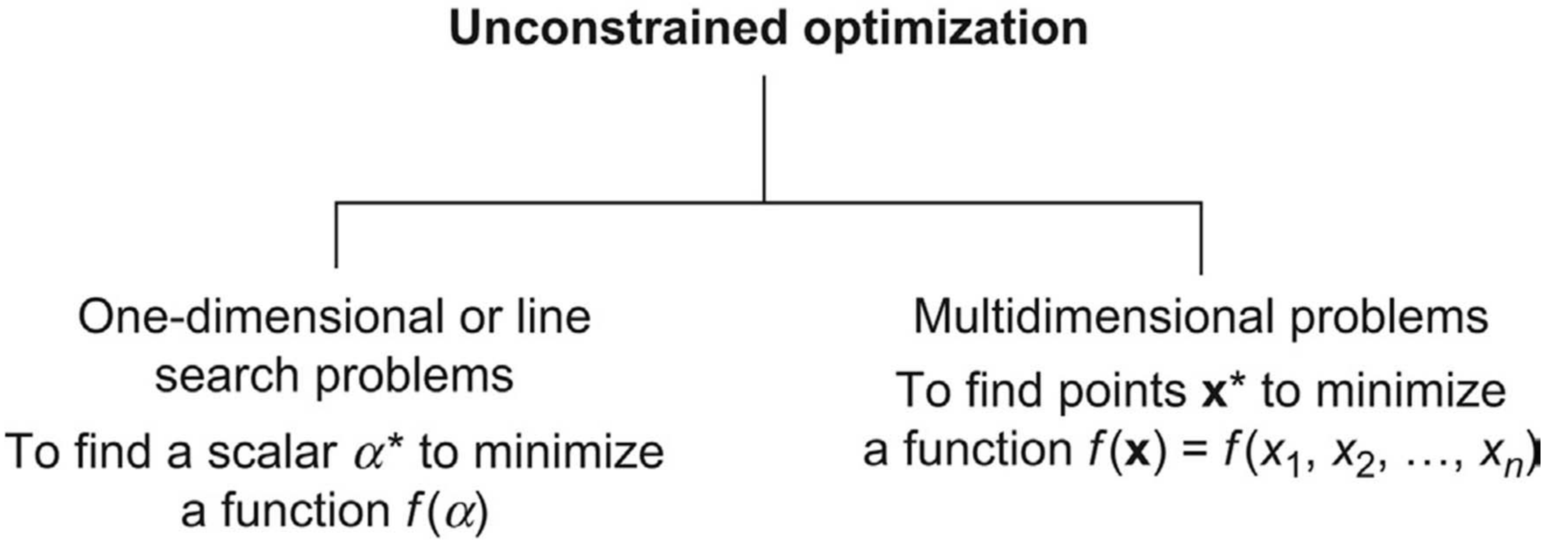
Contents

General Concepts

- Derivative(or gradient)-based search methods
  - estimate an initial design
  - improve it iteratively, until optimality conditions are satisfied



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/

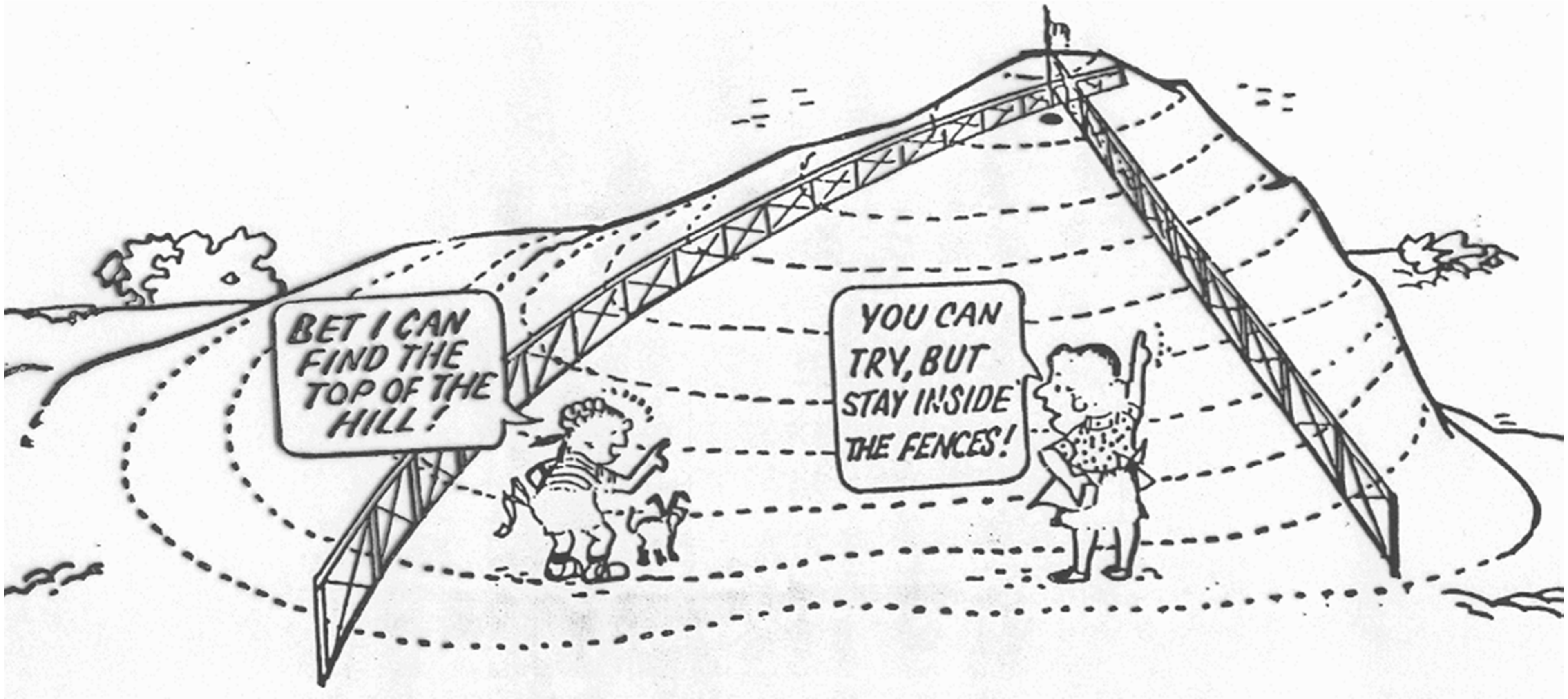
Advantages of Numerical Optimization

- 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

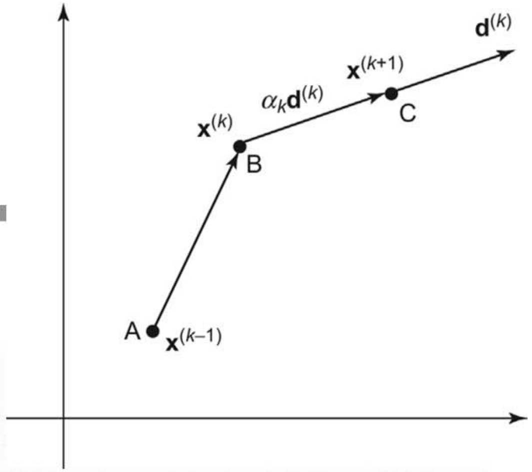
Limitations of Numerical Optimization

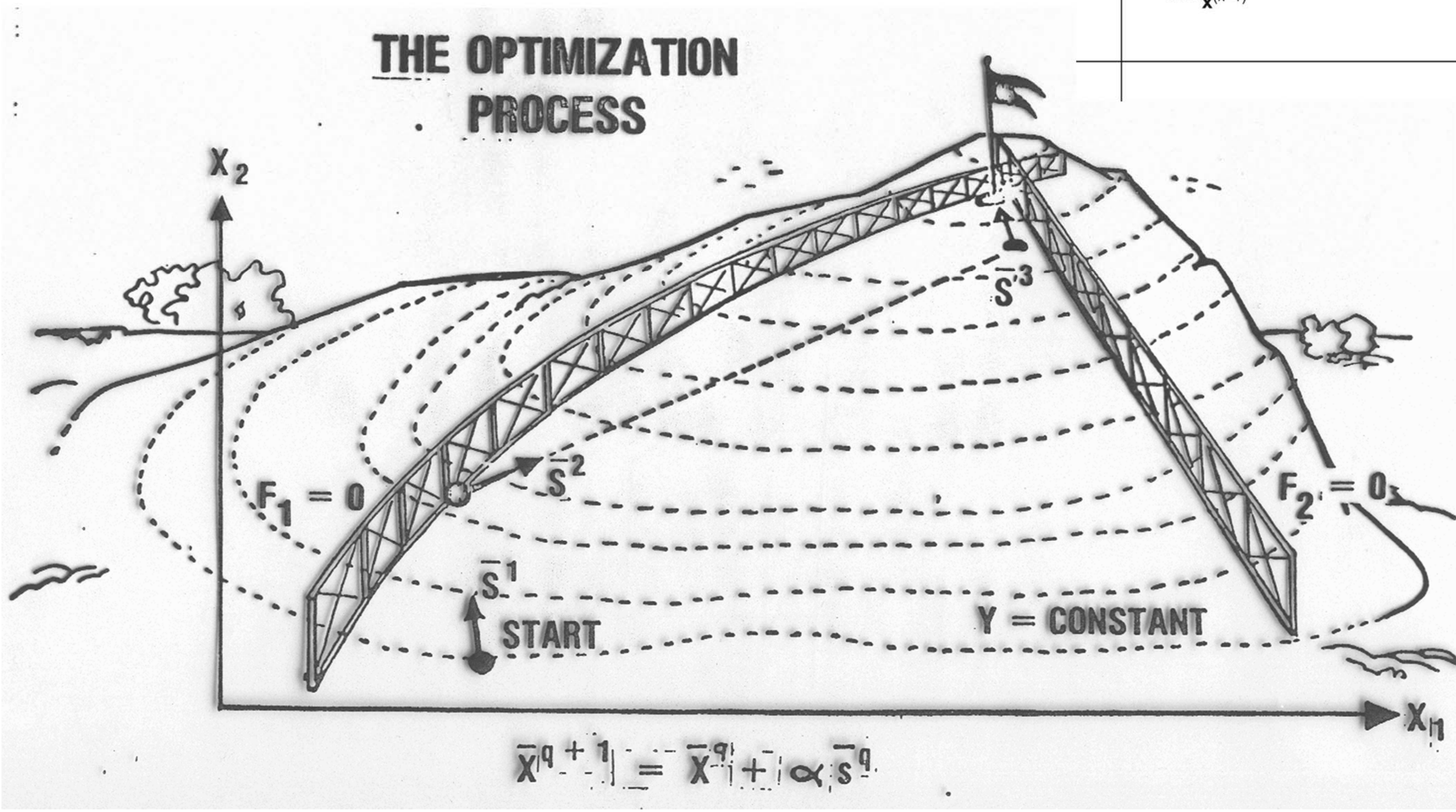
- Increased computational time as the number of design variables increases (ill-conditioned?)
- No stored experience or intuition
- Misleading results if the analysis program is not theoretically precise
- Difficulty in dealing with discontinuous functions and highly nonlinear problems
- Seldom be guaranteed 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





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?

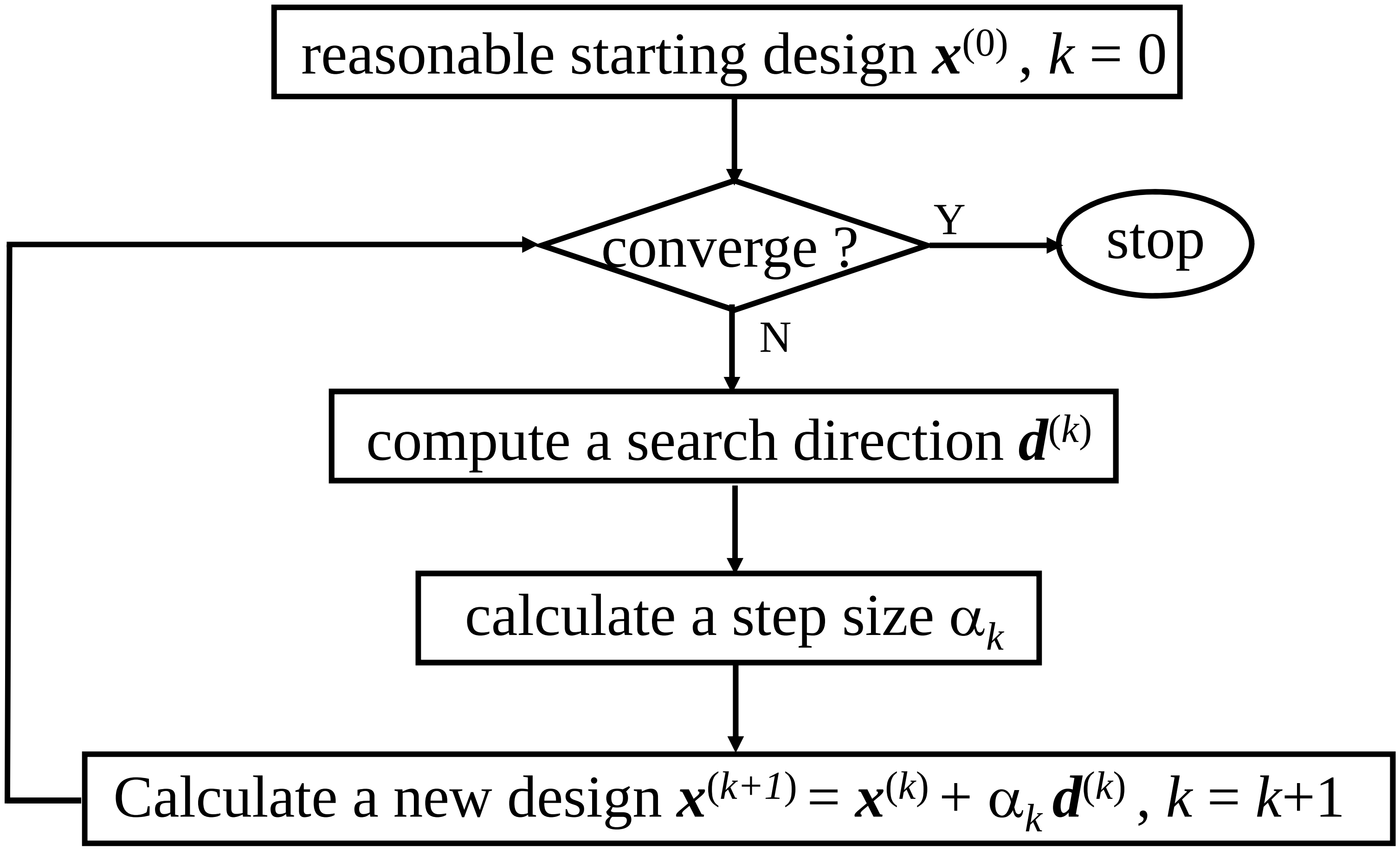
Basic Steps in Nonlinear Optimization

- 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



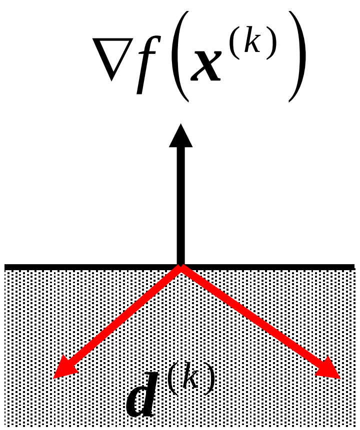
Descent Direction

- Desirable direction of design change in the iterative process: directions of descent for the cost function
- Descent condition

$$f\left(\boldsymbol{x}^{(k+1)}\right) < f\left(\boldsymbol{x}^{(k)}\right)$$
$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}$$
$$f\left(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}\right) < f\left(\boldsymbol{x}^{(k)}\right)$$

linear Taylor series expansion

$$f\left(\boldsymbol{x}^{(k)}\right) + \alpha_k \left(\nabla f\left(\boldsymbol{x}^{(k)}\right)\cdot \boldsymbol{d}^{(k)}\right) < f\left(\boldsymbol{x}^{(k)}\right)$$
$$\alpha_k \left(\nabla f\left(\boldsymbol{x}^{(k)}\right)\cdot \boldsymbol{d}^{(k)}\right) < 0 \quad [\alpha_k > 0]$$
$$\nabla f\left(\boldsymbol{x}^{(k)}\right)\cdot \boldsymbol{d}^{(k)} < 0$$



Ex. $f(x) = x_1{}^2 - x_1 x_2 + 2x_2{}^2 - 2x_1 + e^{(x_1 + x_2)}$

$\boldsymbol{d} = (1,2)$ at the point (0,0) is a decent direction?

Gradients Evaluation (1)

- Finite Differences
  - Forward difference

$$f(x_{i+1}) = f(x_i) + f'(x_i)\underbrace{(x_{i+1}-x_i)}_{h} + \frac{f''(x_i)}{2!}\underbrace{(x_{i+1}-x_i)}_{h}^2 + \cdots \rightarrow f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + \underbrace{\frac{f''(x_i)}{2}h + \cdots}_{O(h)}$$

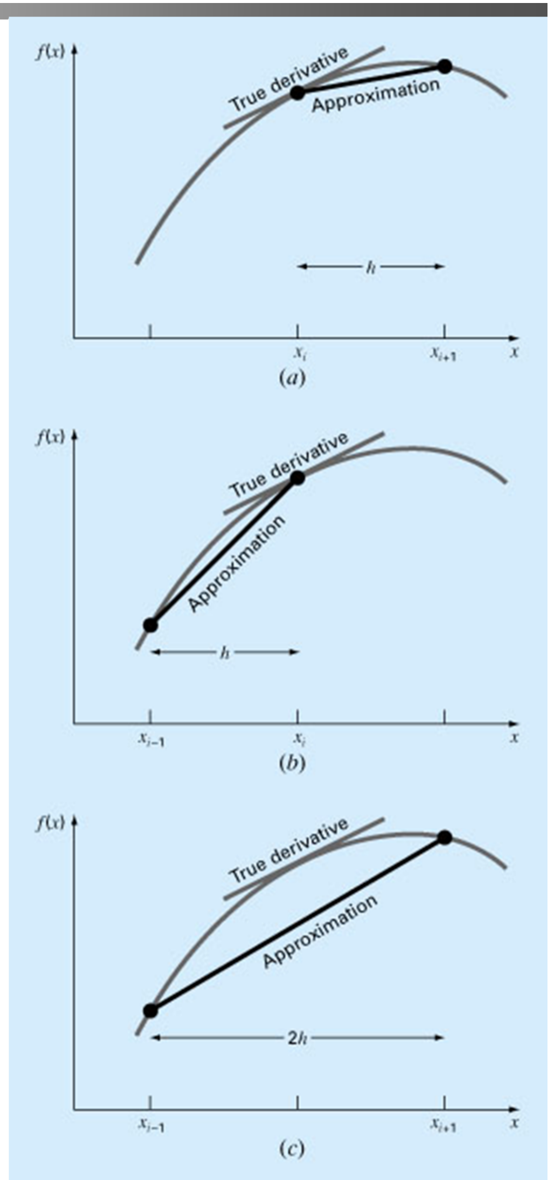
  - Backward difference

$$f(x_{i-1}) = f(x_i) + f'(x_i)\underbrace{(x_{i-1}-x_i)}_{-h} + \frac{f''(x_i)}{2!}\underbrace{(x_{i-1}-x_i)}_{-h}^2 + \cdots \rightarrow f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + \underbrace{\frac{f''(x_i)}{2}h + \cdots}_{O(h)}$$

  - Central difference
    - Error↓ # of function evaluation↑
    - Perturbation?
    - If the function is not too nonlinear, $h = 0.01|x_i|$

$$\begin{cases} f(x_{i+1}) = f(x_i) + f'(x_i)\underbrace{(x_{i+1}-x_i)}_{h} + \frac{f''(x_i)}{2!}\underbrace{(x_{i+1}-x_i)}_{h}^2 + \frac{f'''(x_i)}{3!}\underbrace{(x_{i+1}-x_i)}_{h}^3 + \cdots \\ f(x_{i-1}) = f(x_i) + f'(x_i)\underbrace{(x_{i-1}-x_i)}_{-h} + \frac{f''(x_i)}{2!}\underbrace{(x_{i-1}-x_i)}_{-h}^2 + \frac{f'''(x_i)}{3!}\underbrace{(x_{i-1}-x_i)}_{-h}^3 + \cdots \end{cases}$$

$$\rightarrow f(x_{i+1}) - f(x_{i-1}) = 2hf'(x_i) + \frac{f'''(x_i)}{3}h^3 \rightarrow f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + O(h^2)$$



Gradients Evaluation (2)

- Automatic Differentiation
  - Computer code for evaluating the function can be broken down into elementary arithmetic operations (chain rule)
  - ADIFOR, ADOL-C
- Symbolic Differentiation
  - Algebraic specification for the function is manipulated by symbolic manipulation tools
  - Mathematica, Maple, Macsyma
- Usefulness of derivatives
  - Algorithms for optimization
  - Post-optimal sensitivity analysis

A Good Algorithm

- Robust: algorithm must be reliable for general design applications and must theoretically converge to the solution point starting from any given point
- General: should not impose restrictions on the model's constraints and objective functions
- Accurate: ability to converge to precise mathematical optimum point is important, though it may not be required in practice
- Easy to use: by both experienced and inexperienced users. Should not have problem dependent tuning parameters
- Efficient: the number of repeated analysis should be kept to a minimum. 1) fast rate of convergence requiring fewer iterations 2) least number of calculations within one iteration

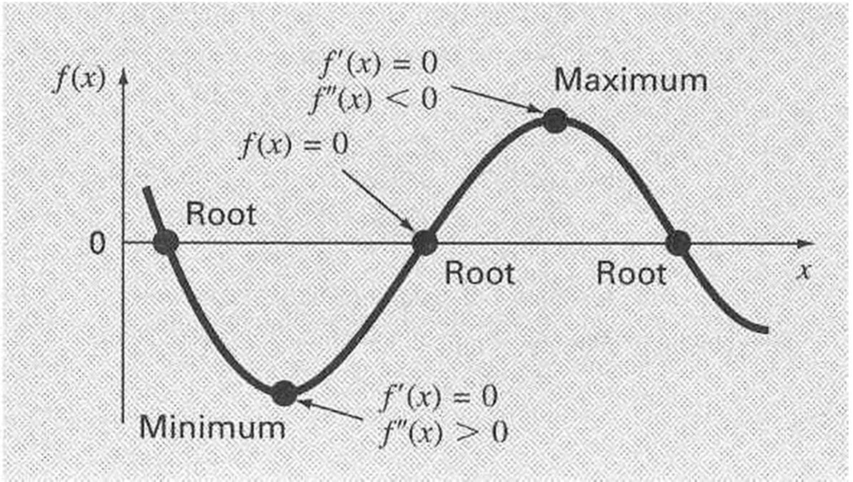
Classification of Unconstrained Optimization

- 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

One-dimensional Unconstrained Optimization

- Function of a single variable
  - "roller coaster"-like function: multimodal
- Bracketing method
  - Golden-section search
  - Quadratic interpolation
- Open method
  - Newton method: $f'(x) = 0$



- Roots and Optima
  - Guess and search for a point on a function
    - Root location: zeros of a function or functions
    - Optimization: either the minimum or the maximum

One-Dimensional Search

– Assume that the desirable direction $\boldsymbol{d}^{(k)}$ has been found

$$f\left(\boldsymbol{x}^{(k+1)}\right) = f\left(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}^{(k)}\right) \equiv \bar{f}(\alpha)$$

– $\bar{f}(0) = f\left(\boldsymbol{x}^{(k)}\right) \; @ \alpha = 0$: current value of the cost function

– If $\mathbf{x}^{(k)}$ is not a minimum point,

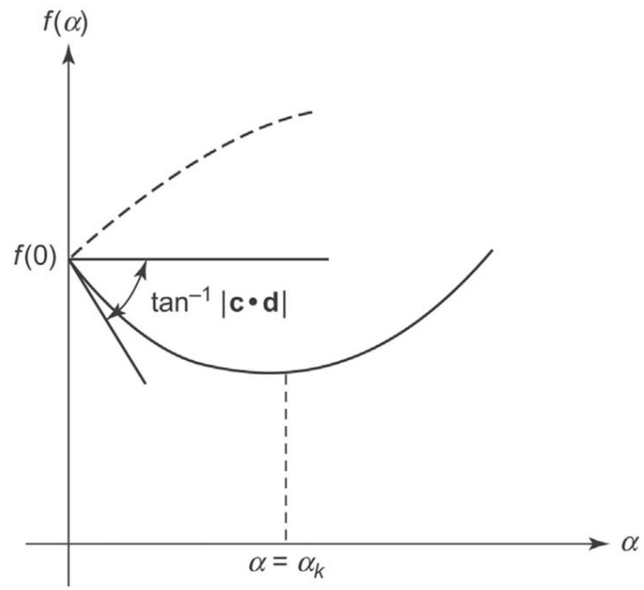
$$\left[\bar{f}(\alpha) = \right] f\left(\boldsymbol{x}^{(k+1)}\right) < f\left(\boldsymbol{x}^{(k)}\right)\left[= \bar{f}(0)\right] \rightarrow \bar{f}(\alpha) < \bar{f}(0)$$

$$\rightarrow \text{negative slope} \; @ \alpha = 0 \rightarrow \bar{f}'(0) < 0$$

$$\bar{f}'(0) = \left.\frac{\partial f\left(\boldsymbol{x}^{(k+1)}\right)}{\partial \alpha}\right|_{\alpha=0} = \left.\frac{\partial f^T\left(\boldsymbol{x}^{(k+1)}\right)}{\partial x}\right|_{\alpha=0} \frac{d\boldsymbol{x}^{(k+1)}}{d\alpha}$$

$$= \nabla f\left(\boldsymbol{x}^{(k)}\right) \cdot \boldsymbol{d}^{(k)} < 0$$

$$\rightarrow \text{descent direction confirmed!!!}$$



Step Size Determination (1)

- Analytical method
  - If $\boldsymbol{d}^{(k)}$ is a descent direction, then α must be a positive scalar
  - Find α such that $f(\alpha)$ is minimized

$$\begin{cases} \text{necessary condition:} \\ \dfrac{\partial f(\alpha_k)}{\partial \alpha} = \dfrac{\partial f\left(\boldsymbol{x}^{(k+1)}\right)}{\partial \alpha} = \dfrac{\partial f^T\left(\boldsymbol{x}^{(k+1)}\right)}{\partial x} \dfrac{d\boldsymbol{x}^{(k+1)}}{d\alpha} \\ \qquad = \nabla f\left(\boldsymbol{x}^{(k+1)}\right) \cdot \boldsymbol{d}^{(k)} = 0 \\ \text{sufficient condition:} \dfrac{\partial^2 f(\alpha_k)}{\partial \alpha^2} > 0 \end{cases}$$

  - Gradient of the cost function at the new point is orthogonal to the search direction at the *k*-th iteration

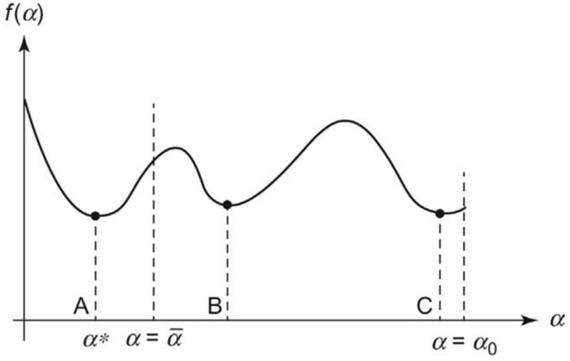
Ex. $f(x) = 3x_1^{\ 2} + 2x_1x_2 + 2x_2^{\ 2} + 7$ at the point (1,2),
step size α to minimize $f(x)$ in the given $\boldsymbol{d} = (-1,-1)$?

Step Size Determination (2)

- Numerical method
  - Consider only unimodal functions
    - Existence of a minimum / uniqueness in the interval of interest
  - Not an unimodal function?
    - Only a local minimum closest to the starting point
  - Interval of uncertainty in which the minimum lies

$$I = \alpha_u - \alpha_l < \varepsilon$$

  - Interval reducing methods (zero order)
    - Step 1: initial interval of uncertainty (bracketing)
    - Step 2: refinement of the interval of uncertainty
      - Equal Interval Search
      - Golden Section Search
      - Polynomial Interpolation

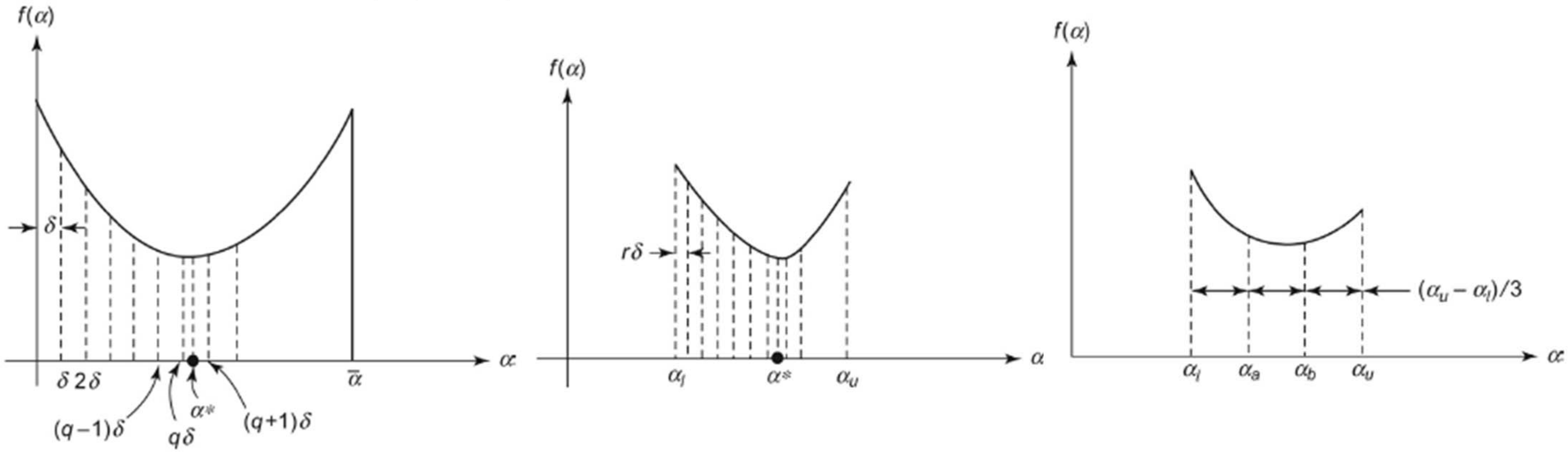


Equal Interval Search

$$\left.\begin{array}{l} <\text{bracketing}> \\ f(q\delta) < f((q+1)\delta) \rightarrow \alpha_l = (q-1)\delta \text{ and } \alpha_u = (q+1)\delta \\ I = \alpha_u - \alpha_l = 2\delta \end{array}\right\} \rightarrow \left\{\begin{array}{l} \text{Restart} \\ r\delta(r \ll 1) \\ I = 2r\delta \end{array}\right.$$

- ☹ δ dependent: inefficient bracketing
- Alternatives: two points α_a , α_b (I/3, 2I/3)

$$\left.\begin{array}{l} f(\alpha_a) < f(\alpha_b) \rightarrow \alpha_l' = \alpha_l \text{ and } \alpha_u' = \alpha_b \\ f(\alpha_a) > f(\alpha_b) \rightarrow \alpha_l' = \alpha_a \text{ and } \alpha_u' = \alpha_u \end{array}\right\} \rightarrow I \rightarrow I' = 2I/3$$



Golden Section (1)

- One of the league of the "infinite, non recurring decimal" number constants of mathematics: Pi (3.141592653589) and e (2.71828182846)
- Golden Section provides the answer to the question...
  - "Which rectangle shape is just right, neither too wide or too narrow?"

- (1) a straight line (or a rectangle) is divided into two unequal parts in such a way, that the ratio of the smaller to the greater part is the same as that of the greater part to the whole figure (AC:CB=AB:AC)

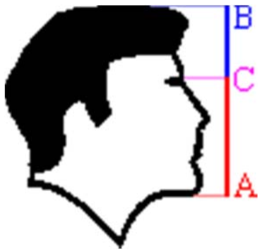
$$AB = 1, AC = x$$

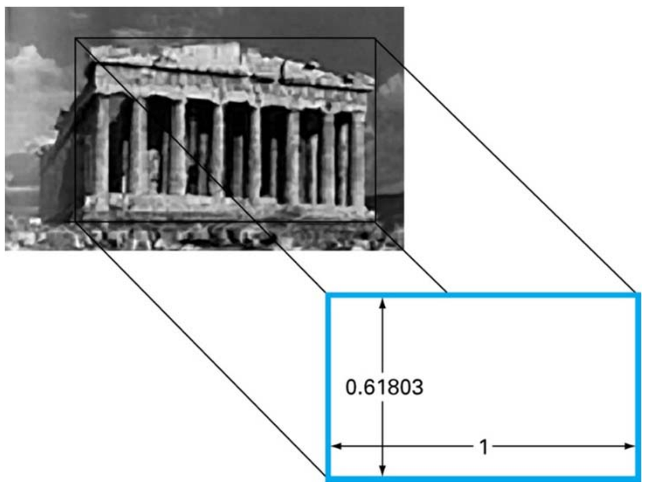
$$\frac{AC}{CB} = \frac{AB}{AC} \rightarrow \frac{x}{1-x} = \frac{1}{x} \rightarrow x^2 + x - 1 = 0 \rightarrow x = \frac{-1+\sqrt{5}}{2} = 0.61803...$$

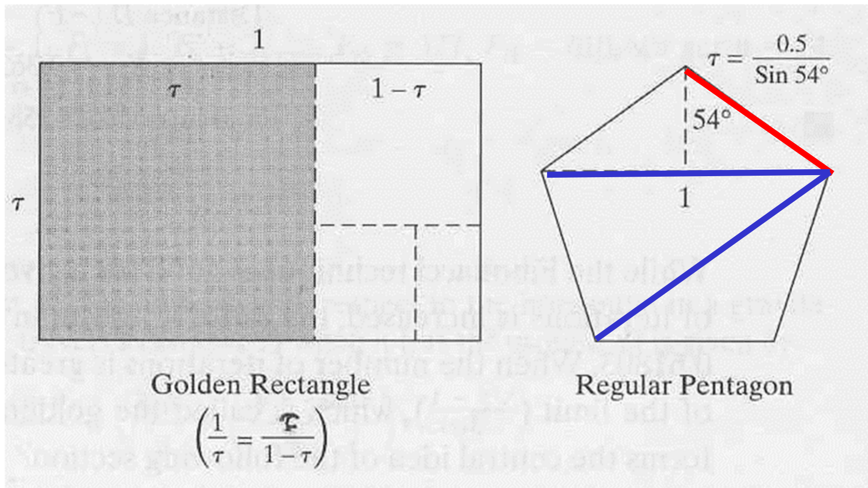
Golden Section (2)

– (2) The reciprocal of the Golden Section (0.61803398875) is 1.61803398875. $\frac{1}{x} = x + 1$

– (3) If a Golden Rectangle is cut so a square and a rectangle remains, the new rectangle will also be Golden.

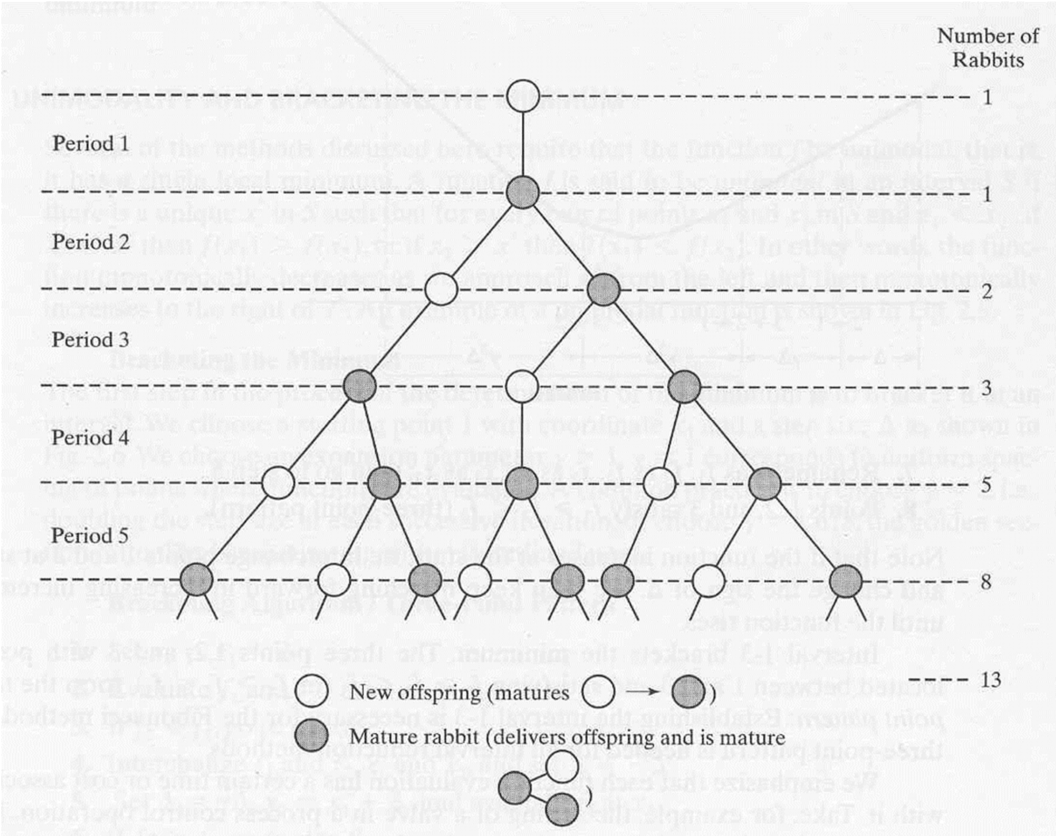






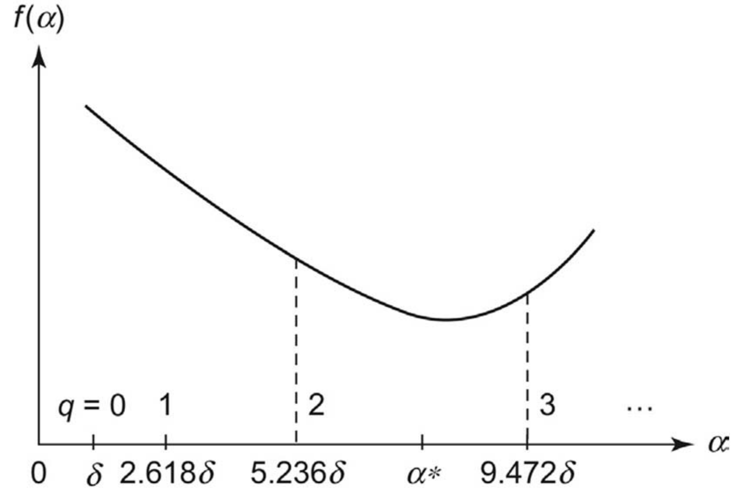
Fibonacci Sequence

- Fibonaccian numbers: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …
  - In the study of rabbit reproductions



Golden Section Search (1)

- Variable Interval Search Method
- 1) Initial bracketing of minimum
  - Rapid initial bracketing with large span (r > 1)
  - r = 1.618 : golden ratio



$$\frac{F_n}{F_{n-1}} \to 1.618 = \frac{\sqrt{5}+1}{2} \left(\text{or, } \frac{F_{n-1}}{F_n} \to 0.618 \right) \text{as } n \to \infty$$

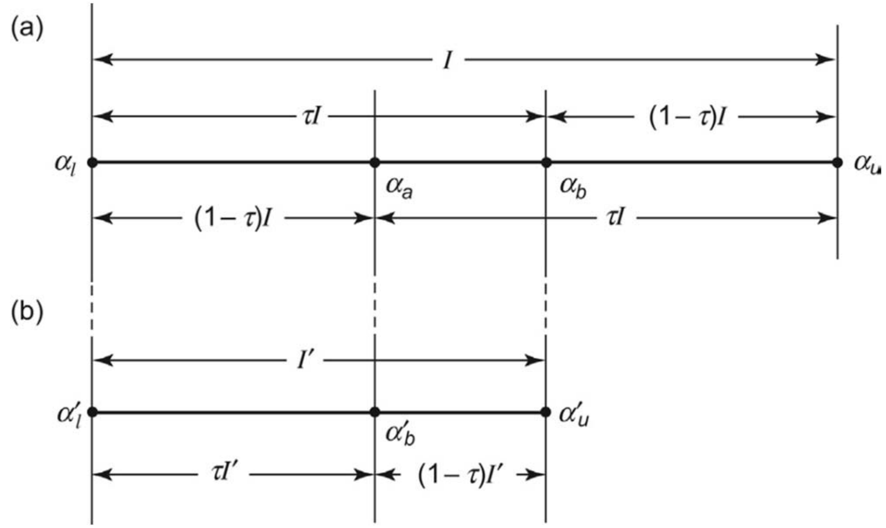
$$\alpha_q = \sum_{j=0}^{q} \delta (1.618)^j \qquad q = 0,1,2,\ldots$$

$$f(\alpha_{q-1}) < f(\alpha_{q-2}) \text{and } f(\alpha_{q-1}) < f(\alpha_q)$$

$$I = \alpha_u - \alpha_l = \sum_{j=0}^{q} \delta (1.618)^j - \sum_{j=0}^{q-2} \delta (1.618)^j = 2.618 (1.618)^{q-1} \delta$$

Golden Section Search (2)

- 2) Reduction of interval of uncertainty
  - Two points $\alpha_a = 0.382I$, $\alpha_b = 0.618I$: how?



$$\tau I' = (1-\tau)I \rightarrow \tau^2 + \tau - 1 = 0$$

$$\rightarrow \tau = \frac{-1+\sqrt{5}}{2} = 0.618$$

  - $\alpha_a = \alpha_{q-1}$: how?
    - Only one additional function evaluation is required

$$I = 2.618(1.618)^{q-1}\delta$$

$$\alpha_a = \alpha_l + 0.382I = \alpha_{q-2} + (1.618)^{q-1}\delta = \alpha_{q-1}$$

Golden Section Search (3)

$$\left(\alpha_l = \alpha_{q-2}, \alpha_{q-1}, \alpha_q = \alpha_u\right) \text{ from bracketing}$$

$$\begin{cases} \alpha_a = \alpha_l + 0.382I = \alpha_{q-1} \\ \alpha_b = \alpha_l + 0.618I \leftarrow \text{new point} \end{cases}$$

$$f\left(\alpha_a\right) < f\left(\alpha_b\right) \xrightarrow{\alpha_l, \alpha_a, \alpha_b} \begin{cases} \alpha_l = \alpha_l \\ \alpha_a = \alpha_l + 0.382\left(\alpha_b - \alpha_l\right) \\ \alpha_b = \alpha_a \\ \alpha_u = \alpha_b \end{cases}$$

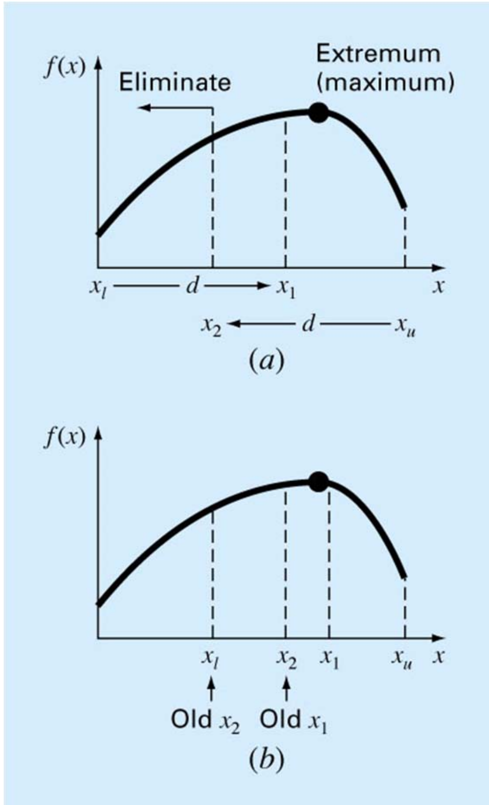
$$f\left(\alpha_a\right) > f\left(\alpha_b\right) \xrightarrow{\alpha_a, \alpha_b, \alpha_u} \begin{cases} \alpha_l = \alpha_a \\ \alpha_a = \alpha_b \\ \alpha_b = \alpha_a + 0.618\left(\alpha_u - \alpha_a\right) \\ \alpha_u = \alpha_u \end{cases}$$

$$f\left(\alpha_a\right) = f\left(\alpha_b\right) \rightarrow \begin{cases} \alpha_l = \alpha_a \\ \alpha_u = \alpha_b \end{cases}$$

Pseudocode for the golden-section algorithm

```
FUNCTION Gold (xlow, xhigh, maxit, es, fx)
R = (5^0.5 - 1)/2
xℓ = xlow; xu = xhigh
iter = 1
d  = R * (xu - xℓ)
x1 = xℓ + d; x2 = xu - d
f1 = f(x1)
f2 = f(x2)
IF f1 > f2 THEN
  xopt = x1
  fx = f1
ELSE
  xopt = x2
  fx = f2
END IF
DO
  d = R*d
  IF f1 > f2 THEN
    xℓ = x2
    x2 = x1
    x1 = xℓ+d
    f2 = f1
    f1 = f(x1)
  ELSE
    xu = x1
    x1 = x2
    x2 = xu-d
    f1 = f2
    f2 = f(x2)
  END IF
  iter = iter+1
  IF f1 > f2 THEN
    xopt = x1
    fx = f1
  ELSE
    xopt = x2
    fx = f2
  END IF
  IF xopt ≠ 0. THEN
    ea = (1.-R) *ABS((xu - xℓ)/xopt) * 100.
  END IF
  IF ea ≤ es OR iter ≥ maxit EXIT
END DO
Gold = xopt
END Gold
```

(*a*) **Maximization**

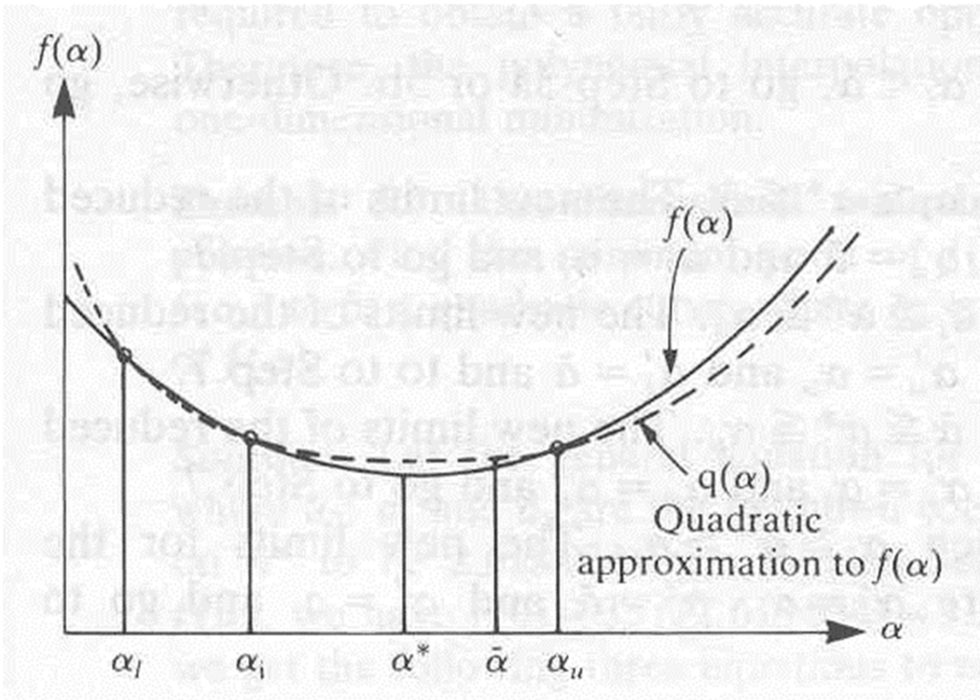


$$\begin{cases} \Delta x_a = x_1 - x_2 = x_l + R(x_u - x_l) - x_u + R(x_u - x_l) = (2R-1)(x_u - x_l) = 0.236(x_u - x_l) \\ \Delta x_b = x_u - x_1 = x_u - [x_l + R(x_u - x_l)] = (1-R)(x_u - x_l) = 0.382(x_u - x_l) \end{cases}$$

$$\rightarrow \varepsilon_a = (1-R)\left|\frac{x_u - x_l}{x_{opt}}\right| \times 100\% = 0.382\left|\frac{x_u - x_l}{x_{opt}}\right| \times 100\%$$

Polynomial Interpolation (1)

- Many function evaluations ☹
  - Approximate function → explicit minimum point
- Quadratic curve fitting: $q(\alpha) = a_0 + a_1\alpha + a_2\alpha^2$
  - known: $f(\alpha_l)$, $f(\alpha_i)$, $f(\alpha_u)$; unknown: a_0, a_1, a_2



Polynomial Interpolation (2)

$$\left.\begin{aligned} a_0 + a_1\alpha_l + a_2\alpha_l^2 &= f(\alpha_l) \\ a_0 + a_1\alpha_i + a_2\alpha_i^2 &= f(\alpha_i) \\ a_0 + a_1\alpha_u + a_2\alpha_u^2 &= f(\alpha_u) \end{aligned}\right\} \rightarrow \begin{cases} a_2 = \dfrac{1}{\alpha_u - \alpha_i}\left[\dfrac{f(\alpha_u) - f(\alpha_l)}{\alpha_u - \alpha_l} - \dfrac{f(\alpha_i) - f(\alpha_l)}{\alpha_i - \alpha_l}\right] \\ a_1 = \dfrac{f(\alpha_i) - f(\alpha_l)}{\alpha_i - \alpha_l} - a_2(\alpha_l + \alpha_i) \\ a_0 = f(\alpha_l) - a_1\alpha_l - a_2\alpha_l^2 \end{cases}$$

$$\frac{dq(\bar{\alpha})}{d\alpha} = 0 \rightarrow \bar{\alpha} = -\frac{a_1}{2a_2}; \quad \frac{d^2q(\bar{\alpha})}{d\alpha^2} > 0 \rightarrow 2a_2 > 0$$

$$(\alpha_l, \alpha_u) \text{ from bracketing}$$

$$\begin{cases} \alpha_i < \bar{\alpha} \rightarrow \begin{cases} f(\alpha_i) < f(\bar{\alpha}) : \alpha_l, \alpha_i, \bar{\alpha} \\ f(\alpha_i) > f(\bar{\alpha}) : \alpha_i, \bar{\alpha}, \alpha_u \end{cases} \\ \alpha_i > \bar{\alpha} \rightarrow \begin{cases} f(\alpha_i) < f(\bar{\alpha}) : \bar{\alpha}, \alpha_i, \alpha_u \\ f(\alpha_i) > f(\bar{\alpha}) : \alpha_l, \bar{\alpha}, \alpha_i \end{cases} \end{cases}$$

Example 10.3

$$f(\alpha) = 2 - 4\alpha + e^{\alpha}$$

$$\delta = 0.5 \left[f(\delta) < f(0) \right]$$

$$\varepsilon = 0.001 \left[I \left(= \alpha_u - \alpha_l \right) < \varepsilon \right]$$

$$\alpha^* = 1.386511, f\left(\alpha^*\right) = 0.454823$$

- # of function evaluation
  - Equal interval search : 37
  - Golden section search : 22
  - Polynomial interpolation : 5