol{d^i} = \beta_0 \nabla \boldsymbol{f}(\boldsymbol{x^i}) + \sum_{j \in J_\varepsilon} \beta_j \nabla \boldsymbol{g_j}(\boldsymbol{x^i})$$

where $\nabla \boldsymbol{f}$ and $\nabla \boldsymbol{g_j}$ are gradient vectors of the objective and constraint functions and J_ε is an “active set" given by $J_\varepsilon = \{j | g_j(\boldsymbol{x^i}) + \varepsilon \geq 0,\ \varepsilon > 0\}$.

Unlike with transformation methods, it is evident that here the gradient vectors of individual constraint functions need to be evaluated. This is a characteristic of all primal methods.

METHOD OF FEASIBLE DIRECTIONS

The first class of primal methods we look at are the feasible directions methods, where each $\boldsymbol{x}^{\boldsymbol{i}}$ is within the feasible region. An advantage is that if the process is terminated before reaching the true optimum solution (as is usually necessary with large-scale problems) the terminating point is feasible and hence acceptable. The general strategy at iteration *i* is:

(1) Let $\boldsymbol{x}^{\boldsymbol{i}}$ be a feasible point.

(2) Choose a direction $\boldsymbol{d}^{\boldsymbol{i}}$ so that

a) $\boldsymbol{x}^{\boldsymbol{i}}+\alpha\boldsymbol{d}^{\boldsymbol{i}}$ is feasible at least for some "sufficiently" small $\alpha>0$, and

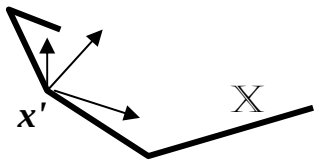
b) $f(\boldsymbol{x}^{\boldsymbol{i}}+\alpha\boldsymbol{d}^{\boldsymbol{i}}) < f(\boldsymbol{x}^{\boldsymbol{i}})$.

(3) Do an unconstrained optimization

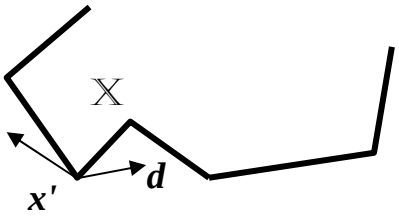
$\text{Min}_{\alpha}\, f(\boldsymbol{x}^{\boldsymbol{i}}+\alpha\boldsymbol{d}^{\boldsymbol{i}})$ to obtain $\alpha^{*}=\alpha_i$ and hence $\boldsymbol{x}^{\boldsymbol{i+1}}=\boldsymbol{x}^{\boldsymbol{i}}+\alpha_i\boldsymbol{d}^{\boldsymbol{i}}$

DEFINITION: For the problem Minimize $f(\boldsymbol{x})$, st $\boldsymbol{x}\in\mathbb{X}$

$\boldsymbol{d}$ ($\neq\boldsymbol{0}$) is called a **feasible direction** at $\boldsymbol{x}'\in\mathbb{X}$ if $\exists\ \delta>0$ ∋ $(\boldsymbol{x}'+\alpha\boldsymbol{d})\in\mathbb{X}$ for $\alpha\in[0,\delta)$



feasible directions



not feasible directions

Further, if we also have $f(\boldsymbol{x}'+\alpha\boldsymbol{d}) < f(\boldsymbol{x}')$ for all $\alpha\in[0,\alpha')$, $\alpha' < \delta$, then $\boldsymbol{d}$ is called an **improving feasible direction**.

Under differentiability, recall that this implies

$$\nabla \boldsymbol{f}^{\mathrm{T}}(\boldsymbol{x}')\cdot\boldsymbol{d} < 0$$

Let $F_{\boldsymbol{x}'} = \{\boldsymbol{d} \mid \nabla \boldsymbol{f}^{\mathrm{T}}(\boldsymbol{x}')\cdot\boldsymbol{d} < 0\}$, and

$D_{\boldsymbol{x}'} = \{\boldsymbol{d} \mid \exists\ \delta>0 \ni (\boldsymbol{x}'+\alpha\boldsymbol{d})\in \mathbb{X} \text{ for all } \alpha\in[0,\delta)\}$,

Thus $F_{\boldsymbol{x}'}$ is the set of **improving directions** at $\boldsymbol{x}'$, and

$D_{\boldsymbol{x}'}$ is the set of **feasible directions** at $\boldsymbol{x}'$.

If $\boldsymbol{x}^*$ is a local optimum then $F_{\boldsymbol{x}^*} \cap D_{\boldsymbol{x}^*} = \phi$ *as long as a constraint qualification is met*.

In general, for the problem

Minimize $f(\boldsymbol{x})$, st $g_j(\boldsymbol{x})\le 0$, $j=1,2,...,m$.

let $J_{\boldsymbol{x}'}=\{j \mid g_j(\boldsymbol{x}')=0\}$ (Index set of active constraints)

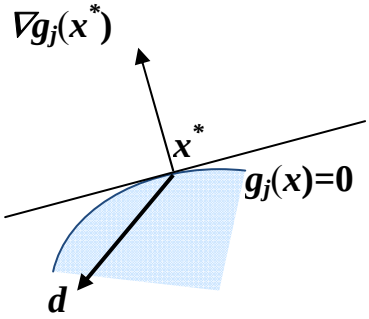
We can show that the set of feasible directions at $\boldsymbol{x}'$ is

$$D_{\boldsymbol{x}'} = \{\boldsymbol{d} \mid \nabla \boldsymbol{g_j}^{\mathrm{T}}(\boldsymbol{x}')\cdot\boldsymbol{d} < 0 \text{ for all } j\in J_{\boldsymbol{x}'}\}$$

NOTE: If $g_j(\boldsymbol{x})$ is a linear function, then the strict inequality (<) used to define the set $D_{\boldsymbol{x}'}$ above can be relaxed to an inequality (≤).

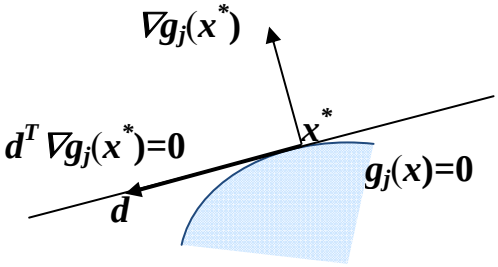
Geometric Interpretation

Minimize $f(\boldsymbol{x})$, st $g_j(\boldsymbol{x}) \le 0$, $j=1,2,\ldots,m$



$d^T \nabla g_j(x^*)<0$ ***d*** *is a feasible direction*

(i)



d *is tangent to the constraint boundary*

(ii)

In (ii) ***d*** any positive step in this direction is infeasible (however we will often consider this a "feasible" direction…)

FARKAS' LEMMA: Given $\boldsymbol{A}\in\mathbb{R}^{m\times n}$, $\boldsymbol{b}\in\mathbb{R}^m$, $\boldsymbol{x}\in\mathbb{R}^n$, $\boldsymbol{y}\in\mathbb{R}^m$, the following statements are equivalent to each other:

1. $\boldsymbol{y}^T\boldsymbol{A} \le \boldsymbol{0} \Rightarrow \boldsymbol{y}^T\boldsymbol{b} \le 0$
2. $\exists\ \boldsymbol{z}$ such that $\boldsymbol{Az=b}$, $\boldsymbol{z}\ge\boldsymbol{0}$

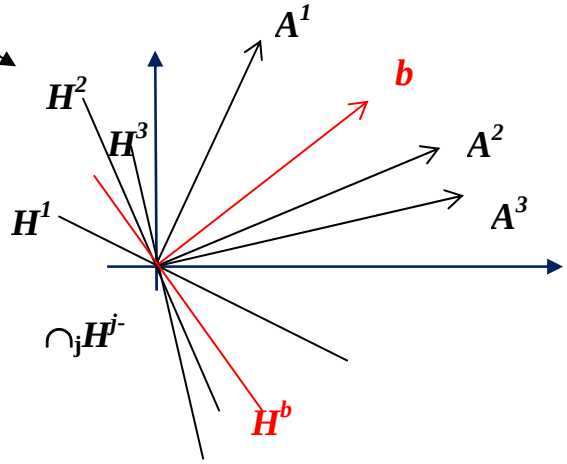
$\boldsymbol{y}^T\boldsymbol{A}\le \boldsymbol{0} \Leftrightarrow \boldsymbol{y}\in \cap_j H^{j-}$

where H^{j-} *is the closed half-space on the side of* H^j *that does not contain* A^j

So…

$\boldsymbol{y}^T\boldsymbol{A}\le \boldsymbol{0} \Rightarrow \boldsymbol{y}^T\boldsymbol{b} \le \boldsymbol{0}$ *simply implies that* $(\cap_j H^{j-}) \subseteq H^{b-}$

Suppose H^j *is the hyperplane through the origin that is orthogonal to* A^j



Application to NLP

Let $\boldsymbol{b} \equiv -\nabla \boldsymbol{f}(\boldsymbol{x}^*)$ $\qquad$ $\boldsymbol{A}^j \equiv \nabla \boldsymbol{g}_j(\boldsymbol{x}^*)$

$\boldsymbol{y} \equiv \boldsymbol{d}$ (direction vector), $\qquad$ $z_j \equiv \lambda_j$ for $j \in \mathcal{J}_{\boldsymbol{x}^*} \equiv \{j \mid \boldsymbol{g}_j(\boldsymbol{x}^*)=0\}$

Farkas' Lemma then implies that

(1) $\boldsymbol{d}^T \nabla \boldsymbol{g}_j(\boldsymbol{x}^*) \le 0 \;\; \forall\, j \in \mathcal{J}_{\boldsymbol{x}^*} \Rightarrow -\boldsymbol{d}^T \nabla \boldsymbol{f}(\boldsymbol{x}^*) \le 0$, i.e., $\boldsymbol{d}^T \nabla \boldsymbol{f}(\boldsymbol{x}^*) \ge 0$

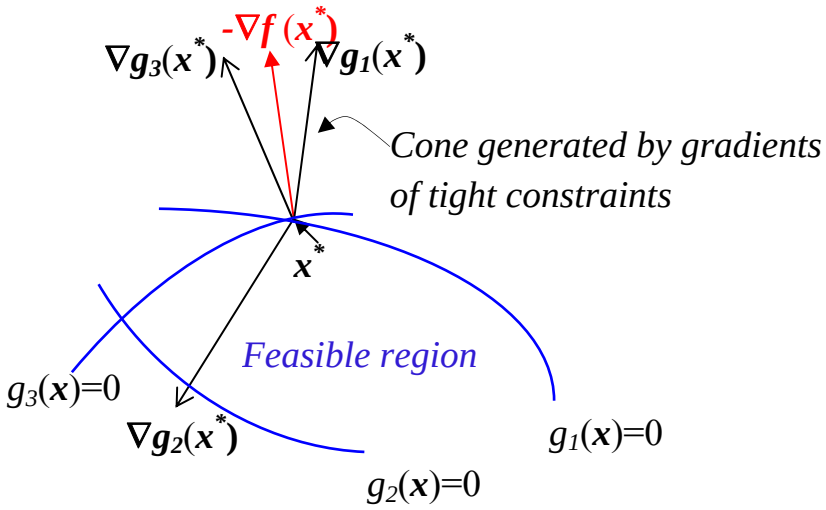
(2) $\exists\, \lambda_j \ge 0$ such that $\sum_{j \in \mathcal{J}_{\boldsymbol{x}^*}} \lambda_j \nabla \boldsymbol{g}_j(\boldsymbol{x}^*) = -\nabla \boldsymbol{f}(\boldsymbol{x}^*)$

are equivalent! Note that (1) indicates that

- directions satisfying $\boldsymbol{d}^T \nabla \boldsymbol{g}_j(\boldsymbol{x}^*) \le 0 \;\forall\, j \in \mathcal{J}_{\boldsymbol{x}^*}$ are feasible directions
- directions satisfying $\boldsymbol{d}^T \nabla \boldsymbol{f}(\boldsymbol{x}^*) \ge 0$ are ascent (non-improving) directions

On the other hand (2) indicates the K-K-T conditions. **They are equivalent!**

So at the optimum, there is no feasible direction that is improving…



K-K-T implies that the steepest descent direction is in the cone generated by gradients of tight constraints

Similarly for the general NLP problem

Minimize $f(\boldsymbol{x})$

st $g_j(\boldsymbol{x}) \le 0,\ j=1,2,...,m$

$h_k(\boldsymbol{x}) = 0,\ k=1,2,...,p$

the set of feasible directions is

$D_{x'} = \{\boldsymbol{d} \mid \nabla \boldsymbol{g_j}^T(\boldsymbol{x}')\cdot\boldsymbol{d} < 0\ \forall\ j \in J_{x'}, \text{ and } \nabla \boldsymbol{h_k}^T(\boldsymbol{x}')\cdot\boldsymbol{d} = 0\ \forall\ k\}$

==========

In a feasible directions algorithm we have the following:

STEP 1 (*direction finding*)

To generate an improving feasible direction from $\boldsymbol{x}'$, we need to find $\boldsymbol{d}$ such that

$F_{x'} \cap D_{x'} \neq \phi$, i.e., ∋

$\nabla \boldsymbol{f}^T(\boldsymbol{x}')\cdot\boldsymbol{d} < 0$ and $\nabla \boldsymbol{g_j}^T(\boldsymbol{x}')\cdot\boldsymbol{d} < 0$ for $j \in J_{x'}$, $\nabla \boldsymbol{h_k}^T(\boldsymbol{x}')\cdot\boldsymbol{d} = 0\ \forall\ k$

We therefore solve the subproblem

Minimize $\nabla \boldsymbol{f}^T(\boldsymbol{x}')\cdot\boldsymbol{d}$

st $\nabla \boldsymbol{g_j}^T(\boldsymbol{x}')\cdot\boldsymbol{d} < 0$ for $j \in J_{x'}$

(≤ 0 for j ∋ g_j linear)

$\nabla \boldsymbol{h_k}^T(\boldsymbol{x}')\cdot\boldsymbol{d} = 0$ for all k,

along with some "normalizing" constraints such as

$-1 \le d_i \le 1$ for $i=1,2,...,n$ or $\|\boldsymbol{d}\|^2 = \boldsymbol{d}^T\boldsymbol{d} \le 1$.

If the objective of this subproblem is negative, we have an improving feasible direction $\boldsymbol{d}^i$.

In practice, the actual subproblem we solve is slightly different since strict inequality (<) constraints are difficult to handle.

Let z = Max $[\nabla \boldsymbol{f}^{T}(\boldsymbol{x}')\cdot\boldsymbol{d}, \nabla \boldsymbol{g_j}^{T}(\boldsymbol{x}')\cdot\boldsymbol{d}$ for $j\in J_{x'}]$. Then we solve:

Minimize z

st $\nabla \boldsymbol{f}^{T}(\boldsymbol{x}')\cdot\boldsymbol{d} \le z$

$\nabla \boldsymbol{g_j}^{T}(\boldsymbol{x}')\cdot\boldsymbol{d} \le z$ for $j\in J$

$\nabla \boldsymbol{h_k}^{T}(\boldsymbol{x}')\cdot\boldsymbol{d} = 0$ for k=1,2,...,p,

$-1 \le d_i \le 1$ for i=1,2,...,n

If $(z^*,\boldsymbol{d}^*)$ is the optimum solution for this then

a) $z^* < 0 \Rightarrow \boldsymbol{d} = \boldsymbol{d}^*$ is an improving feasible direction

b) $z = 0 \Rightarrow \boldsymbol{x}^*$ is a Fritz John point (or if a CQ is met, a K-K-T point)

(Note that z^* can never be greater than 0)

STEP 2: (*line search for step size*)

Assuming that z^* from the first step is negative, we now solve

Min $f(\boldsymbol{x}^i+\alpha\boldsymbol{d}^i)$

st $g_j(\boldsymbol{x}^i+\alpha\boldsymbol{d}^i) \le 0,\ \alpha\ge 0$

This can be rewritten as

Min $f(\boldsymbol{x}^i+\alpha\boldsymbol{d}^i)$

st $0\le\alpha\le\alpha_{max}$

where α_{max}= supremum $\{\alpha \mid g_j(\boldsymbol{x}^i+\alpha\boldsymbol{d}^i) \le 0, j=1,2,...,m\}$. Let the optimum solution to this be at $\alpha^*=\alpha_i$.

Then we set $\boldsymbol{x}^{i+1}= (\boldsymbol{x}^i+\alpha\boldsymbol{d}^i)$ and return to Step 1.

NOTE: (1) In practice the set $J_{x'}$ is usually defined as $J_{x'}=\{ j \mid g_j(\boldsymbol{x}^i)+\varepsilon \geq 0\}$, where the tolerance ε_i is referred to as the "constraint thickness," which may be reduced as the algorithm proceeds.

(2) This procedure is usually attributed to ZOUTENDIJK

EXAMPLE

Minimize $2x_1^2+ x_2^2- 2x_1x_2 - 4x_1 - 6x_2$

st $x_1 + x_2 \leq 8$

$-x_1 + 2x_2 \leq 10$

$-x_1 \leq 0$

$-x_2 \leq 0$ (A **QP**…)

For this, we have

$$\nabla g_1(x) = \begin{bmatrix}1\\1\end{bmatrix}, \quad \nabla g_2(x) = \begin{bmatrix}-1\\2\end{bmatrix}, \quad \nabla g_3(x) = \begin{bmatrix}-1\\0\end{bmatrix}, \quad \nabla g_4(x) = \begin{bmatrix}0\\-1\end{bmatrix},$$

and $\nabla f(x) = \begin{bmatrix}4x_1 - 2x_2 - 4\\2x_2 - 2x_1 - 6\end{bmatrix}$.

Let us begin with $\boldsymbol{x^0} = \begin{bmatrix}0\\0\end{bmatrix}$, with $f(\boldsymbol{x}^0)=0$

ITERATION 1 $J=\{3,4\}$. STEP 1 yields

$$\begin{array}{lll} \text{Min } z & & \text{Min } z \\ \text{st} \quad \nabla \boldsymbol{f}^{\mathrm{T}}(\boldsymbol{x}^1)\cdot\boldsymbol{d} \leq z & & d_1(4x_1\text{-}2x_2\text{-}4) + d_2(2x_2\text{-}2x_1\text{ -}6) \leq z \\ \quad \nabla \boldsymbol{g_3}^{\mathrm{T}}(\boldsymbol{x}^i)\cdot\boldsymbol{d} \leq z & \Rightarrow & -d_1 \leq z \\ \quad \nabla \boldsymbol{g_4}^{\mathrm{T}}(\boldsymbol{x}^i)\cdot\boldsymbol{d} \leq z & & -d_2 \leq z \\ \quad -1 \leq d_1, d_2 \leq 1 & & d_1, d_2 \in [-1,1] \end{array}$$

i.e., Min z

st $-4d_1 - 6d_2 \le z$, $-d_1 \le z$, $-d_2 \le z$ $d_1, d_2 \in [-1,1]$

yielding $z^* = -1$ (<0), with $\boldsymbol{d}^* = \boldsymbol{d}^0 = \begin{bmatrix}1\\1\end{bmatrix}$

STEP 2 (*line search*):

$$\boldsymbol{x}^0 + \alpha \boldsymbol{d}^0 = \begin{bmatrix}0\\0\end{bmatrix} + \alpha \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}\alpha\\\alpha\end{bmatrix}$$

Thus $\alpha_{max} = \{\alpha \mid 2\alpha \le 8, \alpha \le 10, -\alpha \le 0, -\alpha \le 0\} = 4$

We therefore solve: Minimize $f(\boldsymbol{x}^0 + \alpha \boldsymbol{d}^0)$, st $0 \le \alpha \le 4$

$\Rightarrow \alpha^* = \alpha_0 = 4 \Rightarrow \boldsymbol{x}^1 = \boldsymbol{x}^0 + \alpha_0 \boldsymbol{d}^0 = \begin{bmatrix}4\\4\end{bmatrix}$, with $f(\boldsymbol{x}^1) = -24$.

ITERATION 2 $J = \{1\}$. STEP 1 yields

Min z

st $d_1(4x_1 - 2x_2 - 4) + d_2(2x_2 - 2x_1 - 6) \le z$

$d_1 + d_2 \le z$

$-1 \le d_1, d_2 \le 1$

i.e., Min z

st $4d_1 - 6d_2 \le z$, $d_1 + d_2 \le z$, $d_1, d_2 \in [-1,1]$

yielding $z^* = -4$ (<0), with $\boldsymbol{d}^* = \boldsymbol{d}^1 = \begin{bmatrix}-1\\0\end{bmatrix}$.

STEP 2 (*line search*):

$$\boldsymbol{x}^1 + \alpha \boldsymbol{d}^1 = \begin{bmatrix}4\\4\end{bmatrix} + \alpha \begin{bmatrix}-1\\0\end{bmatrix} = \begin{bmatrix}4-\alpha\\4\end{bmatrix}$$

Thus $\alpha_{max} = \{\alpha \mid 8\text{-}\alpha \le 8,\ \alpha\text{-}4+8 \le 10,\ \alpha\text{-}4 \le 0,\ -4 \le 0\} = 4$

We therefore solve: Minimize $2(4\text{-}\alpha)^2 + 4^2 - 2(4\text{-}\alpha)4 - 4(4\text{-}\alpha) - 6(4)$, st $0 \le \alpha \le 4$.

$\Rightarrow \alpha^* = \alpha_1 = 1 \ \Rightarrow\ \boldsymbol{x^2} = \boldsymbol{x^1} + \alpha_1 \boldsymbol{d^1} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$, with $f(\boldsymbol{x^2}) = -26$.

ITERATION 3 $J = \{\phi\}$. STEP 1 yields

Min z

st $d_1(4x_1\text{-}2x_2\text{-}4) + d_2(2x_2\text{-}2x_1\text{ -}6) \le z$

$-1 \le d_1, d_2 \le 1$

i.e., Min z

st $-4d_2 \le z$, $d_1, d_2 \in [-1,1]$

yielding $z^* = -4$ (<0), with $\boldsymbol{d}^* = \boldsymbol{d^2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

STEP 2 (*line search*):

$$\boldsymbol{x^2} + \alpha \boldsymbol{d^2} = \begin{bmatrix} 3 \\ 4 \end{bmatrix} + \alpha \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 + \alpha \end{bmatrix}$$

Thus $\alpha_{max} = \{\alpha \mid 7+\alpha \le 8,\ 5+\alpha \le 10,\ -3 \le 0,\ -4\text{-}\alpha \le 0\} = 1$

We therefore solve: Minimize $f(\boldsymbol{x^2} + \alpha \boldsymbol{d^2})$, st $0 \le \alpha \le 1$

i.e., Minimize $2(3)^2 + (4+\alpha)^2 - 2(3)(4+\alpha) - 4(3) - 6(4+\alpha)$, st $0 \le \alpha \le 1$

$\Rightarrow \alpha^* = \alpha_2 = 1 \ \Rightarrow\ \boldsymbol{x^3} = \boldsymbol{x^2} + \alpha_2 \boldsymbol{d^2} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$, with $f(\boldsymbol{x^3}) = -29$.

ITERATION 4 J={1}. STEP 1 yields

Min z

st $d_1(4x_1-2x_2-4) + d_2(2x_2-2x_1-6) \le z$

$d_1 + d_2 \le z$

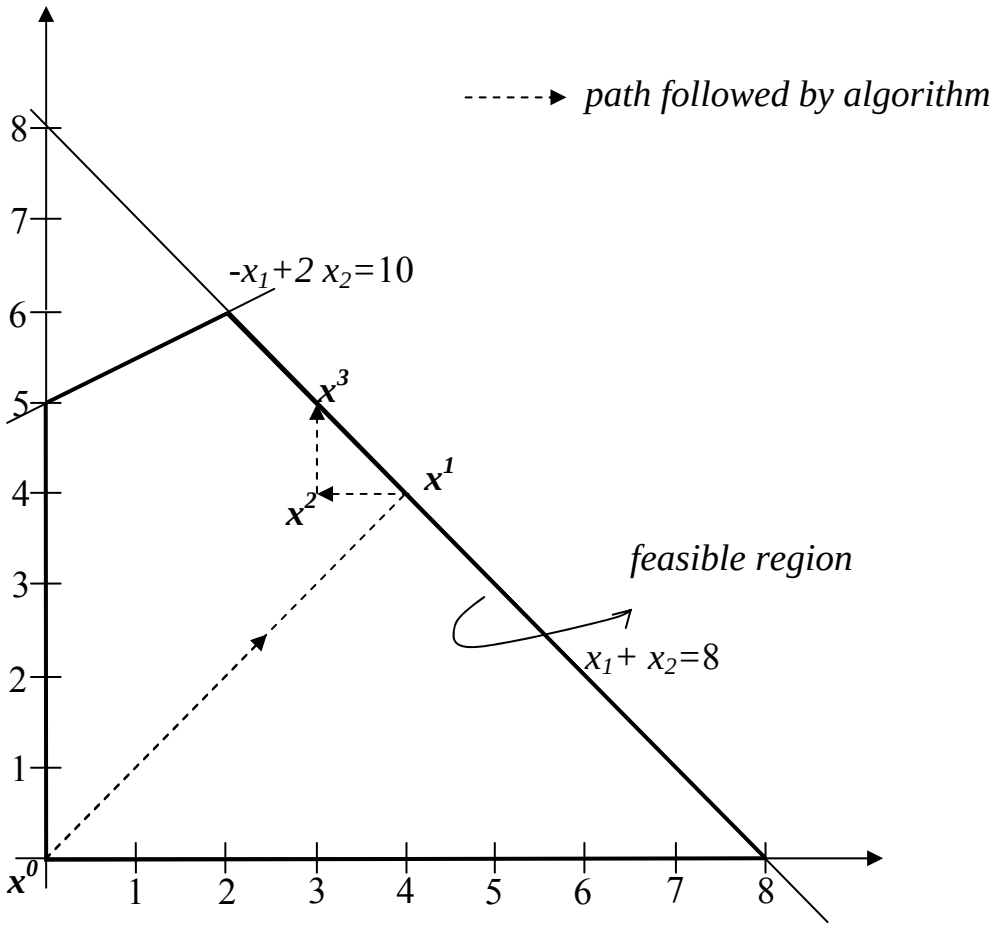
$-1 \le d_1, d_2 \le 1$

i.e., Min z

st $-2d_1 - 10d_2 \le z$, $d_1 + d_2 \le z$, $d_1, d_2 \in [-1,1]$

yielding $z^* = 0$, with $\boldsymbol{d}^* = \begin{bmatrix}0\\0\end{bmatrix}$. **STOP**

Therefore the **optimum solution** is given by $\boldsymbol{x}^* = \begin{bmatrix}3\\5\end{bmatrix}$, with $f(\boldsymbol{x}^*) = -29$.



GRADIENT PROJECTION METHOD (ROSEN)

Recall that the steepest descent direction is $-\nabla \boldsymbol{f}(\boldsymbol{x})$. However, for constrained problems, moving along $-\nabla \boldsymbol{f}(\boldsymbol{x})$ may destroy feasibility. Rosen's method works by projecting $-\nabla \boldsymbol{f}(\boldsymbol{x})$ on to the hyperplane tangent to the set of active constraints. By doing so it tries to improve the objective while simultaneously maintaining feasibility. It uses $\boldsymbol{d}=-\boldsymbol{P}\nabla \boldsymbol{f}(\boldsymbol{x})$ as the search direction, where $\boldsymbol{P}$ is a projection matrix.

Properties of $\boldsymbol{P}$ ($n \times n$ matrix)

1) $\boldsymbol{P}^{\mathrm{T}} = \boldsymbol{P}$ and $\boldsymbol{P}^{\mathrm{T}}\boldsymbol{P} = \boldsymbol{P}$ (i.e., $\boldsymbol{P}$ is **idempotent**)

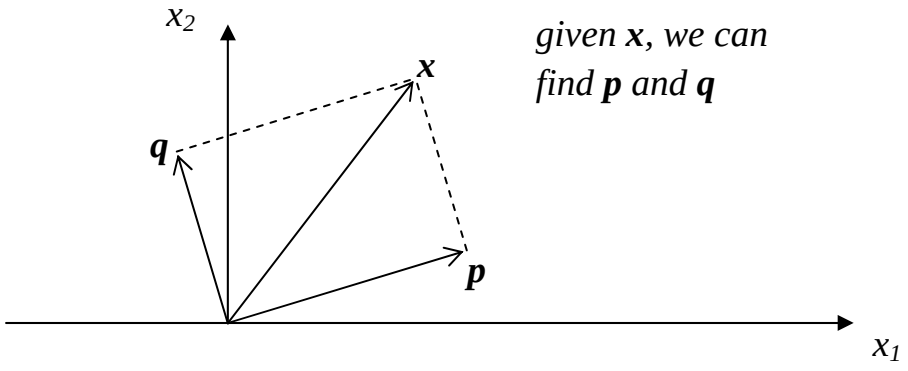
2) $\boldsymbol{P}$ is positive semidefinite

3) $\boldsymbol{P}$ is a projection matrix if, and only if, $\boldsymbol{I}-\boldsymbol{P}$ is also a projection matrix.

4) Let $\boldsymbol{Q}=\boldsymbol{I}-\boldsymbol{P}$ and $\boldsymbol{p}=\boldsymbol{P}\boldsymbol{x}^{1}$, $\boldsymbol{q}=\boldsymbol{Q}\boldsymbol{x}^{2}$ where $\boldsymbol{x}^{1}, \boldsymbol{x}^{2} \in \mathbb{R}^{n}$. Then

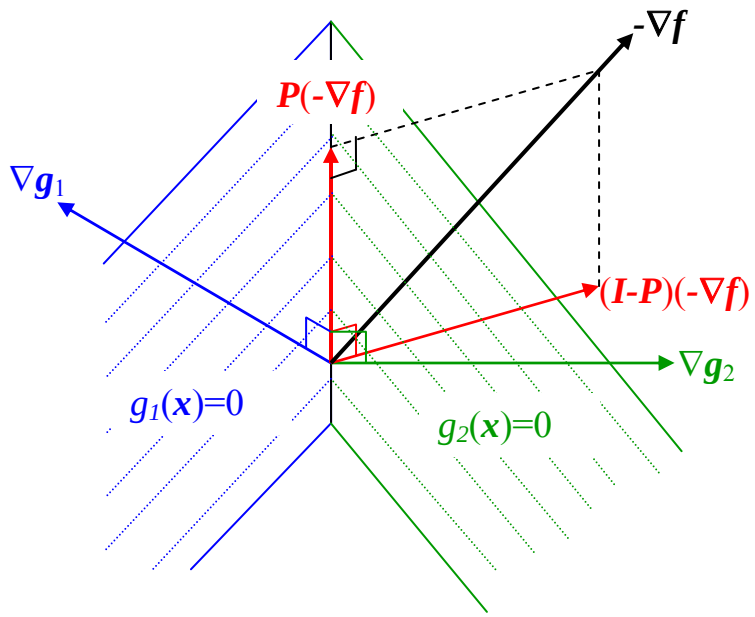
   (a) $\boldsymbol{p}^{\mathrm{T}}\boldsymbol{q}= \boldsymbol{q}^{\mathrm{T}}\boldsymbol{p} = \boldsymbol{0}$ ($\boldsymbol{p}$ and $\boldsymbol{q}$ are orthogonal)

   (b) Any $\boldsymbol{x} \in \mathbb{R}^{n}$ can be uniquely expressed as $\boldsymbol{x}=\boldsymbol{p}+\boldsymbol{q}$, where $\boldsymbol{p}=\boldsymbol{P}\boldsymbol{x}$, $\boldsymbol{q}=(\boldsymbol{I}-\boldsymbol{P})\boldsymbol{x}$



given $\boldsymbol{x}$*, we can find* $\boldsymbol{p}$ *and* $\boldsymbol{q}$

Consider the simpler case where all constraints are linear. Assume that there are 2 linear constraints intersecting along a line as shown below.



Obviously, moving along $-\nabla f$ would take us outside the feasible region. Say we project $-\nabla f$ on to the feasible region using the matrix $\boldsymbol{P}$.

THEOREM: As long as $\boldsymbol{P}$ is a projection matrix and $\boldsymbol{P}\nabla f \neq \boldsymbol{0}$, $\boldsymbol{d}=-\boldsymbol{P}\nabla f$ will be an improving direction.

PROOF: $\nabla f^{T}\boldsymbol{d} = \nabla f^{T}\cdot(-\boldsymbol{P}\nabla f) = -\nabla f^{T}\boldsymbol{P}^{T}\boldsymbol{P}\nabla f = -\left\|\boldsymbol{P}\nabla f\right\|^{2} < 0.$ (QED)

For $-\boldsymbol{P}\nabla f$ to also be feasible we must have $\nabla \boldsymbol{g_1}^{T}(-\boldsymbol{P}\nabla f)$ and $\nabla \boldsymbol{g_2}^{T}(-\boldsymbol{P}\nabla f) \leq \boldsymbol{0}$ (by definition…).

Say we pick $\boldsymbol{P}$ such that $\nabla \boldsymbol{g_1}^T(-\boldsymbol{P}\nabla \boldsymbol{f}) = \nabla \boldsymbol{g_2}^T(-\boldsymbol{P}\nabla \boldsymbol{f}) = 0$, so that $-\boldsymbol{P}\nabla \boldsymbol{f}$ is a feasible direction). Thus if we denote $\boldsymbol{M}$ as the $(2\times n)$ matrix where Row 1 is $\nabla \boldsymbol{g_1}^T$ and Row 2 is $\nabla \boldsymbol{g_2}^T$, then we have $\boldsymbol{M}(-\boldsymbol{P}\nabla \boldsymbol{f}) = \boldsymbol{0}$ (*)

To find the form of the matrix $\boldsymbol{P}$, we make use of Property 4(b) to write the vector $-\nabla \boldsymbol{f}$ as $-\nabla \boldsymbol{f} = \boldsymbol{P}(-\nabla \boldsymbol{f}) + [\boldsymbol{I}-\boldsymbol{P}](-\nabla \boldsymbol{f}) = -\boldsymbol{P}\nabla \boldsymbol{f} - [\boldsymbol{I}-\boldsymbol{P}]\nabla \boldsymbol{f}$

Now, $-\nabla \boldsymbol{g_1}$ and $-\nabla \boldsymbol{g_2}$ are both orthogonal to $-\boldsymbol{P}\nabla \boldsymbol{f}$, and so is $[\boldsymbol{I}-\boldsymbol{P}](-\nabla \boldsymbol{f})$. Hence we must have

$$[\boldsymbol{I}-\boldsymbol{P}](-\nabla \boldsymbol{f}) = -[\boldsymbol{I}-\boldsymbol{P}]\nabla \boldsymbol{f} = \lambda_1 \nabla \boldsymbol{g_1} + \lambda_2 \nabla \boldsymbol{g_2}$$

$$\Rightarrow -[\boldsymbol{I}-\boldsymbol{P}]\nabla \boldsymbol{f} = \boldsymbol{M}^T \boldsymbol{\lambda}, \quad \text{where } \boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}$$

$$\Rightarrow -\boldsymbol{M}[\boldsymbol{I}-\boldsymbol{P}]\nabla \boldsymbol{f} = \boldsymbol{M}\boldsymbol{M}^T \boldsymbol{\lambda}$$

$$\Rightarrow -(\boldsymbol{M}\boldsymbol{M}^T)^{-1}\boldsymbol{M}[\boldsymbol{I}-\boldsymbol{P}]\nabla \boldsymbol{f} = (\boldsymbol{M}\boldsymbol{M}^T)^{-1}\boldsymbol{M}\boldsymbol{M}^T \boldsymbol{\lambda} = \boldsymbol{\lambda}$$

$$\Rightarrow \lambda = -\{ (\boldsymbol{M}\boldsymbol{M}^T)^{-1}\boldsymbol{M}\nabla \boldsymbol{f} - (\boldsymbol{M}\boldsymbol{M}^T)^{-1}\boldsymbol{M}\boldsymbol{P}\nabla \boldsymbol{f} \}$$

$$\Rightarrow \lambda = -\{ (\boldsymbol{M}\boldsymbol{M}^T)^{-1}\boldsymbol{M}\nabla \boldsymbol{f} + (\boldsymbol{M}\boldsymbol{M}^T)^{-1}\boldsymbol{M}(-\boldsymbol{P}\nabla \boldsymbol{f}) \}$$

$$\Rightarrow \lambda = -(\boldsymbol{M}\boldsymbol{M}^T)^{-1}\boldsymbol{M}\nabla \boldsymbol{f} \qquad \text{(from (*) above } \boldsymbol{M}(-\boldsymbol{P}\nabla \boldsymbol{f})=0\ldots)$$

Hence we have

$$-\nabla \boldsymbol{f} = -\boldsymbol{P}\nabla \boldsymbol{f} - [\boldsymbol{I}-\boldsymbol{P}]\nabla \boldsymbol{f} = -\boldsymbol{P}\nabla \boldsymbol{f} + \boldsymbol{M}^T\lambda = -\boldsymbol{P}\nabla \boldsymbol{f} + \boldsymbol{M}^T(\boldsymbol{M}\boldsymbol{M}^T)^{-1}\boldsymbol{M}\nabla \boldsymbol{f}$$

i.e., $\boldsymbol{P}\nabla \boldsymbol{f} = \nabla \boldsymbol{f} - \boldsymbol{M}^T(\boldsymbol{M}\boldsymbol{M}^T)^{-1}\boldsymbol{M}\nabla \boldsymbol{f} = [\boldsymbol{I} - \boldsymbol{M}^T(\boldsymbol{M}\boldsymbol{M}^T)^{-1}\boldsymbol{M}]\nabla \boldsymbol{f}$

i.e., $$\boxed{\boldsymbol{P} = [\boldsymbol{I} - \boldsymbol{M}^T(\boldsymbol{M}\boldsymbol{M}^T)^{-1}\boldsymbol{M}]}$$

Question: What if $\boldsymbol{P}\nabla\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$?

Answer: Then $\boldsymbol{0} = \boldsymbol{P}\nabla\boldsymbol{f} = [\boldsymbol{I} - \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{M}\boldsymbol{M}^{\mathrm{T}})^{-1}\boldsymbol{M}]\,\nabla\boldsymbol{f} = \nabla\boldsymbol{f} + \boldsymbol{M}^{\mathrm{T}}\boldsymbol{w}$, where $\boldsymbol{w} = -(\boldsymbol{M}\boldsymbol{M}^{\mathrm{T}})^{-1}\boldsymbol{M}\nabla\boldsymbol{f}$. If $\boldsymbol{w} \geq \boldsymbol{0}$, then $\boldsymbol{x}$ satisfies the Karush-Kuhn-Tucker conditions and we may stop; if not, a new projection matrix $\widehat{\boldsymbol{P}}$ could be identified such that $\boldsymbol{d} = -\widehat{\boldsymbol{P}}\nabla\boldsymbol{f}$ is an improving feasible direction. In order to identify this matrix $\widehat{\boldsymbol{P}}$ we merely pick any component of $\boldsymbol{w}$ that is negative, and drop the corresponding row from the matrix $\boldsymbol{M}$. Then we use the usual formula to obtain $\widehat{\boldsymbol{P}}$.

==========

Now consider the general problem

Minimize $f(\boldsymbol{x})$

st $g_j(\boldsymbol{x}) \leq 0,\ j=1,2,\ldots,m$

$h_k(\boldsymbol{x}) = 0,\ k=1,2,\ldots,p$

The gradient projection algorithm can be generalized to nonlinear constraints by projecting the gradient on to the tangent hyperplane at $\boldsymbol{x}$ (rather than on to the surface of the feasible region itself).

STEP 1: **SEARCH DIRECTION**

Let $\boldsymbol{x}^i$ be a feasible point and let $\boldsymbol{J}$ be the "active set", i.e., $\boldsymbol{J} = \{j \mid g_j(\boldsymbol{x}^i)=0\}$, or more practically, $\boldsymbol{J} = \{j \mid g_j(\boldsymbol{x}^i)+\varepsilon \geq 0\}$.

Suppose each $j \in \boldsymbol{J}$ is approximated using the Taylor series expansion by:

$$g_j(\boldsymbol{x}) = g_j(\boldsymbol{x}^i) + (\boldsymbol{x}-\boldsymbol{x}^i)\nabla g_j(\boldsymbol{x}^i).$$

Notice that this approximation is a linear function of $\boldsymbol{x}$ with slope $\nabla \boldsymbol{g}_j(\boldsymbol{x}^i)$.

- Let $\boldsymbol{M} \equiv$ matrix whose rows are $\nabla \boldsymbol{g}_j^{\mathrm{T}}(\boldsymbol{x}^i)$ for $j \in \boldsymbol{J}$ (active constraints) and $\nabla \boldsymbol{h}_k^{\mathrm{T}}(\boldsymbol{x}^i)$ for $k=1,2,...,p$ (equality constraints)
- Let let $\boldsymbol{P}=[\boldsymbol{I} - \boldsymbol{M}^{\mathrm{T}}(\boldsymbol{M}\boldsymbol{M}^{\mathrm{T}})^{-1}\boldsymbol{M}]$ ($\equiv$projection matrix). If $\boldsymbol{M}$ does not exist, let $\boldsymbol{P}=\boldsymbol{I}$.
- Let $\boldsymbol{d}^i = -\boldsymbol{P}\nabla \boldsymbol{f}(\boldsymbol{x}^i)$.

a) If $\boldsymbol{d}^i = \boldsymbol{0}$ and $\boldsymbol{M}$ does not exist STOP

b) If $\boldsymbol{d}^i = \boldsymbol{0}$ and $\boldsymbol{M}$ exists find

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix} = -(\boldsymbol{M}\boldsymbol{M}^{\mathrm{T}})^{-1}\boldsymbol{M}\nabla \boldsymbol{f}(\boldsymbol{x}^i)$$

where $\boldsymbol{u}$ corresponds to $\boldsymbol{g}$ and $\boldsymbol{v}$ to $\boldsymbol{h}$. If $\boldsymbol{u} > \boldsymbol{0}$ STOP, else delete the row of $\boldsymbol{M}$ corresponding to some $u_j < 0$ and return to Step 1.

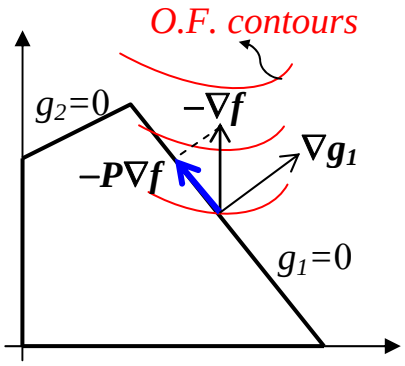
c) If $\boldsymbol{d}^i \neq \boldsymbol{0}$ go to Step 2.

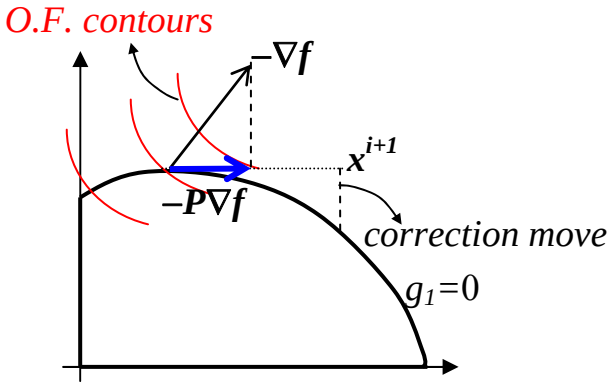
STEP 2. **STEP SIZE** (Line Search)

Solve $\underset{0\le\lambda\le\lambda_{max}}{\text{Minimize}}\, f\left(x^i + \lambda d^i\right)$

where λ_{max}= Supremum $\{\lambda \mid g_j(\boldsymbol{x}^i+\lambda\boldsymbol{d}^i) \le 0,\ h_j(\boldsymbol{x}^i+\lambda\boldsymbol{d}^i) =0\}$.

Call the optimal solution $\lambda^*$and find $\boldsymbol{x}^{i+1} = \boldsymbol{x}^i+\lambda^*\boldsymbol{d}^i$. If all constraints are not linear make a "correction move" to return $\boldsymbol{x}$ into the feasible region. The need for this is shown below. Generally, it could be done by solving $\boldsymbol{g}(\boldsymbol{x})=\boldsymbol{0}$ starting at $\boldsymbol{x}^{i+1}$ using the Newton Raphson approach, or by moving orthogonally from $\boldsymbol{x}^{i+1}$ etc.





g_1, g_2 are linear constraints

NO CORRECTION REQD.

g_1 is a nonlinear constraint

REQUIRES CORRECTION

LINEARIZATION METHODS

The general idea of linearization methods is to replace the solution of the NLP by the solution of a series of linear programs which approximate the original NLP.

The simplest method is **RECURSIVE LP**:

Transform Min $f(\boldsymbol{x})$, st $g_j(\boldsymbol{x})$, $j=1,2,...,m$,

into the LP

$$\text{Min} \quad \{f(\boldsymbol{x}^i) + (\boldsymbol{x}-\boldsymbol{x}^i)^T \nabla \boldsymbol{f}(\boldsymbol{x}^i)\}$$

$$\text{st} \quad g_j(\boldsymbol{x}^i) + (\boldsymbol{x}-\boldsymbol{x}^i)^T \nabla \boldsymbol{g_j}(\boldsymbol{x}^i) \leq 0, \text{ for all } j.$$

Thus we start with some initial $\boldsymbol{x}^i$ where the objective and constraints (usually only the tight ones) are linearized, and then solve the LP to obtain a new $\boldsymbol{x}^{i+1}$ and continue...

Note that $f(\boldsymbol{x}^i)$ is a constant and $\boldsymbol{x}-\boldsymbol{x}^i=\boldsymbol{d}$ implies the problem is equivalent to

$$\text{Min} \quad \boldsymbol{d}^T \nabla \boldsymbol{f}(\boldsymbol{x}^i)$$

$$\text{st} \quad \boldsymbol{d}^T \nabla \boldsymbol{g_j}(\boldsymbol{x}^i) \leq -\boldsymbol{g_j}(\boldsymbol{x}^i), \text{ for all } j.$$

and this is a direction finding LP problem, and $\boldsymbol{x}^{i+1} = \boldsymbol{x}^i + \boldsymbol{d}$ implies that the step size=1.0!

EXAMPLE: Min $f(\boldsymbol{x}) = 4x_1 - x_2^2 - 12$,

st $g_1(\boldsymbol{x}) = x_1^2 + x_2^2 - 25 \leq 0$,

$g_2(\boldsymbol{x}) = -x_1^2 - x_2^2 + 10x_1 + 10x_2 - 34 \geq 0$

If this is linearized at the point $\boldsymbol{x}^i = (2,4)$, then since

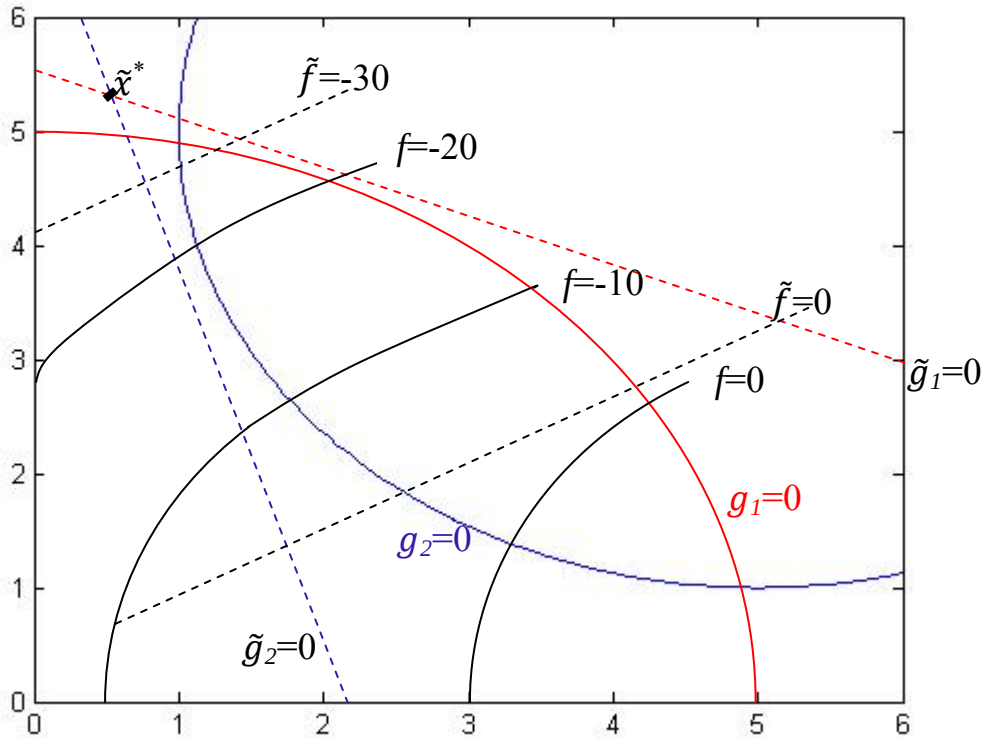
$$\nabla f(\boldsymbol{x^i}) = \begin{bmatrix} 4 \\ -2x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ -8 \end{bmatrix}; \nabla g_1(\boldsymbol{x^i}) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 8 \end{bmatrix}; \nabla g_2(\boldsymbol{x^i}) = \begin{bmatrix} -2x_1 + 10 \\ -2x_2 + 10 \end{bmatrix} = \begin{bmatrix} 6 \\ 2 \end{bmatrix}$$

it yields the following linear program (VERIFY…)

$$\text{Min } \tilde{f}(\boldsymbol{x}) = 4x_1 - 8x_2 + 4,$$

$$\text{st} \quad \tilde{g}_1(\boldsymbol{x}) = 4x_1 + 8x_2 - 45 \le 0,$$

$$\tilde{g}_2(\boldsymbol{x}) = 6x_1 + 2x_2 - 14 \ge 0$$



The method however has limited application since it does not converge if the local minimum occurs at a point that is not a vertex of the feasible region; in such cases it oscillates between adjacent vertices. To avoid this one may limit the step size (for instance, the **Frank-Wolfe** method minimizes f between $\boldsymbol{x^i}$ and $\boldsymbol{x^i}+\boldsymbol{d}$ to get $\boldsymbol{x^{i+1}}$).

RECURSIVE QUADRATIC PROGRAMMING (RQP)

RQP is similar in logic to RLP, except that it solves a sequence of quadratic programming problems. RQP is a better approach because the second order terms help capture **curvature information**.

$$\text{Min} \quad \boldsymbol{d}^{\mathrm{T}}\nabla \boldsymbol{L}(\boldsymbol{x}^{\boldsymbol{i}}) + \tfrac{1}{2}\, \boldsymbol{d}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{d}$$

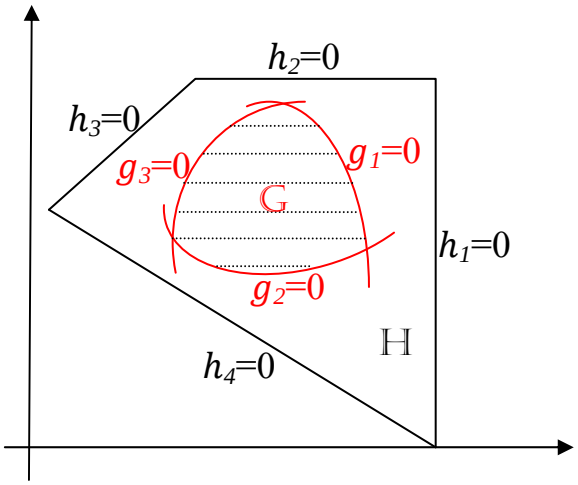
$$\text{st } \boldsymbol{d}^{\mathrm{T}}\nabla \boldsymbol{g_j}\,(\boldsymbol{x}^{\boldsymbol{i}}) \le -g_j(\boldsymbol{x}^{\boldsymbol{i}}) \;\forall\; j$$

where $\boldsymbol{B}$ is the Hessian (or an approximation of the Hessian) of the Lagrangian function of the objective function f at the point $\boldsymbol{x}^{\boldsymbol{i}}$. Notice that this a QP in $\boldsymbol{d}$ with linear constraints. Once the above QP is solved to get $\boldsymbol{d}^{\boldsymbol{i}}$, we obtain $\boldsymbol{x}^{\boldsymbol{i}}$ by finding an optimum step size along $\boldsymbol{x}^{\boldsymbol{i}}+\boldsymbol{d}^{\boldsymbol{i}}$ and continue. The matrix $\boldsymbol{B}$ is never explicitly computed --- it is usually updated by schemes such as the BFGS approach using information from $\nabla \boldsymbol{f}(\boldsymbol{x}^{\boldsymbol{i}})$ and $\nabla \boldsymbol{f}(\boldsymbol{x}^{\boldsymbol{i+1}})$ only. (Recall the quasi-Newton methods for unconstrained optimization)

KELLEY'S CUTTING PLANE METHOD

One of the better linearization methods is the cutting plane algorithm of J.E.Kelley developed for convex programming problems.

Suppose we have a set of nonlinear constraints $g_j(\boldsymbol{x})\leq 0$ for $j=1,2,...,m$ that determine the feasible region $\mathbb{G}$. Suppose further that we can find a polyhedral set $\mathbb{H}$ that entirely contains the region determined by these constraints, i.e., $\mathbb{G}\subseteq\mathbb{H}$.



$\mathbb{G}$: *feasible region*

$g_j(\boldsymbol{x})$: *nonlinear*

$h_j(\boldsymbol{x})$: *linear; hence* $\mathbb{H}$ *is polyhedral*

In general, a cutting plane algorithm would operate as follows:

1. Solve the problem with linear constraints.
2. If the optimal solution to this problem is feasible in the original (nonlinear) constraints then STOP; it is also optimal for the original problem.
3. Otherwise add an extra linear constraint (i.e., an extra line/hyperplane) so that the current optimal point becomes infeasible for the new problem with the additional constraint (we thus "cut out" part of the infeasibility).

   Now return to Step 1.

JUSTIFICATION

Step 1: It is in general easier to solve problems with linear constraints than ones with nonlinear constraints.

Step 2: $\mathbb{G}$ is determined by $g_j(\boldsymbol{x})\leq 0$, $j=1,...,m$, where each g_j is convex, while $\mathbb{H}$ is determined by $h_k\leq 0\ \forall$ k, where each h_k is linear. $\mathbb{G}$ is wholly contained inside the polyhedral set $\mathbb{H}$.

Thus every point that satisfies $g_j(\boldsymbol{x})\leq 0$ automatically satisfies $h_k(\boldsymbol{x})\leq 0$, but not vice-versa. Thus $\mathbb{G}$ places "extra" constraints on the design variables and therefore the optimal solution with $\mathbb{G}$ can never be any better than the optimal solution with $\mathbb{H}$. Therefore, if for some region $\mathbb{H}$, the optimal solution $\boldsymbol{x}^*$ is also feasible in $\mathbb{G}$, it MUST be optimal for $\mathbb{G}$ also. (NOTE: In general, the optimal solution for $\mathbb{H}$ at some iteration will not be feasible in $\mathbb{G}$).

Step 3: Generating a "CUT"

In Step 2, let $\bar{g}(\boldsymbol{x^i}) = \text{Maximum}_{j\in J}\, g_j(\boldsymbol{x^i})$, where J is the set of violated constraints, $J=\{j \mid g_j(\boldsymbol{x})>0\}$, i.e., $\bar{g}(\boldsymbol{x^i})$ is the value of the most violated constraint at $\boldsymbol{x^i}$. By the Taylor series approximation around $\boldsymbol{x^i}$,

$$\bar{g}(\boldsymbol{x}) = \bar{g}(\boldsymbol{x^i}) + (\boldsymbol{x}-\boldsymbol{x^i})^{\mathrm{T}}\boldsymbol{\nabla}\bar{\boldsymbol{g}}(\boldsymbol{x^i})+\boldsymbol{\ldots}$$

$$\Rightarrow \quad \bar{g}(\boldsymbol{x}) \geq \bar{g}(\boldsymbol{x^i}) + (\boldsymbol{x}-\boldsymbol{x^i})^{\mathrm{T}}\boldsymbol{\nabla}\bar{\boldsymbol{g}}(\boldsymbol{x^i}),\ \text{for every } \boldsymbol{x}\in\mathbb{R}^n \text{ (since } \bar{g} \text{ is convex).}$$

If $\nabla\bar{g}(x^i) = 0$ then $\bar{g}(x) \geq \bar{g}(x^i)$

But since $\bar{g}$ is violated at x^i, $\bar{g}(x^i) > 0$. Therefore $\bar{g}(x) > 0 \; \forall \; x \in \mathbb{R}^n$.

$\Rightarrow$ $\bar{g}$ is violated for <u>all</u> x,

$\Rightarrow$ the original problem is infeasible.

If $\nabla\bar{g}(x^i) \neq 0$, then consider the following linear constraint:

$$(x - x^i)^T \nabla\bar{g}(x^i) \leq -\bar{g}(x^i)$$

i.e., $h_k(x) = \bar{g}(x^i) + (x - x^i)^T \nabla\bar{g}(x^i) \leq 0$ (⊗)

(Note that x^i, $\nabla\bar{g}(x^i)$ and $\bar{g}(x^i)$ are all known…)

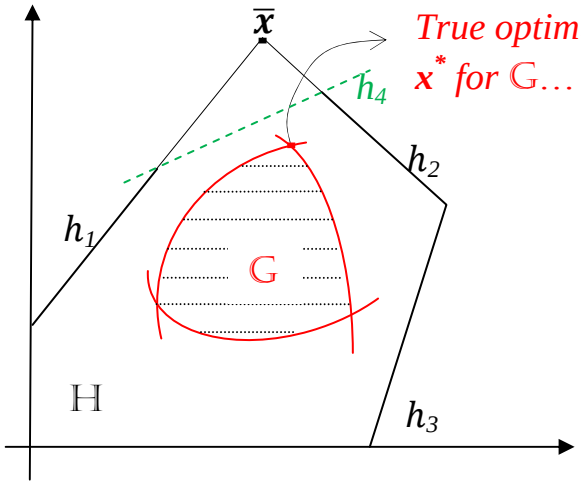
The current point x^i is obviously infeasible in this constraint --- since $h_k(x^i) \leq 0$ implies that $\bar{g}(x^i) + (x^i - x^i)^T \nabla\bar{g}(x^i) = \bar{g}(x^i) \leq 0$, which contradicts the fact that $\bar{g}$ is violated at x^i , (i.e, $\bar{g}(x^i) > 0$).

So, if we add this extra linear constraint, then the optimal solution would change because the current optimal solution would no longer be feasible for the new LP with the added constraint. Hence (⊗) defines a suitable cutting plane.

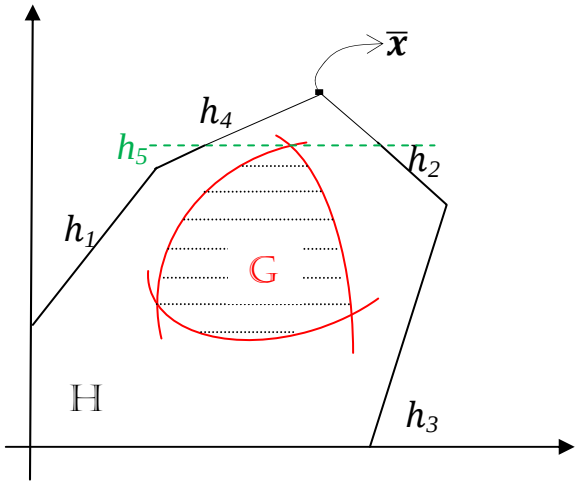
<u>NOTE</u>: If the objective for the original NLP was linear, then at each iteration we merely solve a linear program!

An Example (in $\mathbb{R}^2$) of how a cutting plane algorithm might proceed.

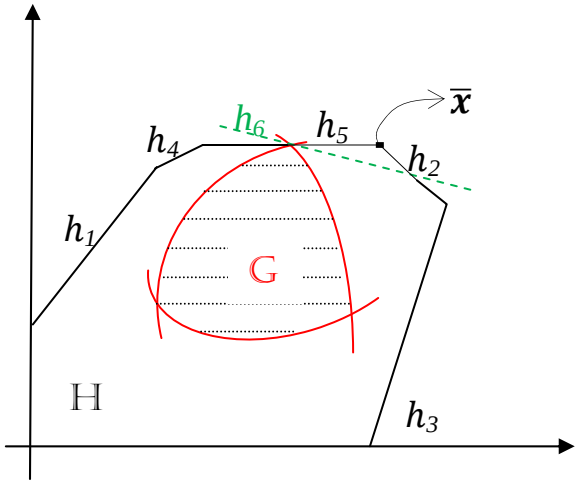
(--------- cutting plane).



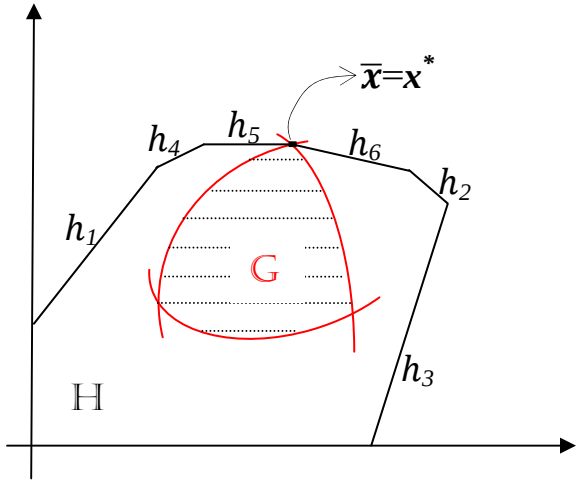
Nonnegativity + 3 linear constraints



Nonnegativity + 4 linear constraints



Nonnegativity + 5 linear constraints



Nonnegativity + 6 linear constraints