Constrained Optimization: Direct Methods

AE 6310: Optimization for the Design of Engineered Systems

Spring 2017

Dr. Glenn Lightsey

Lecture Notes Developed By Dr. Brian German

Georgia Institute of Technology

Daniel Guggenheim School of Aerospace Engineering

Direct Methods

As the name implies, “direct methods” of constrained optimization deal with the constraints “directly”, i.e. without incorporating them into a pseudo-objective function.

For line search methods, the implication is that search directions are based on considerations of the constraint geometry or gradients.

Direct Methods

We will examine the following types of direct methods in this series of lectures:

- Sequential linear programming
- Method of feasible directions
- Generalized reduced gradient method
- Sequential quadratic programming

Direct Methods

Some authors, including Vanderplaats, consider metaheuristic and stochastic algorithms including,

- Genetic algorithms
- Particle swarm algorithms

to be direct methods for constrained optimization.

These methods *may* account for the constraints directly but in rather unsophisticated ways. The methods are also very different from calculus-based approaches. We will therefore consider them separately in subsequent lectures.

Sequential Linear Programming

As its name implies, sequential linear programming (SLP) repeatedly solves linear programming problems to march toward a solution to a constrained nonlinear optimization problem.

Sequential Linear Programming

The general approach of sequential linear programming is as follows:

1. Linearize the optimization problem around a baseline point in the design space to obtain a linear program (LP).
2. Add additional side constraints called “move limits” around the baseline point to prevent the LP from problem from being unbounded (solutions at $\pm\infty$ for some of the design variables)
3. Solve the LP with any appropriate technique such as simplex.
4. Set the new baseline point as the optimum found in step 3.
5. Repeat steps 1 to 4 until a convergence criterion is met.

Sequential Linear Programming

Consider the original constrained optimization problem:

Minimize $f(\boldsymbol{x})$

Subject to:

$\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}$	Inequality constraints
$\boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}$	Equality constraints
$\boldsymbol{x_L} \leq \boldsymbol{x} \leq \boldsymbol{x_U}$	Side constraints

Sequential Linear Programming

Sequential linear programming linearizes the problem as:

Minimize $f(\boldsymbol{x}) \approx f(\boldsymbol{x_0}) + \nabla f(\boldsymbol{x_0})^T \boldsymbol{\delta x}$

Subject to:

$$g_j(\boldsymbol{x}) \approx g_j(\boldsymbol{x_0}) + \nabla g_j(\boldsymbol{x_0})^T \boldsymbol{\delta x} \leq 0$$

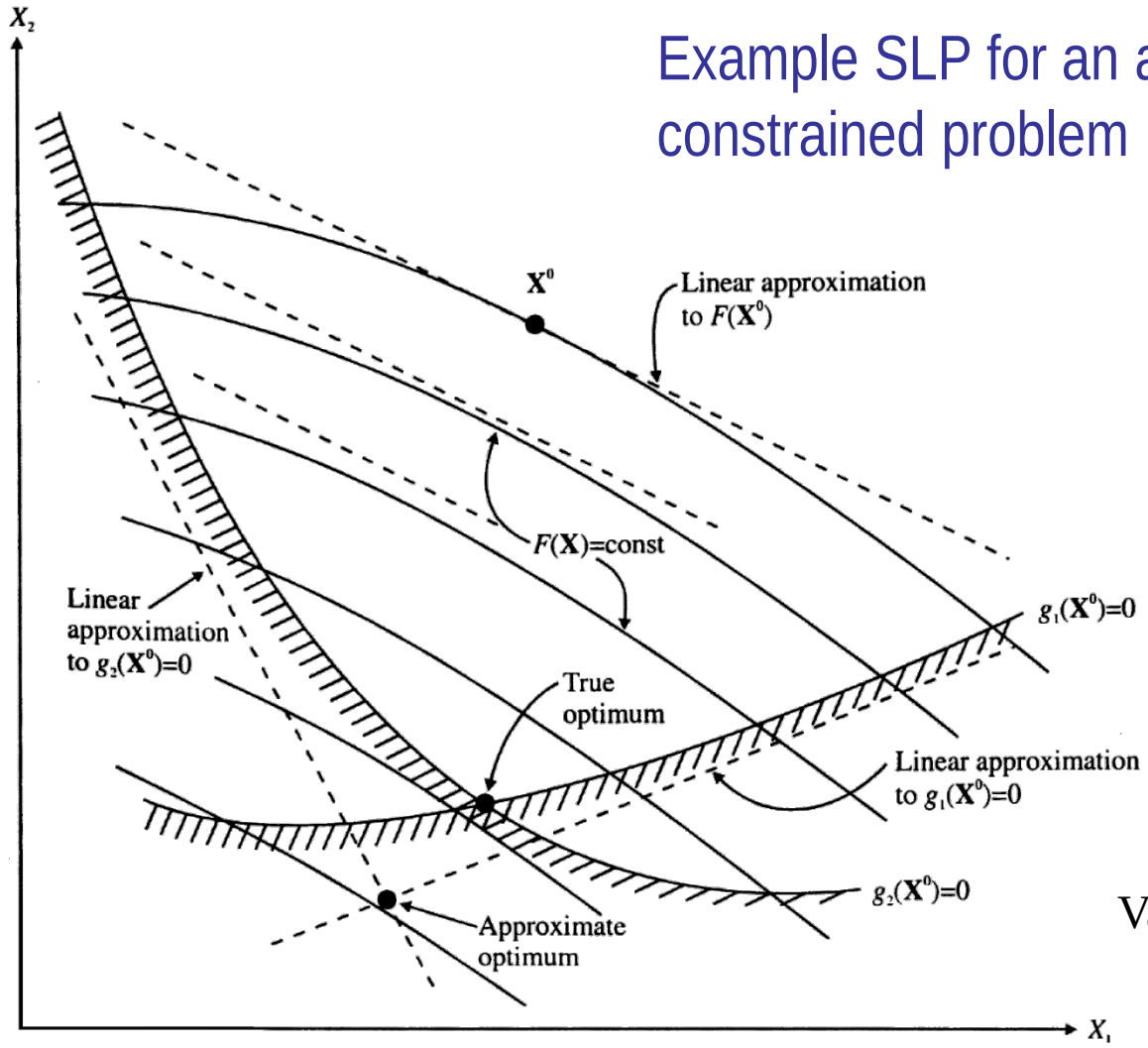
$$h_k(\boldsymbol{x}) \approx h_k(\boldsymbol{x_0}) + \nabla h_k(\boldsymbol{x_0})^T \boldsymbol{\delta x} = 0$$

$$x_{i,L} \leq x_{0,\,i} + \delta x_i \leq x_{i,U}$$

where $\boldsymbol{\delta x} = \boldsymbol{x} - \boldsymbol{x_0}$.

Sequential Linear Programming

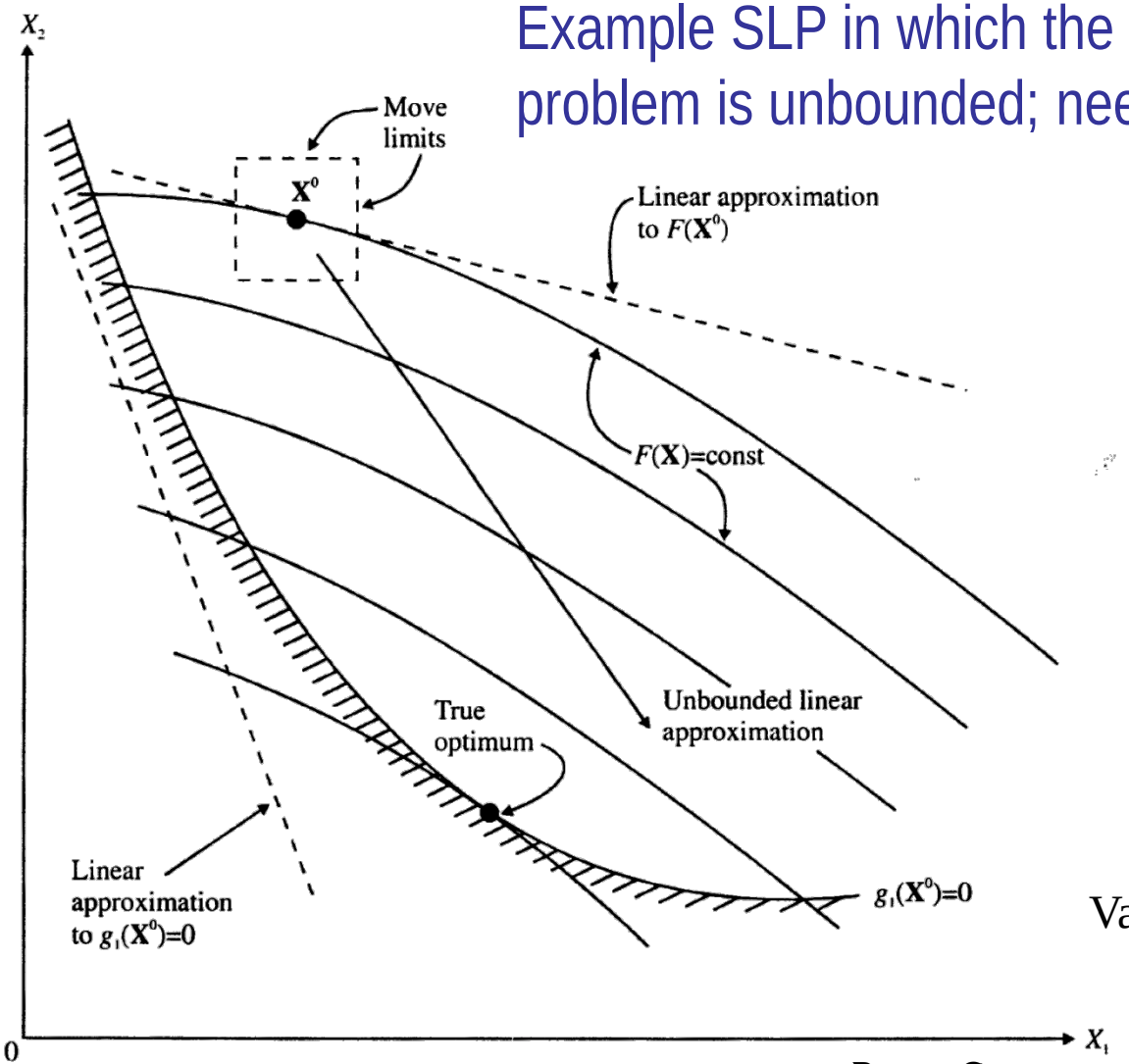
Example SLP for an adequately constrained problem



Vanderplaats, Fig. 6.1

Sequential Linear Programming

Example SLP in which the linearized problem is unbounded; need for move limits



Vanderplaats, Fig. 6.2

Sequential Linear Programming

Move limits can be implemented by adding additional side constraints to the problem that are more restrictive and localized than the side constraints for the original problem:

Overall side constraints: $x_{i,L} \leq x_{0,\,i} + \delta x_i \leq x_{i,U}$

Move limit side constraints:

$$x_{0,\,i} - \delta x_{i,L} \leq x_{0,\,i} + \delta x_i \leq x_{0,\,i} + \delta x_{i,U}$$

where $\delta x_{i,L} > 0$ and $\delta x_{i,U} > 0$ are "small" changes.

Values of $\delta x_{i,L}$ and $\delta x_{i,U}$ may be reduced each iteration.

Method of Feasible Directions

The method of feasible directions (MoFD) is a line search method in which the search direction is determined in a way to ensure that it is…

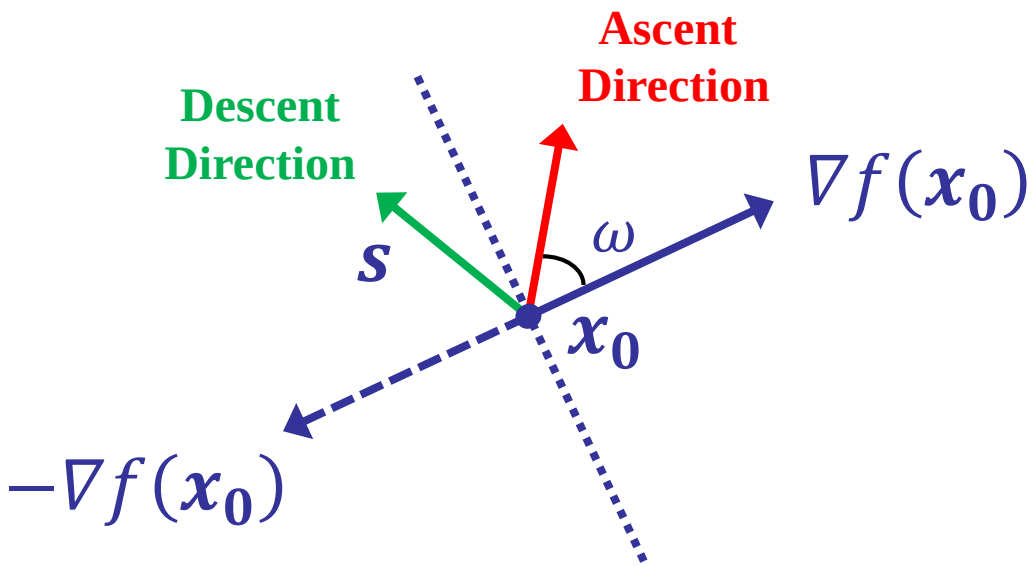
Usable, i.e. it decreases the objective function

and

Feasible, i.e. it moves away from or along the active constraints within the feasible region

Usable Directions

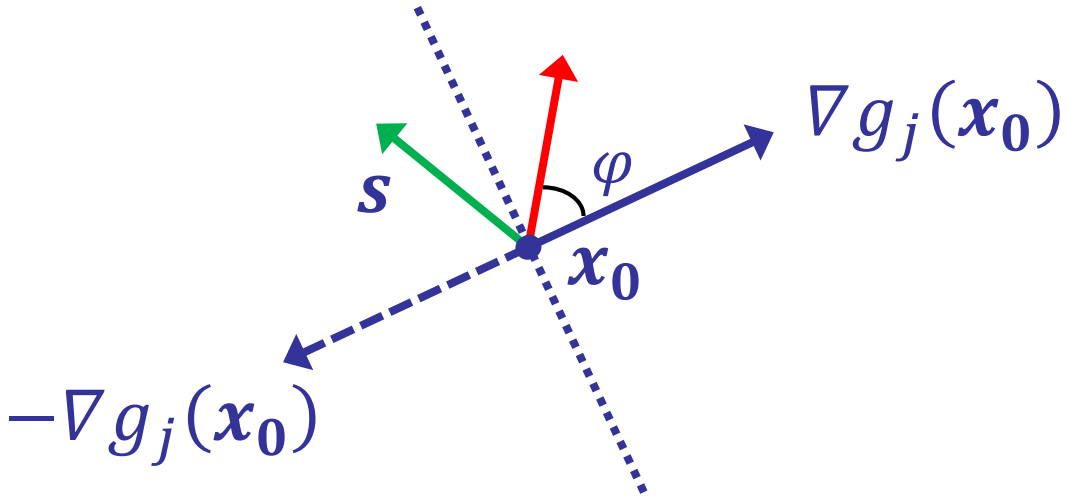
Usable directions are *descent directions.*



A usable direction points in a direction $90° < \omega < 270°$ from $\nabla f(\boldsymbol{x_0})$, i.e. it forms an acute angle with $-\nabla f(\boldsymbol{x_0})$.

Feasible Directions

Recall that we require $g_j(\boldsymbol{x}) \leq 0$ for feasibility. If a constraint *j* is active at a particular point ($g_j(\boldsymbol{x_0}) = 0$), then MoFD requires that our next move be a *feasible direction:*



A direction that is feasible with respect to a particular inequality constraint *j* forms a $90° < \varphi < 270°$ angle from $\nabla g(\boldsymbol{x_0})$, i.e. it forms an acute angle with $-\nabla g(\boldsymbol{x_0})$.

Usable and Feasible Directions

We can write the requirements for usability and feasibility as follows:

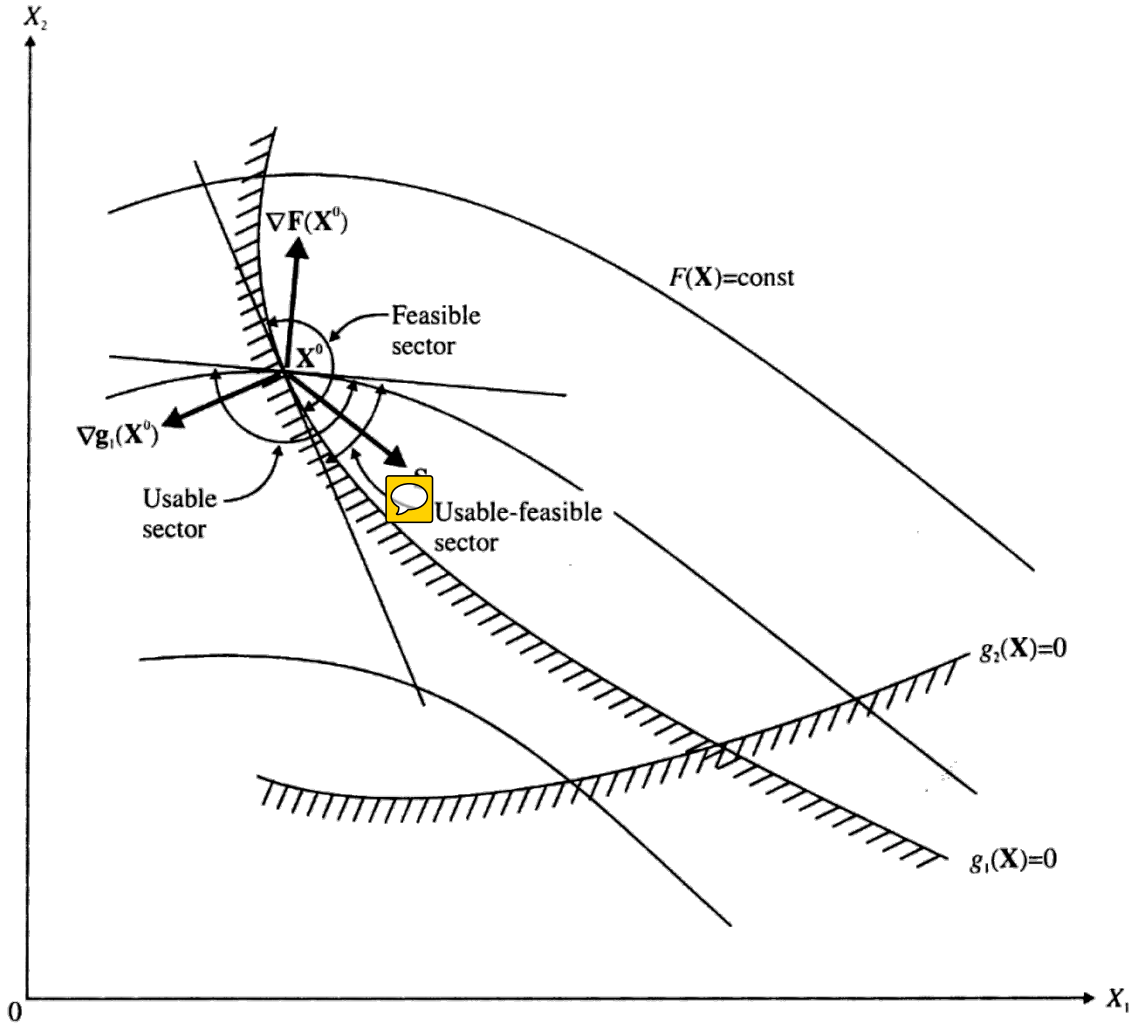
Usability:

$$\nabla f(\boldsymbol{x_0})^T \boldsymbol{s} = \|\nabla f(\boldsymbol{x_0})\| \, \|\boldsymbol{s}\| \cos\omega \leq 0$$

Feasibility:

$$\nabla g_j(\boldsymbol{x_0})^T \boldsymbol{s} = \left\|\nabla g_j(\boldsymbol{x_0})\right\| \|\boldsymbol{s}\| \cos\phi \leq 0$$

Usable and Feasible Directions



Vanderplaats, Fig. 6.4

MoFD: Direction Finding

The "fastest feasible improvement" could be achieved if we could select the search direction $\boldsymbol{s}$ such that

$$\nabla f(\boldsymbol{x_0})^T \boldsymbol{s}$$

is "as negative possible without violating the constraints." This would imply that the search direction would form as small of an angle as possible with $-\nabla f(\boldsymbol{x_0})$.

Recall that for the unconstrained case, this would imply the "steepest descent" criterion, i.e. $\boldsymbol{s} \sim -\nabla f(\boldsymbol{x_0})$.

MoFD: Direction Finding

The direction of "fastest feasible improvement" can be formulated as a "direction-finding" optimization problem:

Maximize: β

Subject to: $\nabla f(\boldsymbol{x_0})^T \boldsymbol{s} + \beta \leq 0$

Additional inequality constraints based on the constraints $g_j(\boldsymbol{x_0})$

What is the form of these additional constraints?

MoFD: Direction Finding

The direction of "fastest feasible improvement" is often in a direction tangent to one or more of the active constraints.

This implies that such a direction satisfies

$$\nabla g_j(\boldsymbol{x_0})^T \boldsymbol{s} = 0$$

for those active constraints.

The problem with searching in such a direction is that if the constraint has any curvature, our search direction may move into the infeasible region.

MoFD: Direction Finding

To avoid this problem, MoFD specifies that the search direction obey the following requirement for all active constraints:

$$\nabla g_j(\boldsymbol{x_0})^T \boldsymbol{s} + \beta\theta_j \leq 0$$

$\theta_j > 0$ is a constant with "small" magnitude called the **push-off factor**. Since $\beta > 0$ for usable directions, the 2nd term is positive.

The push-off factor ensures that the angle between $\nabla g_j(\boldsymbol{x_0})$ and $\boldsymbol{s}$ is greater than $90°$ with some buffer for constraint curvature.

MoFD: Direction Finding

Revisiting our direction-finding problem:

Maximize: β

Subject to: $\nabla f(\boldsymbol{x_0})^T \boldsymbol{s} + \beta \leq 0$

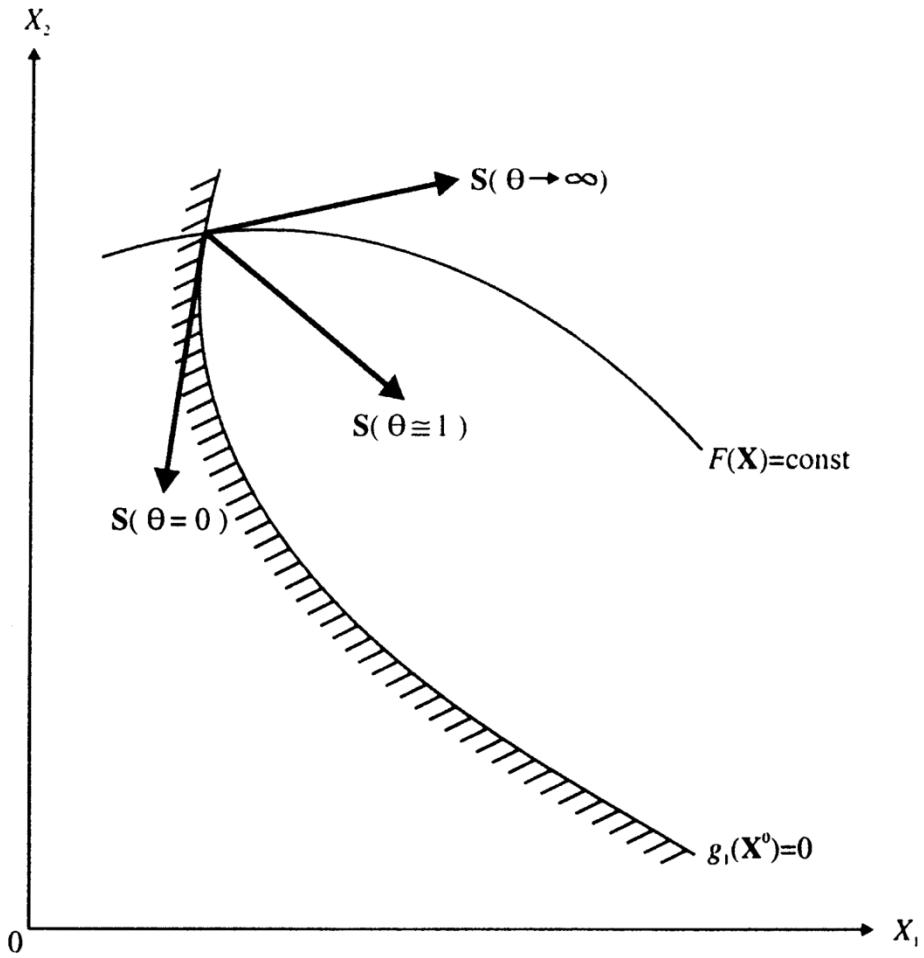
$\nabla g_j(\boldsymbol{x_0})^T \boldsymbol{s} + \beta\theta_j \leq 0, \quad j \in J$

$\boldsymbol{s}$ bounded

where J is the set of indices corresponding to the active inequality constraints.

Method of Feasible Directions

What happens to the search direction as the value of θ is changed?



Vanderplaats, Fig. 6.5

Method of Feasible Directions

Questions related to our direction-finding optimization problem:

- How/when do we define a constraint as active?
- What are reasonable values of the push-off factors, θ_j?
- What are the bounds we should impose on $\boldsymbol{s}$?

MoFD: Identifying Active Constraints

We consider a constraint "becoming active" when,

$$g_j(\boldsymbol{x}) \geq \varepsilon$$

where ε is a small (-) number, i.e. activity occurs when the constraint boundary is approached from within the feasible region.

ε is chosen to be larger in magnitude at the beginning of the optimization, e.g. $\varepsilon = -0.1$ and smaller in magnitude near the end, e.g. $\varepsilon = -0.001$. This approach avoids constraints turning on and off repetitively during the early phases of the search.

MoFD: Defining the Push-Off Factor

A typical approach to specifying θ is,

$$\theta_j = \left[1 - \frac{g_j(\boldsymbol{x})}{\varepsilon}\right]^2 \theta_0$$

where $\theta_0 = 1$.

As the constraint starts becoming active by our definition of ε, we have $g_j(\boldsymbol{x}) = \varepsilon$ and $\theta_j = 0$. When the constraint is *truly* active, $g_j(\boldsymbol{x}) = 0$ and $\theta_j = \theta_0$.

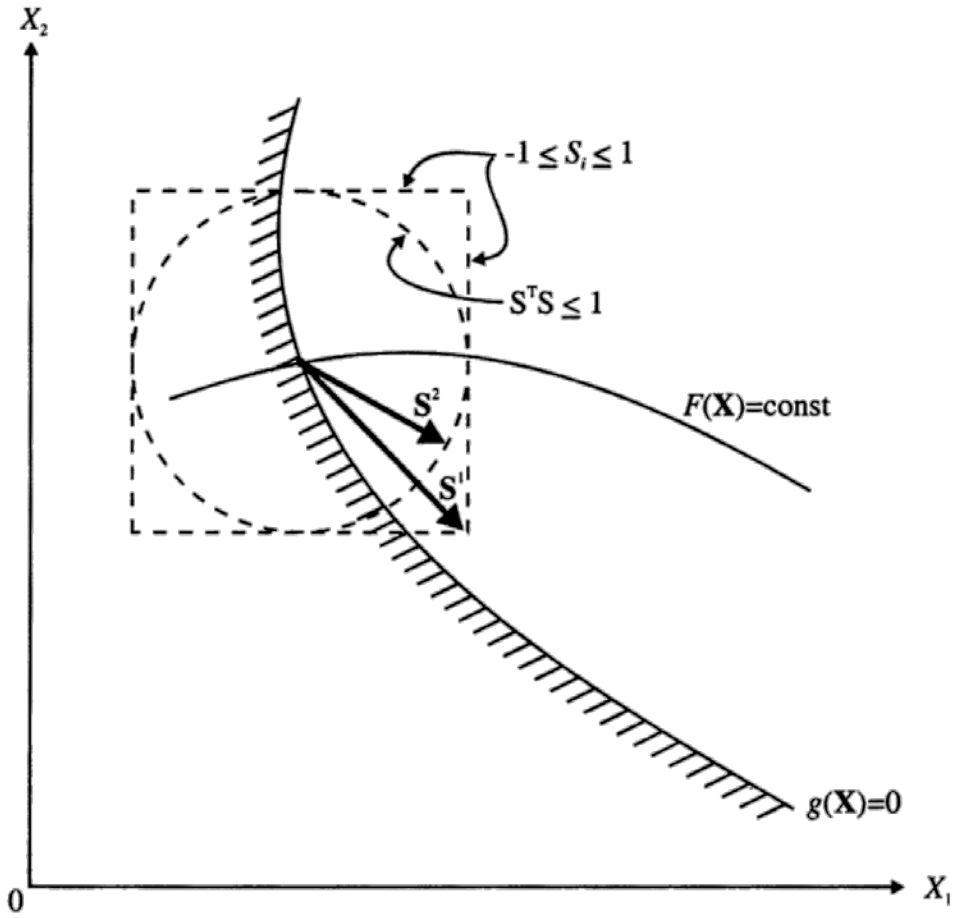
What is the search direction when $\theta_j = 0$?

Method of Feasible Direction

There are two common forms for bounds on the magnitude of the search direction, $\boldsymbol{s}$:

- Side constraints: $-1 \leq s_i \leq 1$
- 2-norm constraint: $\boldsymbol{s}^T \boldsymbol{s} \leq 1$

Method of Feasible Directions



Vanderplaats, Fig. 6.6

What is the disadvantage of imposing side constraints on s_i?

MoFD: Direction Finding

The direction-finding problem with side constraints on $\boldsymbol{s}$ is:

Maximize: β

By changing: $\boldsymbol{s}$, β

Subject to: $\nabla f(\boldsymbol{x_0})^T \boldsymbol{s} + \beta \leq 0$

$\nabla g_j(\boldsymbol{x_0})^T \boldsymbol{s} + \beta\theta_j \leq 0, \quad j \in J$

$-1 \leq s_i \leq 1$

This is a LP in terms of the variables $\boldsymbol{s}$ and β. **Important:** The solution of this problem is just the search direction. We still have to do line searches to solve the real optimization problem!!

MoFD: Direction Finding

The direction-finding problem with 2-norm constraints on $\boldsymbol{s}$ is:

Maximize: β

By changing: $\boldsymbol{s}, \beta$

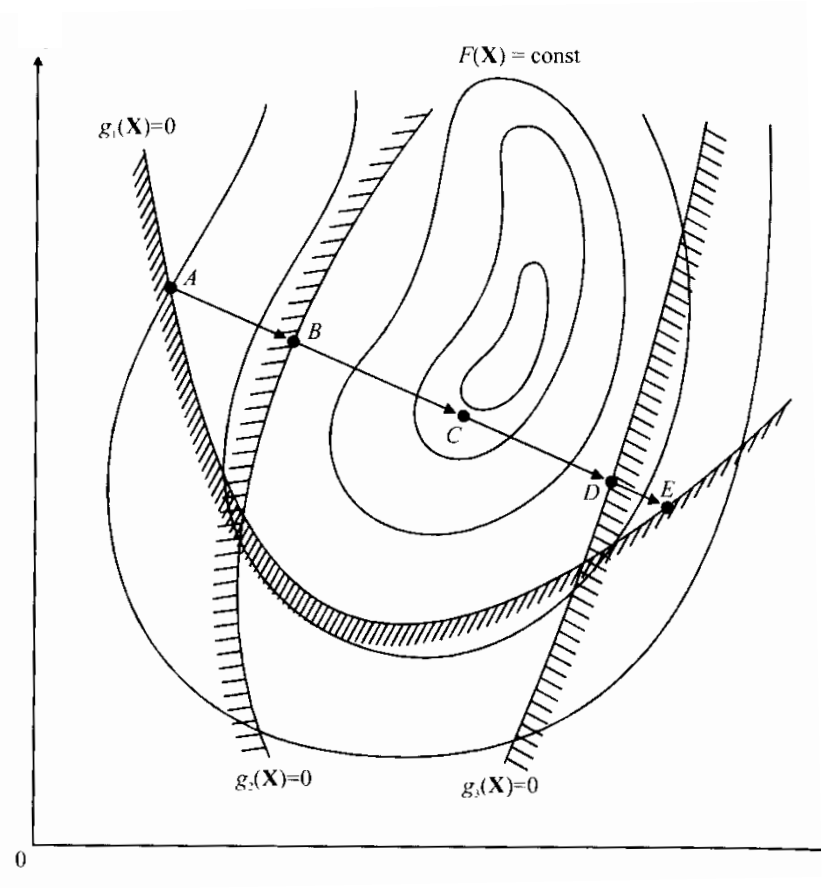
Subject to:
$$\nabla f(\boldsymbol{x_0})^T \boldsymbol{s} + \beta \leq 0$$
$$\nabla g_j(\boldsymbol{x_0})^T \boldsymbol{s} + \beta\theta_j \leq 0, \quad j \in J$$
$$\boldsymbol{s}^T \boldsymbol{s} \leq 1$$

This problem would be an LP except for the quadratic constraint. Zoutendijk showed that it can be solved as an LP through a transformation. See Vanderplaats for details.

MoFD: Line Searches

Let's say that we have found a search direction.

Can we use the same techniques as before to conduct the line search to find α^*? Why or why not?



Vanderplaats, Fig. 6.9

MoFD: Line Searches

To do a line search along the search direction found in the direction-finding problem we can proceed as follows:

1. Use the unconstrained line search techniques to find α^* along the search direction. Set $\alpha = \alpha^*$.
2. Evaluate $g_j(\boldsymbol{x_0} + \alpha \boldsymbol{s})$ for all j. Also evaluate $f(\boldsymbol{x_0} + \alpha \boldsymbol{s})$.
3. If $g_j(\boldsymbol{x_0} + \alpha \boldsymbol{s}) > 0$ for any j or if $f(\boldsymbol{x_0} + \alpha \boldsymbol{s}) > f(\boldsymbol{x_0})$, reduce α in some systematic way, e.g. $\alpha = \alpha/2$ and repeat from step 2.
4. If $g_j(\boldsymbol{x}) \geq \varepsilon$ for any j, where ε is the small (-) number defining constraint activity, stop the line search and move to this point.
5. If neither condition in 3 or 4 occurs, stop the line search and move to the unconstrained minimum defined by α^* in step 1.

There are many possible variations for this procedure!

MoFD: Convergence Criteria

In order to determine if the algorithm has reached a minimum, we have to check two possibilities:

- A minimum occurs at a point at which one or more of the constraints are active, i.e. the minimum is on the constraint boundary

- A minimum occurs without any constraints active, i.e. the minimum is in the interior of the feasible region

MoFD: Convergence Criteria

To determine if we have reached a minimum on the constraint boundary, recall our direction finding condition:

$$\nabla f(\boldsymbol{x_0})^T \boldsymbol{s} + \beta \leq 0 \Longrightarrow \boldsymbol{s}^T \nabla f(\boldsymbol{x_0}) + \beta \leq 0$$

Next, recall the Lagrangian requirement from the KKT conditions:

$$\nabla f(\boldsymbol{x_0}) + \sum_{j=1}^{p} \lambda_j \nabla g_j(\boldsymbol{x_0}) = \boldsymbol{0}$$

Multiply this KKT condition by $\boldsymbol{s}^T$:

$$\boldsymbol{s}^T \nabla f(\boldsymbol{x_0}) + \sum_{j=1}^{p} \lambda_j \boldsymbol{s}^T \nabla g_j(\boldsymbol{x_0}) = 0$$

MoFD: Convergence Criteria

Substituting the expression for $\boldsymbol{s}^T \nabla f(\boldsymbol{x_0})$ from the KKT condition,

$$\boldsymbol{s}^T \nabla f(\boldsymbol{x_0}) + \sum_{j=1}^{p} \lambda_j \boldsymbol{s}^T \nabla g_j(\boldsymbol{x_0}) = 0$$

into the direction constraint,

$$\boldsymbol{s}^T \nabla f(\boldsymbol{x_0}) + \beta \leq 0$$

gives,

$$\beta \leq \sum_{j=1}^{p} \lambda_j \boldsymbol{s}^T \nabla g_j(\boldsymbol{x_0})$$

MoFD: Convergence Criteria

$$\beta \leq \sum_{j=1}^{p} \lambda_j \boldsymbol{s}^T \nabla g_j(\boldsymbol{x_0})$$

But, $\nabla g_j(\boldsymbol{x_0})^T \boldsymbol{s} = \boldsymbol{s}^T \nabla g_j(\boldsymbol{x_0}) \leq 0$ for feasible directions, $\lambda_j > 0$ for active constraints, and we require $\beta \geq 0$.

The only possibility that satisfies these restrictions is:

$$\beta = 0, \qquad \boldsymbol{s} = \boldsymbol{0}$$

The converge criterion for a solution at the constraint boundary is therefore, $\beta = 0$ or better, $\beta \leq \epsilon_1$ where ϵ_1 is a small (+) number. When the direction finding algorithm returns $\beta \leq \epsilon_1$, stop the search.

MoFD: Convergence Criteria

To determine whether we have converged within the interior of the feasible region, stop whenever either of the following conditions holds,

$$\left| \frac{f(\boldsymbol{x_k}) - f(\boldsymbol{x_{k-1}})}{f(\boldsymbol{x_{k-1}})} \right| \leq \epsilon_2$$

$$\|\boldsymbol{x_k} - \boldsymbol{x_{k-1}}\| \leq \epsilon_3$$

where ϵ_2 and ϵ_3 are small (+) numbers.

MoFD: Overall Algorithm

1. Select an initial feasible point $\boldsymbol{x_0}$ and values for ϵ_1, ϵ_2 and ϵ_3.
2. If $g_j(\boldsymbol{x_0}) < 0$ for all j, set the search direction as steepest descent: $\boldsymbol{s} = -\nabla f(\boldsymbol{x_0})$ (best to normalize $\boldsymbol{s}$ too)
3. If at least one $g_j(\boldsymbol{x_0}) = 0$ (or $g_j(\boldsymbol{x}) \geq \varepsilon$), solve the direction-finding problem via LP or similar methods to determine $\boldsymbol{s}$.
4. If $\beta \leq \epsilon_1$ from the direction finding problem, stop the search. If not, proceed.
5. Conduct a line search in the $\boldsymbol{s}$ direction following the procedure we discussed and move to the point identified.
6. If the line search terminated within the interior of the feasible region, check for convergence with the unconstrained criteria.
7. Update iteration number, θ, and ε and repeat from step 2.

Algorithm adapted from (Rao, 1996)

MoFD: Modification for Initially Infeasible Designs

If $\boldsymbol{x_0}$ is not feasible, we modify the direction-finding algorithm to produce a direction that will move us toward the feasible region. The idea is to omit the usability constraint and modify the objective function:

Maximize: $-\nabla f(\boldsymbol{x_0})^T \boldsymbol{s} + \Phi\beta$

By changing: $\boldsymbol{s}$

Subject to: $\nabla g_j(\boldsymbol{x_0})^T \boldsymbol{s} + \beta\theta_j \leq 0, \quad j \in J$

$\boldsymbol{s}^T \boldsymbol{s} \leq 1$

Φ is set to a large value, e.g. $\Phi \geq 10^5$.

Generalized Reduced Gradient Method

The concept of the generalized reduced gradient method is to move in search directions along active constraints, presuming that the constraints stay active during the move.

If a constraint becomes inactive during the move because of nonlinearity, we step back to the constraint with Newton's method for root finding.

The formulation begins by converting a general constrained optimization problem to a problem with only equality constraints by introducing slack variables.

Generalized Reduced Gradient Method

Slack variables x_{n+j} are added to the design variable vector to convert the inequality constraints to equality constraints:

Minimize: $f(\boldsymbol{x})$

Subject to:
$$g_j(\boldsymbol{x}) + x_{n+j} = 0, \quad j = 1, \ldots, m$$
$$h_k(\boldsymbol{x}) = 0, \quad k = 1, \ldots, l$$
$$x_{L,i} \leq x_i \leq x_{U,i}, \quad i = 1, \ldots, n$$
$$x_{n+j} \geq 0$$

Generalized Reduced Gradient Method

Equality constraints effectively reduce the dimensionality of the search.

If we have l equality constraints and n design variables, we have only $n - l$ independent degrees of freedom in the search.

The remaining $m + l$ design variables are dependent.

This concept should seem familiar from our application of direct substitution of equality constraints.

Generalized Reduced Gradient Method

To account for this dependency, we partition the design variables into two groups, one group of independent variables and one group that is dependent based on the equality constraints:

$\boldsymbol{z}$, $(n-l) \times 1$ vector of independent variables

$\boldsymbol{y}$, $(m+l) \times 1$ vector of dependent variables

$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{y} \end{bmatrix}$, $(n+m) \times 1$ vector

Generalized Reduced Gradient Method

With the inclusion of slack variables, we can subsume the original inequality constraints, $g_j(\boldsymbol{x})$, into the set of equality constraints.

We can also include the nonnegativity constraints on the slack variables in the side constraints, with 0 as the lower bounds and very large upper bounds.

Minimize: $f(\boldsymbol{z}, \boldsymbol{y})$

Subject to: $h_j(\boldsymbol{x}) = 0, \quad j = 1, \ldots, m + l$

$x_{L,i} \leq x_i \leq x_{U,i}, \quad i = 1, \ldots, n + m$

Generalized Reduced Gradient Method

Next, we write the total differentials of the objective function and equality constraints:

$$df = \nabla_z f(\boldsymbol{x})^T \boldsymbol{dz} + \nabla_y f(\boldsymbol{x})^T \boldsymbol{dy}$$

$$dh_j = \nabla_z h_j(\boldsymbol{x})^T \boldsymbol{dz} + \nabla_y h_j(\boldsymbol{x})^T \boldsymbol{dy}$$

Generalized Reduced Gradient Method

The differentials for the constraints can be written as,

$$\boldsymbol{dh} = A\boldsymbol{dz} + B\boldsymbol{dy}$$

where,

$$A = \begin{bmatrix} \nabla_z h_1(\boldsymbol{x})^T \\ \vdots \\ \nabla_z h_{m+l}(\boldsymbol{x})^T \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} \nabla_y h_1(\boldsymbol{x})^T \\ \vdots \\ \nabla_y h_{m+l}(\boldsymbol{x})^T \end{bmatrix}$$

Generalized Reduced Gradient Method

Presuming that the equality constraints were initially satisfied, they must remain satisfied for feasibility.

This requires that,

$$\boldsymbol{dh} = A\boldsymbol{dz} + B\boldsymbol{dy} = \boldsymbol{0}$$

or,

$$\boldsymbol{dy} = -B^{-1}A\boldsymbol{dz}$$

Generalized Reduced Gradient Method

Substituting,

$$\boldsymbol{dy} = -B^{-1}A\boldsymbol{dz}$$

into the differential of f,

$$df = \nabla_z f(\boldsymbol{x})^T \boldsymbol{dz} + \nabla_y f(\boldsymbol{x})^T \boldsymbol{dy}$$

gives, $$df = \boldsymbol{G_R^T dz}$$

where $$\boldsymbol{G_R^T} = \left\{\nabla_z f(\boldsymbol{x})^T - \nabla_y f(\boldsymbol{x})^T [B^{-1}A]\right\}$$

is transpose of the ***generalized reduced gradient*** of f.

Generalized Reduced Gradient Method

The generalized reduced gradient can be used to define a search direction. For example, we could step in the "steepest descent" direction with respect to $\boldsymbol{G_R}$:

$$\boldsymbol{s = -G_R}$$

Note that this is a search in terms of the independent variables $\boldsymbol{z}$.

In general, we could use other search direction methods for unconstrained optimization to set $\boldsymbol{s}$ based on the reduced set of variables $\boldsymbol{z}$.

Generalized Reduced Gradient Method

With $\boldsymbol{s}$ defined, we now conduct a line search in the $\boldsymbol{z}$ variables.

Because we linearized the constraints to find $\boldsymbol{s}$, it is likely that the constraints are not crisply enforced at any particular $\alpha \neq 0$.

To fix this issue, we need to find a new $\boldsymbol{dh}$ such that,

$$\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{dh}(\boldsymbol{x}) = 0$$

where $\boldsymbol{x}$ now indicates the new point defined by α from the line search. $\boldsymbol{h}(\boldsymbol{x})$ is the actual value of the equality constraints at this new point found from the line search in the $\boldsymbol{z}$ variables.

Generalized Reduced Gradient Method

Since,

$$\boldsymbol{dh}(\boldsymbol{x}) = A\boldsymbol{dz}(\boldsymbol{x}) + B\boldsymbol{dy}(\boldsymbol{x}),$$

we have,

$$-\boldsymbol{h}(\boldsymbol{x}) = A\boldsymbol{dz}(\boldsymbol{x}) + B\boldsymbol{dy}(\boldsymbol{x})$$

or,

$$\boldsymbol{dy}(\boldsymbol{x}) = -B^{-1}[\boldsymbol{h}(\boldsymbol{x}) + A\boldsymbol{dz}(\boldsymbol{x})]$$

where we hold $\boldsymbol{dz}(\boldsymbol{x})$ fixed as the value from the line search.

Generalized Reduced Gradient Method

Finally, we set,

$$\boldsymbol{y_{new}(x) = y_{old}(x) + dy(x)}$$

and reevaluate $\boldsymbol{h(x)}$. We iterate this process of evaluating $\boldsymbol{h(x)}$ and updating $\boldsymbol{y_{new}(x)}$ until $\boldsymbol{h(x)} \approx 0$ to some tolerance, e.g. $\|\boldsymbol{h(x)}\| \leq \varepsilon$.

This is an application of Newton's method for root finding.

Generalized Reduced Gradient Method

After setting up the problem in slack variable form in terms of equality constraints, the general algorithm is as follows:

1. Calculate the gradients of the objective and constraints.
2. Calculate the reduced gradient, $\boldsymbol{G_R}$.
3. Determine a search direction, $\boldsymbol{s}$.
4. Perform a line search w.r.t the independent variables, $\boldsymbol{z}$. Calculate $\boldsymbol{h}$ and iteratively update the dependent variables, $\boldsymbol{y}$, using Newton's method for each trial α.
5. Repeat steps 1 through 4 until convergence

Generalized Reduced Gradient Method

There are other subtle but important issues that must be addressed to implement the generalized reduced gradient method:

- How to select which variables are independent and which are dependent to make sure that the matrix B is not singular
- How to handle side constraints effectively
- How to deal with initially infeasible designs

See Vanderplaats section 6.6 for details.

Generalized Reduced Gradient Method

Advantages:

- Very efficient compared to SUMT or SLP if function and gradient evaluations are expensive
- Somewhat challenging to implement robustly

Disadvantages:

- Newton's method for updating the dependent variables may fail for highly nonlinear problems
- Memory requirements can be considerable because addition of slack variables increases dimensionality
- Moves from one constraint vertex to another, generally requiring n-1 line search iterations
- Spends much of its time recovering from moves into infeasible region

Sequential Quadratic Programming

As its name implies, Sequential Quadratic Programming (SQP) develops quadratic approximations to the objective function.

The constraints are linearized.

These approximations are solved sequentially, in a manner similar to SLP, to converge to the solution.

Because SQP involves a quadratic approximation, certain implementations of SQP share similarities to Newton's method and quasi-Newton methods such as BFGS.

Sequential Quadratic Programming

The basic form of the quadratic approximation is built as a Taylor series around a current point $\boldsymbol{x}$ as follows:

Minimize: $$q(\boldsymbol{s}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T B \boldsymbol{s}$$

Subject to: $$g_j(\boldsymbol{x}) + \nabla g_j(\boldsymbol{x})^T \boldsymbol{s} \leq 0, \quad j = 1, \ldots, m$$
$$h_k(\boldsymbol{x}) + \nabla h_k(\boldsymbol{x})^T \boldsymbol{s} = 0, \quad k = 1, \ldots, l$$

- The matrix B is either the Hessian or is "Hessian-like," e.g. found via BFGS.
- This is a "quadratic program."

Quadratic Programming

Methods for solving inequality-constrained QPs include:

- Active set methods
- Interior point methods

See Nocedal and Wright Chapter 16 for details.

As their name implies, active set methods attempt to identify the "active set" of inequality constraints.

This active set can be added to the set of equality constraints.

The equality-constrained problem can then be solved as a linear system with efficient algorithms.

Equality-Constrained Quadratic Programming

When all constraints are equalities, or we know the active set of inequality constraints, the QP formulation is straightforward:

Minimize: $\frac{1}{2}\boldsymbol{x}^T G\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{c}$

Subject to: $A\boldsymbol{x} = \boldsymbol{b}$

The Lagrangian of this problem is

$$\mathcal{L} = \left(\frac{1}{2}\boldsymbol{x}^T G\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{c}\right) + \boldsymbol{\lambda}^T(A\boldsymbol{x} - \boldsymbol{b})$$

Equality-Constrained Quadratic Programming

$$\mathcal{L} = \left(\frac{1}{2}\boldsymbol{x}^T G\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{c}\right) + \boldsymbol{\lambda}^T(A\boldsymbol{x} - \boldsymbol{b})$$

The KKT necessary conditions therefore require:

$$\nabla_x \mathcal{L} = (G\boldsymbol{x}^* + \boldsymbol{c}) + (A^T\boldsymbol{\lambda}^*) = \mathbf{0}$$

$$A\boldsymbol{x}^* - \boldsymbol{b} = \mathbf{0}$$

These equations can be written as a linear system:

$$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix}\begin{bmatrix} \boldsymbol{x}^* \\ \boldsymbol{\lambda}^* \end{bmatrix} = \begin{bmatrix} -\boldsymbol{c} \\ \boldsymbol{b} \end{bmatrix}$$

Equality-Constrained Quadratic Programming

The linear system of KKT conditions for equality-constrained QPs can be solved by several different approaches:

Direct solution approaches

- Symmetric indefinite factorization
- Schur-complement method
- Null-space method

See Nocedal and Wright Chapter 16 for details.

Iterative solution approaches

- Conjugate gradient method
- Projected conjugate gradient method

SQP: Practical Considerations

There are many algorithms for SQP. Most leverage equality-constrained QP algorithms by determining the "active set" or a "working set" of constraints at each iteration.

Both line search and trust region implementations are possible.

Most methods "modify" the simple quadratic approximation we examined earlier to account for practical considerations.

Nocedal & Wright provides a comprehensive discussion in Chp. 18.

Vanderplaats discusses a line search method that we will discuss.

SQP: Direction-Finding Problem

In the Vanderplaats line search implementation, the SQP direction-finding problem is as follows:

Minimize: $q(\boldsymbol{s}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T B \boldsymbol{s}$

By changing: $\boldsymbol{s}$

Subject to: $\nabla g_j(\boldsymbol{x})^T \boldsymbol{s} + \delta_j g_j(\boldsymbol{x}) \leq 0, \quad j = 1, \ldots, m$

$\nabla h_k(\boldsymbol{x})^T \boldsymbol{s} + \bar{\delta} h_k(\boldsymbol{x}) = 0, \quad k = 1, \ldots, l$

Let's look at each of the individual components…

SQP: Direction-Finding Problem

The objective function is a quadratic approximation of the function in the neighborhood of the current $f(\boldsymbol{x})$:

$$q(\boldsymbol{s}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T B \boldsymbol{s}$$

The matrix B is "Hessian-like" in the sense of quasi-Newton methods.

Like Newton's method, an initial guess of $\alpha = 1$ is a good estimate for α^*.

SQP: Direction-Finding Problem

The constraints

$$\nabla g_j(\boldsymbol{x})^T \boldsymbol{s} + \delta_j g_j(\boldsymbol{x}) \leq 0, \quad j = 1, \ldots, m$$
$$\nabla h_k(\boldsymbol{x})^T \boldsymbol{s} + \bar{\delta} h_k(\boldsymbol{x}) = 0, \quad k = 1, \ldots, l$$

are modified with the factors δ_j and $\bar{\delta}$ which satisfy,

$$\delta_j = 1 \text{ if } g_j(\boldsymbol{x}) < 0$$
$$\delta_j = \bar{\delta} \text{ if } g_j(\boldsymbol{x}) \geq 0$$
$$0 \leq \bar{\delta} \leq 1 \quad \text{(typically ~0.95)}$$

The purpose of δ_j and $\bar{\delta}$ is to prevent the linearization from creating artificially over-constrained and infeasible search regions.

SQP: Line Search Problem

Once $\boldsymbol{s}$ is found from the direction-finding problem, Vanderplaats suggests solving a line search along $\boldsymbol{s}$ with an exterior penalty function of the form:

$$\phi(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} u_j \{\max[0, g_j(\boldsymbol{x})]\} + \sum_{k=1}^{l} u_{m+k} |h_k(\boldsymbol{x})|$$

where $\boldsymbol{x} = \boldsymbol{x_{k-1}} + \alpha \boldsymbol{s}$,

$u_j = |\lambda_j|, \quad j = 1, .., m + l$ (first iteration)

$u_j = \max\left[|\lambda_j|, \frac{1}{2}(u_j' + |\lambda_j|)\right]$ (subsequent iteration)

and u_j' is the u_j from the previous iteration. λ_j are Lagrange multipliers from the solution to the direction-finding algorithm.

SQP Overall Line Search Algorithm

The basic algorithm for Vanderplaat's line search approach is as follows:

1. Solve a direction-finding problem based on a quadratic approximation of the objective function and linearized constraints. For the first iteration, set $B = I$.
2. Conduct a 1-D search along the search direction to find the minimum of a pseudo-objective involving an exterior penalty function. A good initial guess is for α^* is $\alpha = 1$.
3. Update the matrix B with the BFGS formulation and repeat from step 1 until convergence