Why Numerical Method ?

- Analytical method $\rightarrow$ Numerical method
- # of design variables and constraints can be large.
  - Necessary conditions $\rightarrow$ a large number of equations
  - Functions for the design problem (cost and constraint) can be highly nonlinear.
- Cost and/or constraint functions can be implicit in terms of design variables.
- Search for the general purpose code through the internet to minimize developing your own code
  - Appendix B, https://neos-guide.org/

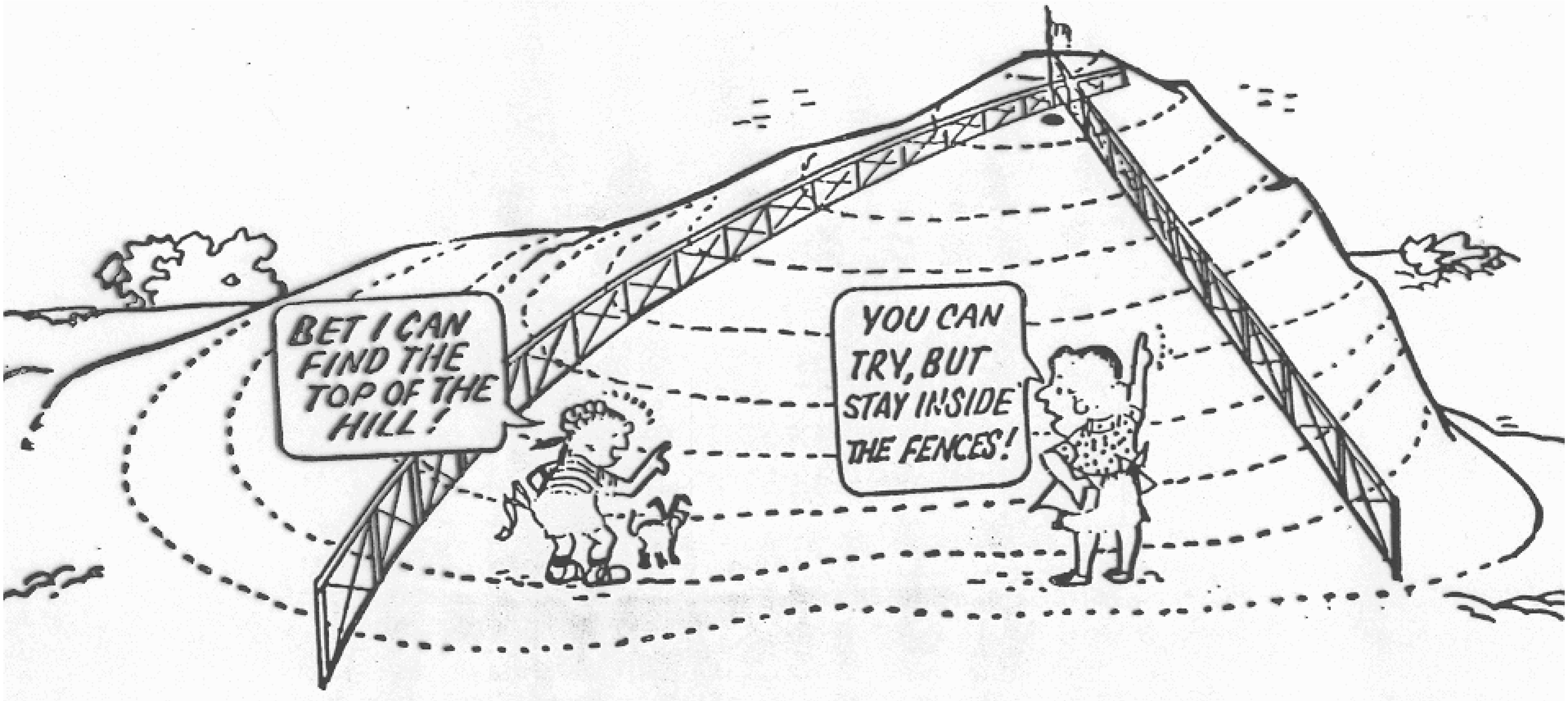
Advantages of Numerical Optimization

- Reduce the design time
  - When the same computer program can be applied to many design projects
- Provide a systematized logical design procedure
- Deal with a wide variety of design variables and constraints
- Yield some design improvement
- Not biased by intuition or experience in engineering
- Require a minimal amount of human-machine interaction

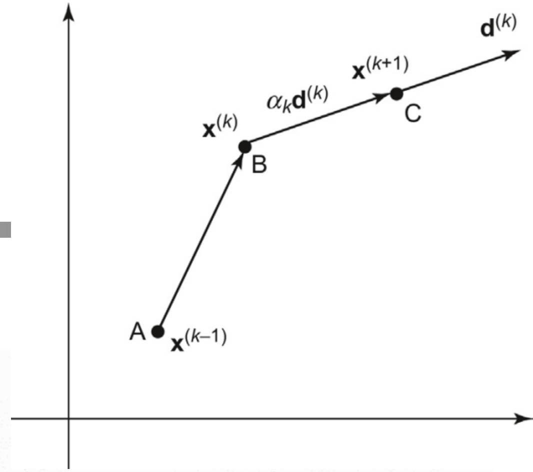
Limitations of Numerical Optimization

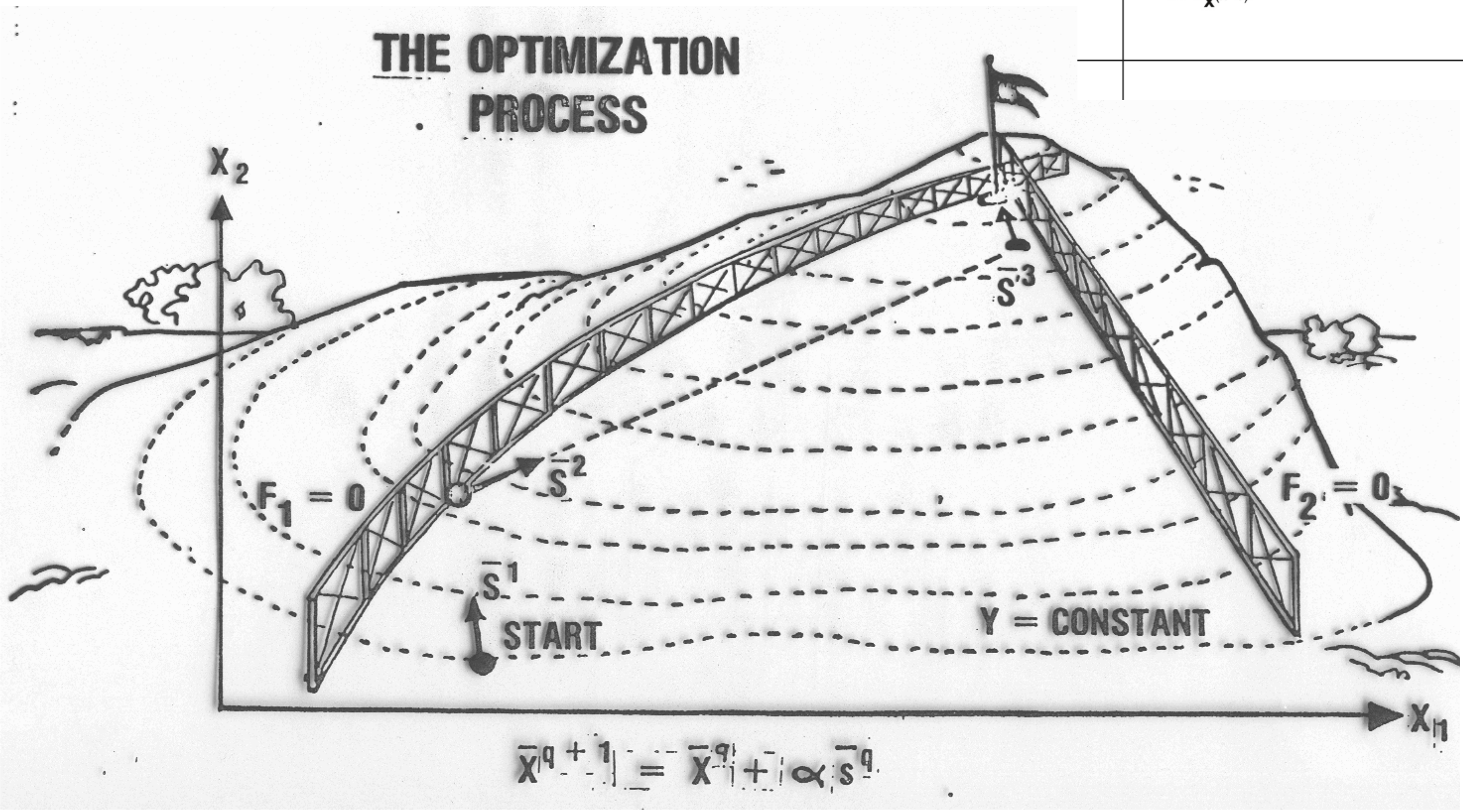
- Increased computational time as the number of design variables increases (ill-conditioned?)
- No stored experience or intuition
- Misleading results if the analysis program is not theoretically precise
- Difficulty in dealing with discontinuous functions and highly nonlinear problems
- Seldom be guaranteed that the optimization algorithm will obtain the global optimum design
- Significant reprogramming of analysis routines for adaptation to an optimization code

Physical Problem



Optimization Process





Linear Programming (LP) Problem

- Constrained optimization
- "Liner": the objective and the constraints
- "Programming": scheduling or setting an agenda
- Minimization of a function with equality constraints and nonnegativity of design variables

$$\text{Minimize} \quad f = \sum_{i=1}^{n} c_i x_i$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j = b_i; \qquad i = 1, \ldots, m$$

$$x_j \geq 0; \qquad j = 1, \ldots, n$$

$$\left(b_i \geq 0 : \text{resource limits, } c_i \text{ and } a_{ij} : \text{known constants} \right)$$

$$\begin{aligned} &\text{Minimize} && f = \mathbf{c}^T \mathbf{x} \\ &\text{subject to} && \mathbf{A}\mathbf{x} = \mathbf{b} \\ & && \mathbf{x} \geq 0 \end{aligned}$$

Standard LP Definition

- Linear constraints
  - Inequality: nonnegative slack variable s_i ($s_i \geq 0$)
    - Why not s_i^2 ? (nonlinear)
  - Treatment of "≤ type" / "≥ type" constraints

$$\begin{cases} 2x_1 - x_2 \leq 4 \rightarrow 2x_1 - x_2 + s_1 = 4 & (s_1 \geq 0) \\ -x_1 + 2x_2 \geq 2 \rightarrow -x_1 + 2x_2 - s_1 = 2 & (s_1 \geq 0) \end{cases}$$

- Unrestricted variables in sign
  - All design variables to be nonnegative

$$x_j = x_j^+ - x_j^- = \begin{cases} \text{nonnegative}: x_j^+ \geq x_j^- \\ \text{nonpositive}: x_j^+ \leq x_j^- \end{cases}$$

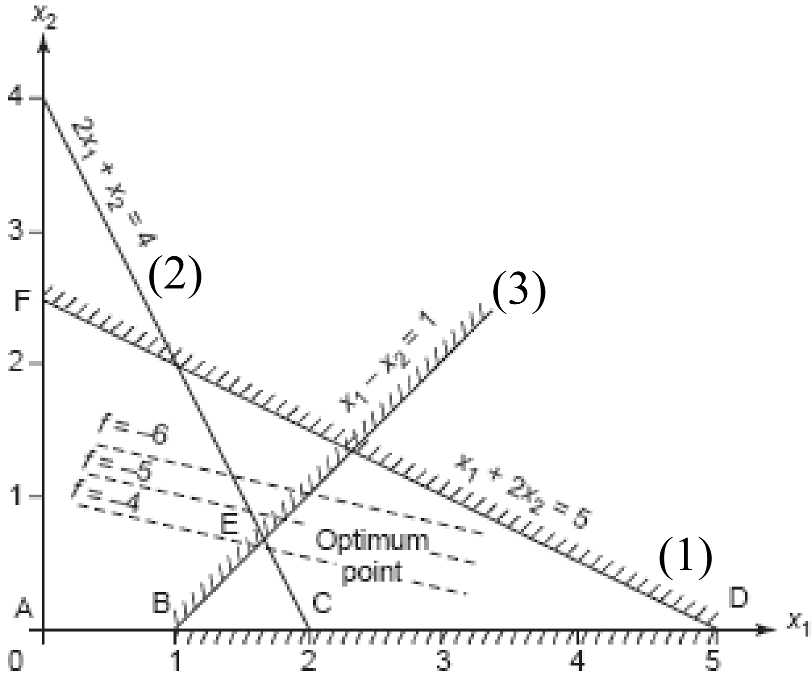
$$x_j^+ \geq 0 \text{ and } x_j^- \geq 0$$

Basic Concepts

- LP problem is convex. If an optimum solution exists, it is global.
  - Feasible region (constraint set) is convex
  - Cost function is linear, so it is convex
- Solution always lies on the boundary of the feasible region if it exists.
  - For an unconstrained optimum, contradiction: $\frac{\partial f}{\partial x_i} = 0 \rightarrow c_i = 0$
- Optimum solution must satisfy equality constraints $\rightarrow$ more than one solution (m < n)
  - Infinite solutions $\rightarrow$ feasible solution that minimizes the cost function

Example 8.19 ← 8.13

$$\left.\begin{array}{ll} \textit{Maximize} & z = x_1 + 4x_2 \\ \textit{subject to} & (1)\ x_1 + 2x_2 \le 5 \\ & (2)\ 2x_1 + x_2 = 4 \\ & (3)\ x_1 - x_2 \ge 1 \\ & x_1, x_2 \ge 0 \end{array}\right\} \rightarrow \left\{\begin{array}{llllll} \textit{Minimize} & f = -x_1 - 4x_2 & & & & \\ \textit{subject to} & x_1 + 2x_2 + x_3 & & & & = 5 \\ & 2x_1 + x_2 & & + x_5 & & = 4 \\ & x_1 - x_2 & - x_4 & & + x_6 & = 1 \\ & x_i \ge 0; \quad i = 1, \ldots, 6 & & & & \end{array}\right.$$

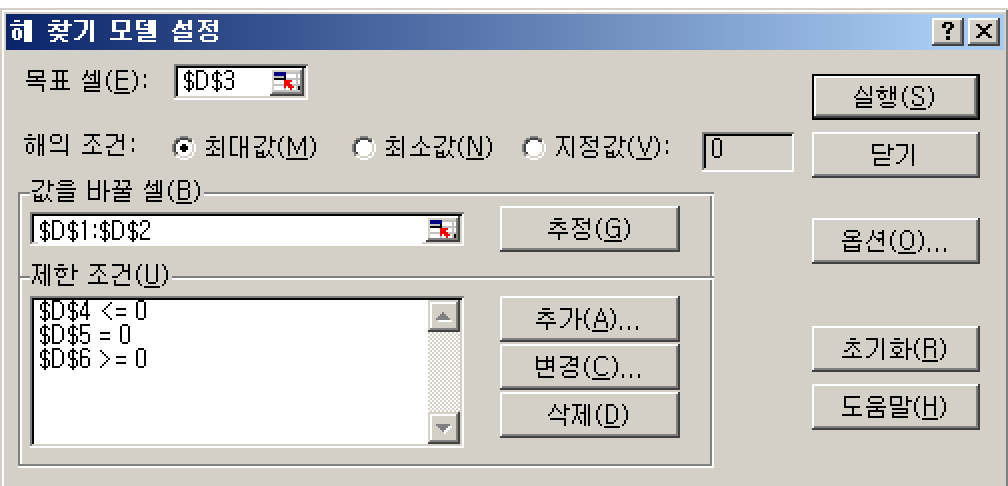


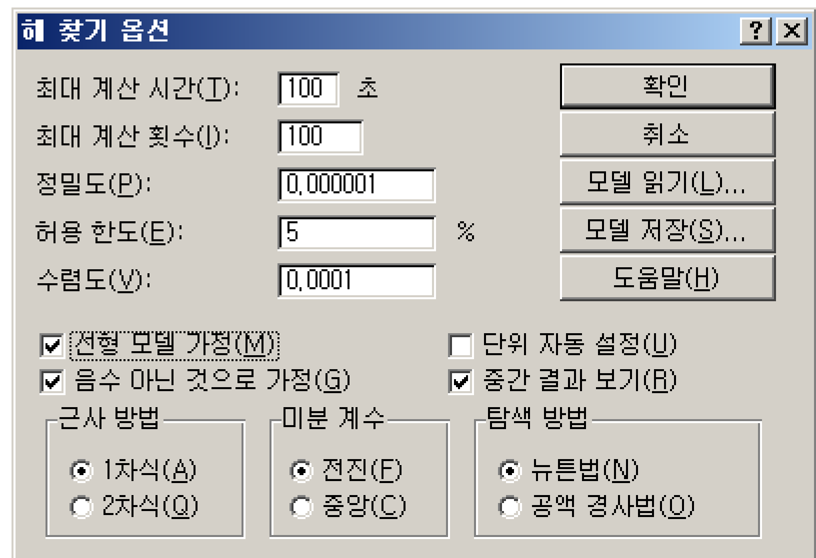
$$\left\{\begin{array}{l} \dfrac{\partial f}{\partial e_2} = -y_2 = \quad ;\ (2)\ 4 \rightarrow 5,\ \ f = \\ \dfrac{\partial f}{\partial e_3} = -y_3 = \quad ;\ (3)\ 1 \rightarrow 2,\ \ f = \end{array}\right.$$

LP in Excel Solver: Example 8.19

	A	B	C	D
1	x1	0		1,66667
2	x2	0		0,66667
3	z	0	max	4,33333
4	g1	-5	<= 0	-2
5	g2	-4	= 0	0
6	g3	-1	>= 0	0

1	2
1	1.666666667
0	0.666666667
1	4.333333333





Reports : Example 8.19+21+23

민감도보고서

Range of cost coeff.

값을 바꿀 셀

셀	이름	계산 값	한계 비용	목표 셀 계수	허용 가능 증가치	허용 가능 감소치
\$D\$1	x1	1.666666667	0	1	7	1E+30
\$D\$2	x2	0.666666667	0	4	1E+30	3.5

$$-7.0 \le \Delta c_1 \le \infty \xrightarrow{c_1=-1} -8.0 \le c_1 \le \infty$$

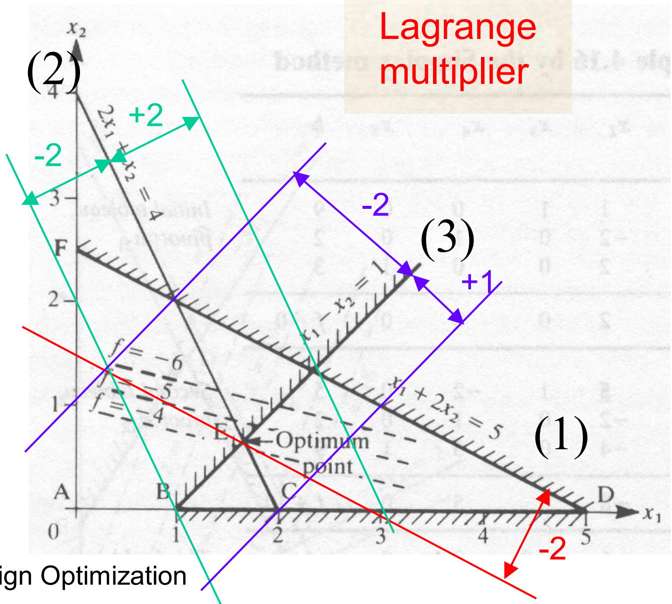
$$-\infty \le \Delta c_2 \le 3.5 \xrightarrow{c_2=-4} -\infty \le c_2 \le -0.5$$

제한 조건

셀	이름	계산 값	잠재 가격	제한 조건 우변	허용 가능 증가치	허용 가능 감소치
\$D\$6	>= 0	0	-2.333333333	0	1	2
\$D\$4	<= 0	-2	0	0	1E+30	2
\$D\$5	0	0	1.666666667	0	2	2

Lagrange multiplier

Range of resource limit



$$(1)\ x_1 + 2x_2 \le 5 \rightarrow -2 \le \Delta_1 \le \infty \rightarrow 3 \le b_1 \le \infty$$

$$(2)\ 2x_1 + x_2 = 4 \rightarrow -2 \le \Delta_2 \le 2 \rightarrow 2 \le b_2 \le 6$$

$$(3)\ x_1 - x_2 \ge 1 \rightarrow -2 \le \Delta_3 \le 1 \rightarrow -1 \le b_3 \le 2$$

Nonlinear Optimization

- Unlike for linear problems, a global optimum for a nonlinear problem cannot be guaranteed, except for special cases, e.g., if you know the space is unimodal, or convex, or monotonicity exists
- Two standard heuristics that most people use:
  - Find local extrema starting from widely varying starting points of variables and then pick the most extreme of these extrema
  - Perturb a local extremum by taking a finite amplitude step away from it, and then see whether your routine returns you to a better point or "always" to the same one
  - Question: How would you "automate" a search for a global extremum?

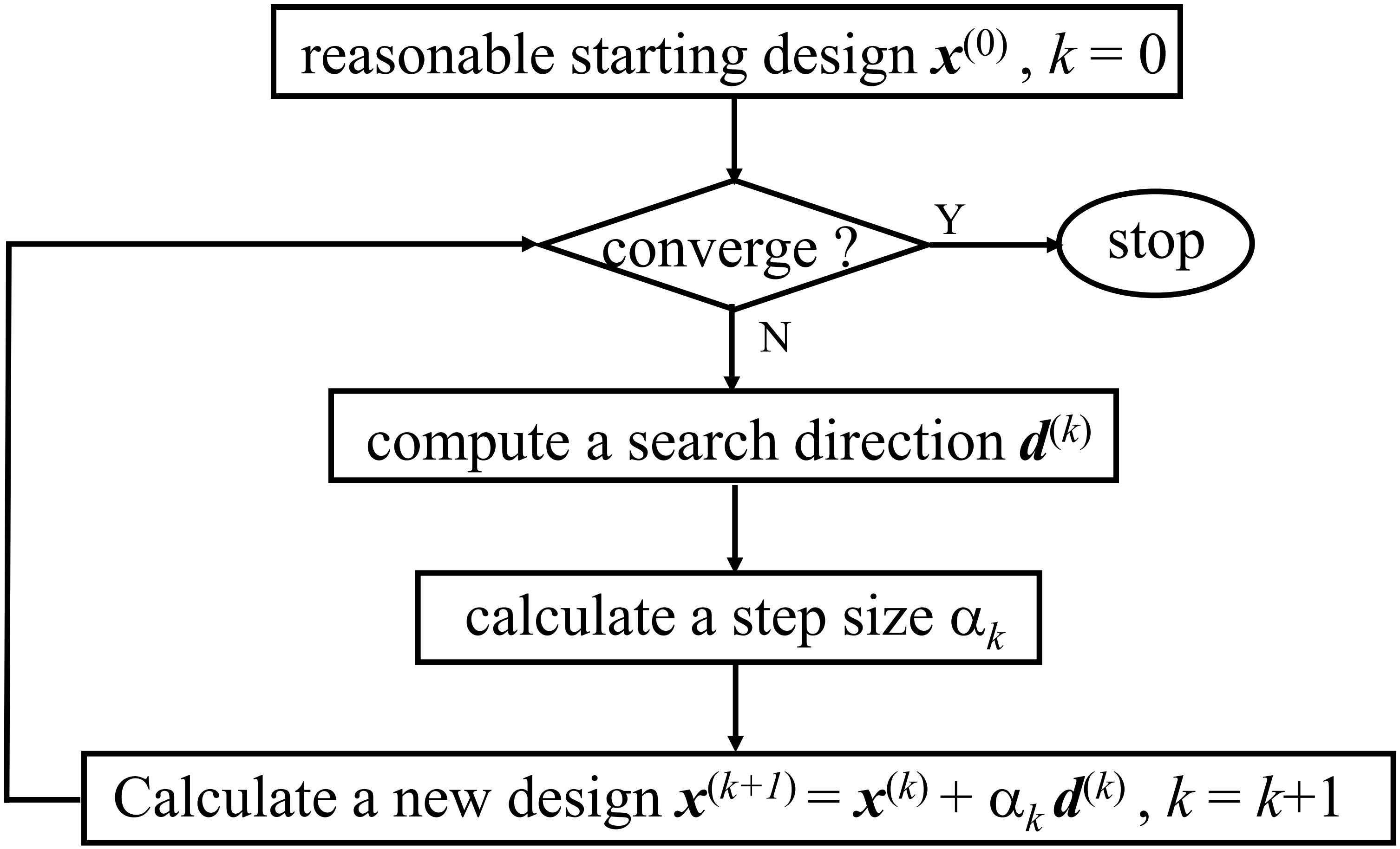
Basic Steps in Nonlinear Optimization

- In its simplest form, a numerical search procedure consists of four steps when applied to unconstrained minimization problem:
  - (1) Selection of an initial design in the *n*-dimensional space, where *n* is the number of design variables
  - (2) A procedure for the evaluation of the objective function at a given point in the design space
  - (3) Comparison of the current design with all of the preceding designs
  - (4) A rational way to select a new design and repeat the process
  - Constrained optimization requires step for evaluation of constraints as well. Same applies for evaluating multiple objective functions

Nonlinear Optimization Process

- Most design tasks seek to find a perturbation to an existing design which will lead to an improvement. Thus we seek a new design which is the old design plus a change
  - $X^{new} = X^{old} + \delta X$
- Optimization algorithms apply a two step process :
  - $X^{(k+1)} = X^{(k)} + \alpha_k d^{(k)}$
  - You have to provide an initial design $X^{(0)}$
  - The optimization will then determine a search direction $d^{(k)}$ that will improve the design
  - How far we can move in direction $d^{(k)} \rightarrow$ one-dimensional search to determine the scalar α_k to improve the design

General Algorithm



Classification of Unconstrained Optimization

- One-dimensional unconstrained optimization: line search
  - Golden-section search
  - Quadratic interpolation
- Multidimensional unconstrained optimization
  - Nongradient or Direct methods
  - Gradient or Descent methods

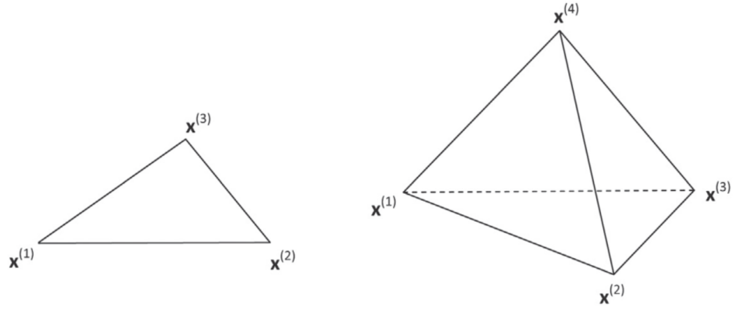
    - You often must choose between algorithms which need only evaluations of the objective function or methods that also require the derivatives of that function
    - Algorithms using derivatives are generally more powerful, but do not always compensate for the additional calculations of derivatives
    - Note that you may not be able to compute the derivatives

Multidimensional Unconstrained Optimization

Direct Search Methods	Indirect(Descent) Methods
▪ Random search method	▪ Steepest descent (Cauchy) method
▪ Univariate method	▪ Conjugate gradient method
▪ Pattern search method	– Fletcher-Reeves
– Powell's method	– Polak-Rebiere
▪ Simplex method	▪ Newton's method
▪ Simulated Annealing (SA)	▪ Marquardt's method
▪ Genetic Algorithm (GA)	▪ Quasi-Newton methods
	– DFP (Davidon-Fletcher-Powell)
	– BFGS(Broydon-Fletcher-Goldfarb-Shanno)

Nelder–Mead Simplex Method

- Does not use gradients of the cost function
- Idea of a *simplex*
  - Geometric figure formed by a set of (n+1) points in the n-dimensional space
  - When the points are equidistant, the simplex is said to be *regular*
- Nelder–Mead method (Nelder and ead, 1965)
  - Compute cost function value at the (n+1) vertices of the simplex
  - Move this simplex toward the minimum point
  - reflection, expansion, contraction, and shrinkage
  - MATLAB: fminsearch



(a) (b)

Descent Directions (1)

- Steepest descent direction: $\mathbf{d} = -\nabla f = -\dfrac{\partial f}{\partial x}$
- Conjugate Gradient direction:

$$\mathbf{d}^{(k)} = -\nabla f\left(\mathbf{x}^{(k)}\right) + \beta_k \mathbf{d}^{(k-1)} \quad \text{where} \quad \beta_k = \frac{\left\|\nabla f\left(\mathbf{x}^{(k)}\right)\right\|^2}{\left\|\nabla f\left(\mathbf{x}^{(k-1)}\right)\right\|^2}$$

- Newton's method:

$$f\left(\mathbf{x} + \Delta\mathbf{x}\right) = f\left(\mathbf{x}\right) + \nabla f^T\left(\mathbf{x}\right)\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^T\mathbf{H}\Delta\mathbf{x}$$

$$\frac{\partial f}{\partial\left(\Delta x\right)} = 0 \Rightarrow \nabla f\left(\mathbf{x}\right) + \mathbf{H}\Delta\mathbf{x} = 0$$

$$\mathbf{d}^{(k)} \equiv \Delta\mathbf{x} = -\mathbf{H}^{-1}\nabla f\left(\mathbf{x}\right) \rightarrow \mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \Delta\mathbf{x} \text{ (step length} = 1\text{)}$$

- Marquardt's method: $\mathbf{d}^{(k)} = -\left(\mathbf{H} + \lambda\mathbf{I}\right)^{-1}\nabla f\left(\mathbf{x}\right)$

Descent Directions (2)

- Quasi-Newton Method (Variable Metric Method)
  - Use of previous information, speed up the convergence !

$$\mathbf{d}^{(k)} = -\mathbf{A}^{(k)}\nabla f\left(\mathbf{x}^{(k)}\right) \Rightarrow \mathbf{A}^{(k+1)} = \mathbf{A}^{(k)} + \mathbf{A}_c^{(k)} \xrightarrow{\text{as } k\to\infty} \mathbf{H}^{-1}$$

  - DFP Method: Davidon (1959) $\rightarrow$ Fletcher and Powell (1963)
    - Approximate inverse of Hessian matrix
  - BFGS Method: Broyden-Fletcher-Goldfarb-Shanno (1981)
    - Direct update the Hessian matrix

$$x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$$

$$d^{(k)} = -A^{(k)}\nabla f\left(x^{(k)}\right)$$

$$A^{(k+1)} = A^{(k)} + \frac{s^{(k)}s^{(k)T}}{s^{(k)T}y^{(k)}} - \frac{z^{(k)}z^{(k)T}}{y^{(k)T}z^{(k)}}$$

$$s^{(k)} = \alpha_k d^{(k)} = x^{(k+1)} - x^{(k)}$$

$$y^{(k)} = \nabla f\left(x^{(k+1)}\right) - \nabla f\left(x^{(k)}\right)$$

$$z^{(k)} = A^{(k)}y^{(k)}$$

$$x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$$

$$H^{(k)}d^{(k)} = -\nabla f\left(x^{(k)}\right)$$

$$H^{(k+1)} = H^{(k)} + \frac{y^{(k)}y^{(k)T}}{y^{(k)T}s^{(k)}} - \frac{H^{(k)}s^{(k)}s^{(k)T}H^{(k)}}{s^{(k)T}H^{(k)}s^{(k)}}$$

$$\xrightarrow[H^{(k)}s^{(k)}=-\alpha_k c^{(k)}]{s^{(k)}=x^{(k+1)}-x^{(k)}=\alpha_k d^{(k)}} H^{(k+1)} = H^{(k)} + \frac{y^{(k)}y^{(k)T}}{y^{(k)T}s^{(k)}} + \frac{c^{(k)}c^{(k)T}}{c^{(k)T}d^{(k)}}$$

$$s^{(k)} = \alpha_k d^{(k)} = x^{(k+1)} - x^{(k)}$$

$$y^{(k)} = c^{(k+1)} - c^{(k)} = \nabla f\left(x^{(k+1)}\right) - \nabla f\left(x^{(k)}\right)$$

Gradient-Based Methods

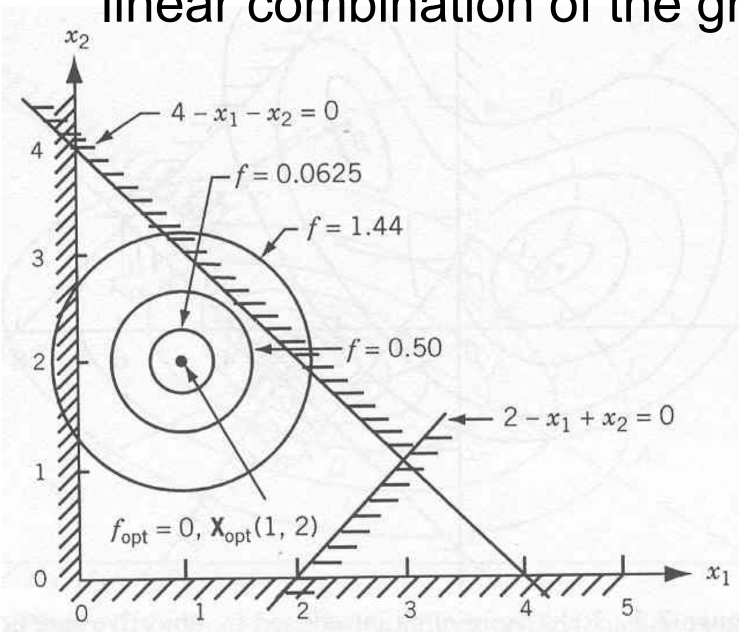
Method	Direction
Steepest Descent	$\boldsymbol{d}^{(k)} = -\nabla f\left(\boldsymbol{x}^{(k)}\right)$
Conjugate Gradient	$\boldsymbol{d}^{(k)} = -\nabla f\left(\boldsymbol{x}^{(k)}\right) + \beta_k \boldsymbol{d}^{(k-1)}$ where $\beta_k = \left\Vert \nabla f\left(\boldsymbol{x}^{(k)}\right)\right\Vert^2 \Big/ \left\Vert \nabla f\left(\boldsymbol{x}^{(k-1)}\right)\right\Vert^2$
Newton's	$\boldsymbol{d}^{(k)} = -\boldsymbol{H}^{-1}\nabla f\left(\boldsymbol{x}^{(k)}\right)$
Quasi-Newton	DFP: $\boldsymbol{d}^{(k)} = -\boldsymbol{A}\nabla f\left(\boldsymbol{x}^{(k)}\right)$ where $A^{(k+1)} = A^{(k)} + \frac{s^{(k)} s^{(k)T}}{s^{(k)T} y^{(k)}} - \frac{z^{(k)} z^{(k)T}}{y^{(k)T} z^{(k)}}$ BFGS: $\boldsymbol{H}^{(k)}\boldsymbol{d}^{(k)} = -\nabla f\left(\boldsymbol{x}^{(k)}\right)$ where $H^{(k+1)} = H^{(k)} + \frac{y^{(k)} y^{(k)T}}{y^{(k)T} s^{(k)}} + \frac{c^{(k)} c^{(k)T}}{c^{(k)T} d^{(k)}}$

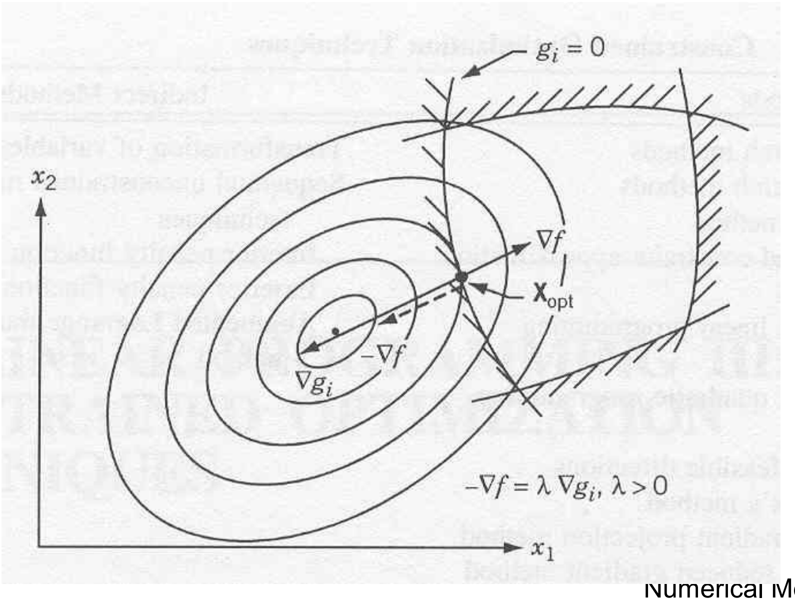
Constrained Optimization Methods

Direct (Primal) Methods	Indirect Methods
▪ Objective and constraint approximation methods – Sequential Linear Programming method – Sequential Quadratic Programming method ▪ Gradient Projection Method ▪ Methods of Feasible Directions ▪ Generalized Reduced Gradient Method	▪ Sequential unconstrained minimization technique – Interior penalty function method – Exterior penalty function method – Augmented Lagrange multiplier method

Characteristics of a Constrained Problem (1)

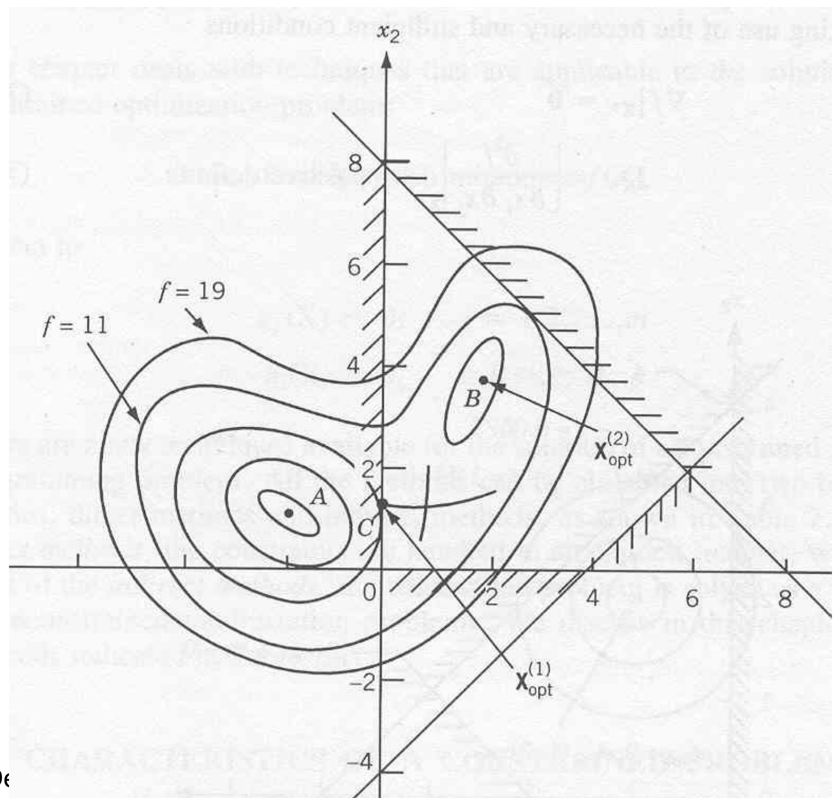
- The constraints may have no effect on the optimum point.
  - In most practical problems, it is difficult to identify whether the constraints have an influence on the minimum point.
- The optimum (unique) solution occurs on a constraint boundary.
  - The negative of the gradient must be expressible as a positive linear combination of the gradients of the active constraints.

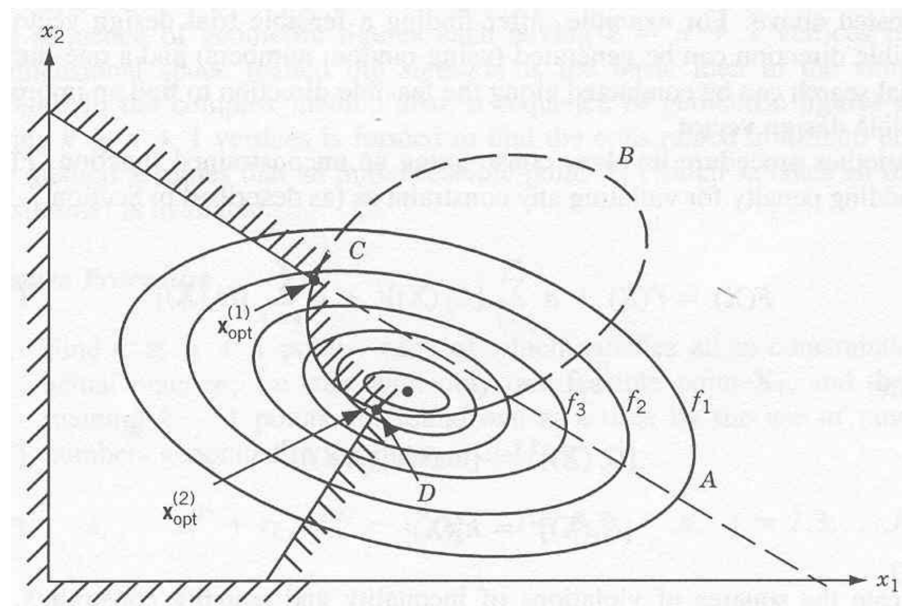




Characteristics of a Constrained Problem (2)

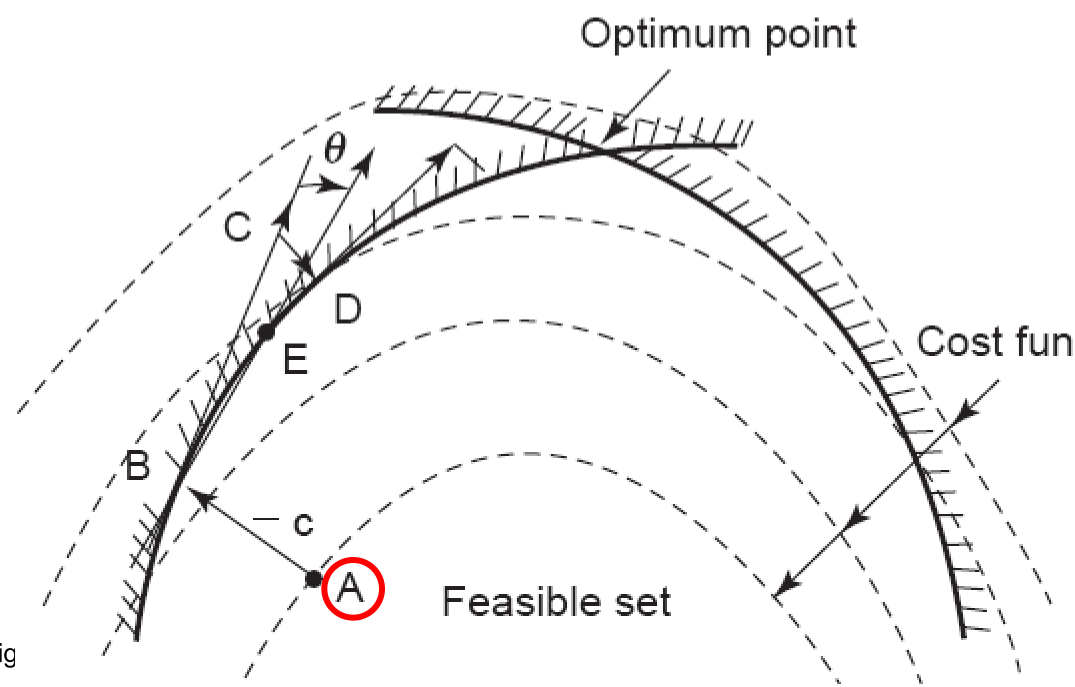
– If the objective function has two or more unconstrained local minima, the constrained problem may have multiple minima.
– Even if the objective function has a single unconstrained minimum, the constraints may introduce multiple local minima.





Basic Concepts (1)

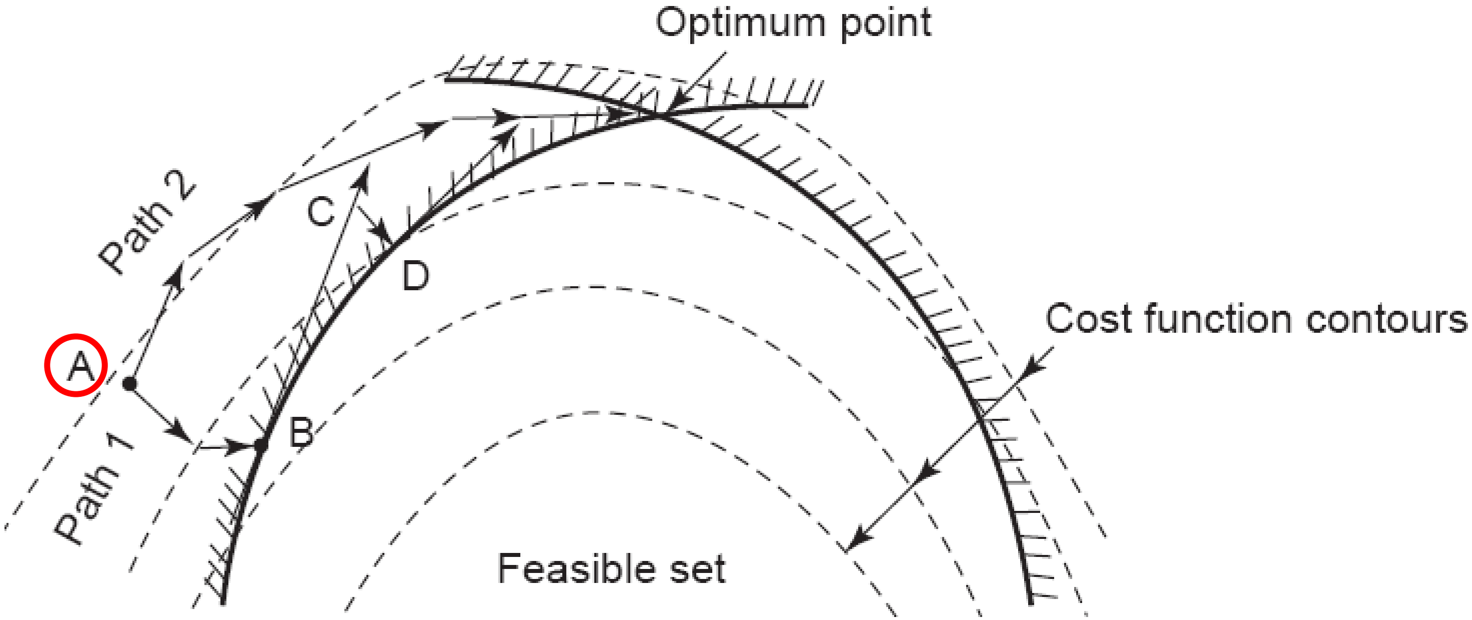
- From feasible starting point (inside the feasible region)
  - $\nabla f = 0$: Unconstrained stationary point→check sufficient condition
  - $\nabla f \neq 0$: Moving along a descent direction
    - (Assume the optimum is on the boundary of the constraint set)
    - Travel along a tangent to the boundary →correct to a feasible point
    - Deflect the tangential direction, toward the feasible region →line search



$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \Delta \boldsymbol{x}^{(k)} \\ &= \boldsymbol{x}^{(k)} + \alpha_k d^{(k)} \end{aligned}$$

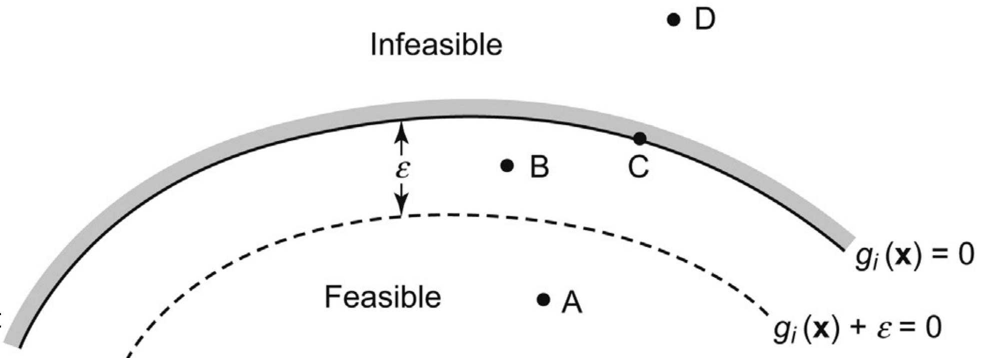
Basic Concepts (2)

- From infeasible starting point
  - Correct constraints to reach the constraint boundary →same as previous steps
  - Iterate through the infeasible region to the optimum point



Basic Concepts (3)

- Numerical algorithm
  - Linearization of cost and constraint functions about the current design point
  - Definition of a search direction determination subproblem using the linearized functions
  - Solution of the subproblem that gives a search direction in the design space.
  - Calculation of a step size to minimize a descent function in the search direction
- Constraint status @ a design point
  - Active / Inactive / Violated / ε–Active

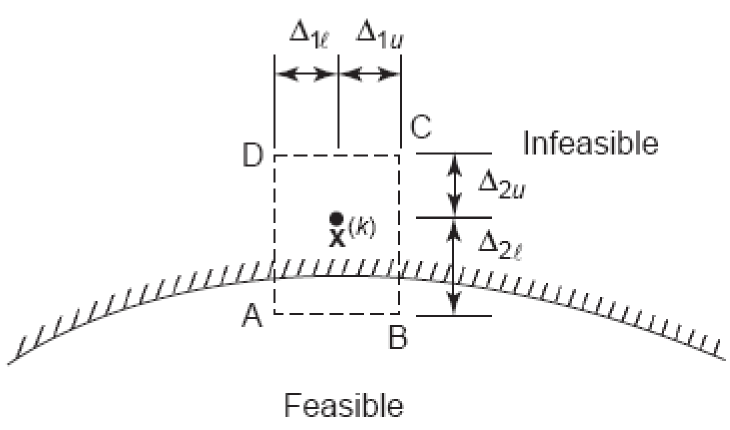


Sequential Linear Programming

- Basic idea
  - Use linear approximation of the nonlinear functions and apply standard linear programming techniques
  - Repeated process successively as the optimization process
  - Major concern: How far from the point of interest are these approximations valid? move limits: depend on degree of nonlinearity)

$$-\Delta_{il}^{(k)} \le d_i \le \Delta_{iu}^{(k)}, \quad i = 1, \ldots, n$$

  - Some fraction of the current design variables (1~100%)
- Quite powerful and efficient for engineering design



Linearization

$$\min \quad f\left(\boldsymbol{x}^{(k)}+\Delta\boldsymbol{x}^{(k)}\right)\cong f\left(\boldsymbol{x}^{(k)}\right)+\nabla f^{T}\left(\boldsymbol{x}^{(k)}\right)\Delta\boldsymbol{x}^{(k)}$$

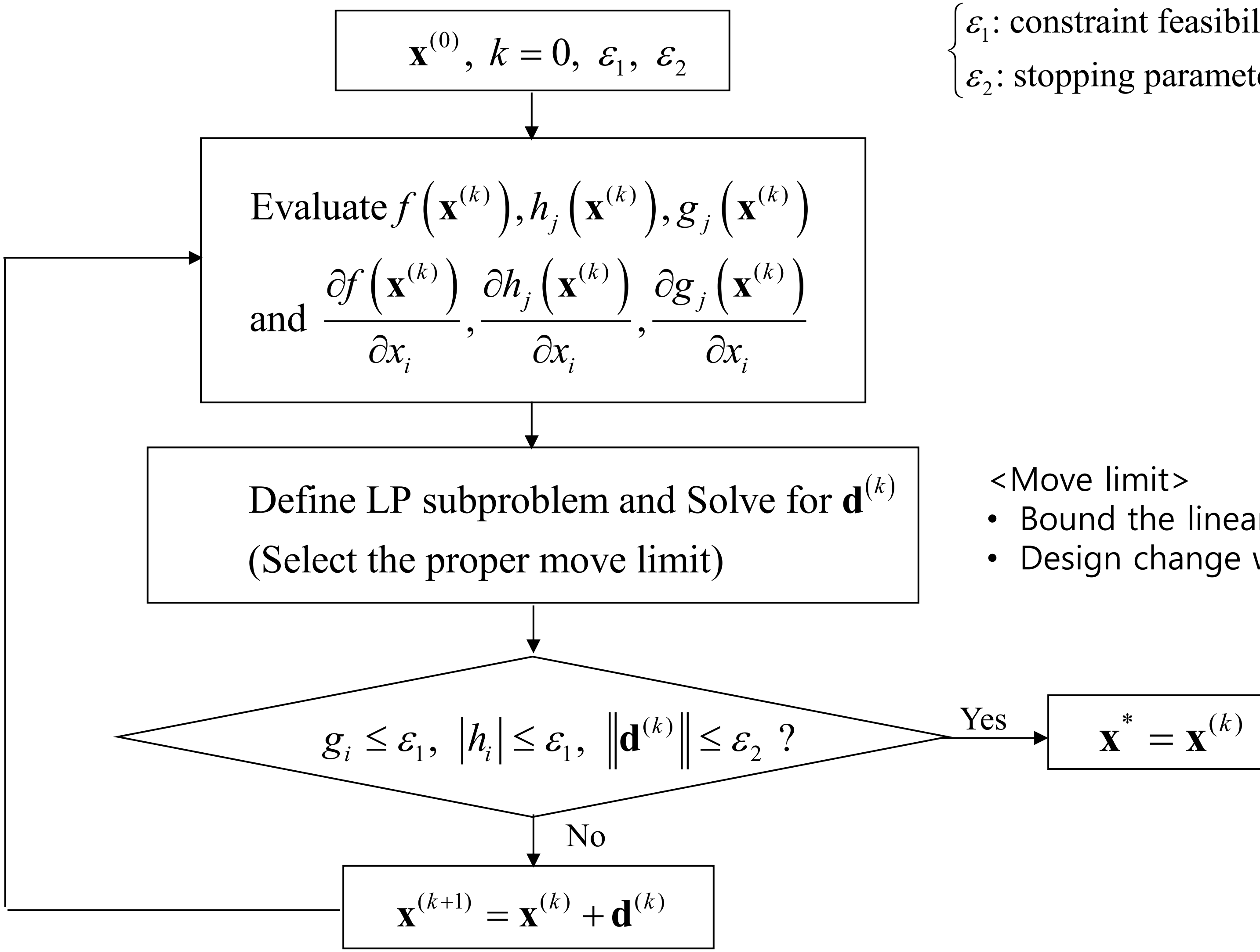
$$\text{subject to} \quad h_j\left(\boldsymbol{x}^{(k)}+\Delta\boldsymbol{x}^{(k)}\right)\cong h_j\left(\boldsymbol{x}^{(k)}\right)+\nabla h_j^{T}\left(\boldsymbol{x}^{(k)}\right)\Delta\boldsymbol{x}^{(k)}=0, \quad j=1,\ldots,p$$

$$g_j\left(\boldsymbol{x}^{(k)}+\Delta\boldsymbol{x}^{(k)}\right)\cong g_j\left(\boldsymbol{x}^{(k)}\right)+\nabla g_j^{T}\left(\boldsymbol{x}^{(k)}\right)\Delta\boldsymbol{x}^{(k)}\le 0, \quad j=1,\ldots,m$$

LP subproblem

$$\left.\begin{aligned}
&\min \quad \bar{f}=\sum_{i=1}^{n}\frac{\partial f\left(\boldsymbol{x}^{(k)}\right)}{\partial x_i}\Delta\boldsymbol{x}^{(k)}\\
&\text{s. t.} \quad \sum_{i=1}^{n}\frac{\partial h_j\left(\boldsymbol{x}^{(k)}\right)}{\partial x_i}\Delta\boldsymbol{x}^{(k)}=-h_j\left(\boldsymbol{x}^{(k)}\right)\\
&\qquad \sum_{i=1}^{n}\frac{\partial g_j\left(\boldsymbol{x}^{(k)}\right)}{\partial x_i}\Delta\boldsymbol{x}^{(k)}\le -g_j\left(\boldsymbol{x}^{(k)}\right)
\end{aligned}\right\}\rightarrow\left\{\begin{aligned}
&\min \quad \bar{f}=\sum_{i=1}^{n}c_i d_i\\
&\text{s. t.} \quad \sum_{i=1}^{n}n_{ij}d_i=e_j\\
&\qquad \sum_{i=1}^{n}a_{ij}d_i\le b_j
\end{aligned}\right\}\rightarrow\left\{\begin{aligned}
&\min \quad \bar{f}=\boldsymbol{c}^T\boldsymbol{d}\\
&\text{s. t.} \quad \underbrace{\boldsymbol{N}}_{(n\times p)}{}^{T}\boldsymbol{d}=\boldsymbol{e}\\
&\qquad \underbrace{\boldsymbol{A}}_{(n\times m)}{}^{T}\boldsymbol{d}\le\boldsymbol{b}
\end{aligned}\right.$$

SLP Algorithm



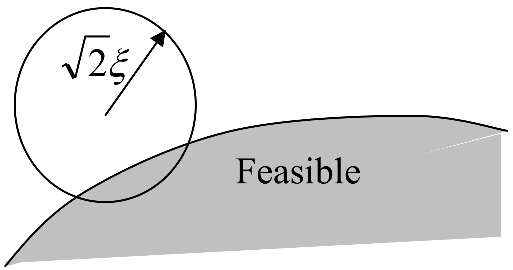
ε_1: constraint feasibility parameter
ε_2: stopping parameter

<Move limit>
- Bound the linearized subproblem
- Design change w/o line search

Quadratic Programming Subproblem

- Quadratic cost function + linear constraints
- SLP: linear move limits → quadratic step size constraint

$$-\Delta_{il}^{(k)} \le d_i \le \Delta_{iu}^{(k)} \rightarrow \|d\| \le \xi \rightarrow 0.5\sum_{i=1}^{n}(d_i)^2 \le \xi^2$$



$$\left.\begin{aligned} &\min \quad \bar{f} = \sum_{i=1}^{n} c_i d_i \\ &\text{s. t.} \quad \sum_{i=1}^{n} n_{ij} d_i = e_j, \quad j = 1,\ldots,p \\ &\qquad \sum_{i=1}^{n} a_{ij} d_i \le b_j, \quad j = 1,\ldots,m \\ &\qquad 0.5\sum_{i=1}^{n}(d_i)^2 \le \xi^2 \end{aligned}\right\} \rightarrow \left\{\begin{aligned} &\min \quad \bar{f} = \boldsymbol{c}^T\boldsymbol{d} \\ &\text{s. t.} \quad \underbrace{\boldsymbol{N}}_{(n\times p)}{}^T \boldsymbol{d} = \boldsymbol{e} \\ &\qquad \underbrace{\boldsymbol{A}}_{(n\times m)}{}^T \boldsymbol{d} \le \boldsymbol{b} \\ &\qquad 0.5\boldsymbol{d}^T\boldsymbol{d} \le \xi^2 \end{aligned}\right\} \leftrightarrow \left\{\begin{aligned} &\min \quad \bar{f} = \boldsymbol{c}^T\boldsymbol{d} + 0.5\boldsymbol{d}^T\boldsymbol{d} \\ &\text{s. t.} \quad \boldsymbol{N}^T\boldsymbol{d} = \boldsymbol{e} \\ &\qquad \boldsymbol{A}^T\boldsymbol{d} \le \boldsymbol{b} \end{aligned}\right.$$

Strictly convex →
Minimum is global and unique

$$(d_1 + c_1)^2 + (d_2 + c_2)^2 = r^2 \rightarrow d_1^2 + c_1^2 + 2c_1 d_1 + d_2^2 + c_2^2 + 2c_2 d_2 = r^2$$

$$\frac{1}{2}\left(r^2 - c_1^2 - c_2^2\right) = c_1 d_1 + c_2 d_2 + \frac{1}{2}\left(d_1^2 + d_2^2\right)$$: hypersphere with its center at $-\mathbf{c}$

Sequential Quadratic Programming (SQP)

- QP subproblem $\leftarrow$ curvature information of Lagrange function into the quadratic cost function
  - Constrained Quasi-Newton Methods
  - Constrained Variable Metric(CVM)
  - Recursive Quadratic Programming(RQP)
- Gradient of the Lagrange function at the two points $\rightarrow$ Approximate Hessian of the Lagrange function
- quite simple and straightforward, but very effective

Generalized Reduced Gradient Method

- Elimination of variables using the equality constraints
  - One variable can be reduced from the set x_i for each of the m+p equality constraints

$$\left.\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & g_i(\boldsymbol{x}) \le 0, \quad i = 1, \ldots, m \\ & h_j(\boldsymbol{x}) = 0, \quad j = 1, \ldots, l \\ & x_k^L \le x_k \le x_k^U, \quad k = 1, \ldots, n \end{array}\right\} \rightarrow \left\{\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & g_i(\boldsymbol{x}) + x_{n+i} = 0, \quad i = 1, \ldots, m \\ & h_j(\boldsymbol{x}) = 0, \quad j = 1, \ldots, l \\ & x_k^L \le x_k \le x_k^U, \quad k = 1, \ldots, n \\ & x_{n+i} \ge 0, \quad i = 1, \ldots, m \end{array}\right.$$

$$\rightarrow \left\{\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \bar{h}_j(\boldsymbol{x}) = 0, \quad j = 1, \ldots, m + l \\ & x_i^L \le x_i \le x_i^U, \quad i = 1, \ldots, n + m \end{array}\right.$$

$$\boldsymbol{x} = \left\{\begin{matrix} \boldsymbol{y} \\ \boldsymbol{z} \end{matrix}\right\}, \quad \underbrace{\boldsymbol{y} = \left\{\begin{matrix} y_1 \\ \vdots \\ y_{m+l} \end{matrix}\right\}}_{\text{state or dependent variables}}, \quad \underbrace{\boldsymbol{z} = \left\{\begin{matrix} z_1 \\ \vdots \\ z_{n-l} \end{matrix}\right\}}_{\text{design or independent variables}}$$

Reduced Gradient

$$df(\boldsymbol{x}) = \sum_{i=1}^{m+l} \frac{\partial f}{\partial y_i} dy_i + \sum_{i=1}^{n-l} \frac{\partial f}{\partial z_i} dz_i = \nabla_y^T f d\boldsymbol{y} + \nabla_z^T f d\boldsymbol{z}$$

$$d\bar{h}_i(\boldsymbol{x}) = \sum_{j=1}^{m+l} \frac{\partial \bar{h}_i}{\partial y_j} dy_j + \sum_{j=1}^{n-l} \frac{\partial \bar{h}_i}{\partial z_j} dz_j \rightarrow d\bar{\boldsymbol{h}} = \boldsymbol{B} d\boldsymbol{y} + \boldsymbol{C} d\boldsymbol{z}$$

$$\nabla_y^T f = \begin{Bmatrix} \dfrac{\partial f}{\partial y_1} \\ \vdots \\ \dfrac{\partial f}{\partial y_{m+l}} \end{Bmatrix}, \nabla_z^T f = \begin{Bmatrix} \dfrac{\partial f}{\partial z_1} \\ \vdots \\ \dfrac{\partial f}{\partial z_{n-l}} \end{Bmatrix}, d\boldsymbol{y} = \begin{Bmatrix} dy_1 \\ \vdots \\ dy_{m+l} \end{Bmatrix}, d\boldsymbol{z} = \begin{Bmatrix} dz_1 \\ \vdots \\ dz_{n-l} \end{Bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} \dfrac{\partial \bar{h}_1}{\partial y_1} & \cdots & \dfrac{\partial \bar{h}_1}{\partial y_{m+l}} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial \bar{h}_{m+l}}{\partial y_1} & \cdots & \dfrac{\partial \bar{h}_{m+l}}{\partial y_{m+l}} \end{bmatrix}, \ \boldsymbol{C} = \begin{bmatrix} \dfrac{\partial \bar{h}_1}{\partial z_1} & \cdots & \dfrac{\partial \bar{h}_1}{\partial z_{n-l}} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial \bar{h}_{m+l}}{\partial z_1} & \cdots & \dfrac{\partial \bar{h}_{m+l}}{\partial y_{n-l}} \end{bmatrix}$$

GRG: Direction

$$d\bar{\boldsymbol{h}} = \boldsymbol{B}d\boldsymbol{y} + \boldsymbol{C}d\boldsymbol{z} = 0 \ \left(\bar{\boldsymbol{h}}\left(\boldsymbol{x}\right) = 0\right) \rightarrow d\boldsymbol{y} = -\boldsymbol{B}^{-1}\boldsymbol{C}d\boldsymbol{z}$$

$$df\left(\boldsymbol{x}\right) = \left(-\nabla_y^T f \boldsymbol{B}^{-1}\boldsymbol{C} + \nabla_z^T f\right)d\boldsymbol{z} \rightarrow \frac{df\left(\boldsymbol{x}\right)}{d\boldsymbol{z}} = \boldsymbol{G}_R$$

$$\boldsymbol{G}_R = \nabla_z f - \left(\boldsymbol{B}^{-1}\boldsymbol{C}\right)^T \nabla_y f$$: generalized reduced gradient

→ projection of the original n-dimensional gradient onto the $(n - m)$ dimensional feasible region described by the design variables

$$\boldsymbol{d} = \begin{bmatrix} \boldsymbol{d}_y \\ \boldsymbol{d}_z \end{bmatrix} \rightarrow \begin{cases} \boldsymbol{d}_y = -\boldsymbol{B}^{-1}\boldsymbol{C}\boldsymbol{d}_z \\ \left(\boldsymbol{d}_z\right)_i = \begin{cases} -\left(\boldsymbol{G}_R\right)_i \\ 0 \quad \text{if} \quad z_i = z_i^L \quad \text{and} \quad \left(\boldsymbol{G}_R\right)_i > 0 \\ 0 \quad \text{if} \quad z_i = z_i^U \quad \text{and} \quad \left(\boldsymbol{G}_R\right)_i < 0 \end{cases} \end{cases}$$