TABLE 10.1 Golden Section Search for $f(\alpha) = 2 - 4\alpha + e^{\alpha}$ of Example 10.3

No., q	Trial step, α	Function value, $f(\alpha)$
Phase I: Initial Bracketing of Minimum.		
1. $\alpha = 0$	0.000000	3.000000
2. $q = 0$	$\alpha_0 = \delta = 0.500000 \leftarrow \alpha_l$	1.648721
3. $q = 1$	$\alpha_1 = \sum_{j=0}^{1} \delta(1.618)^j = 1.309017$	0.466464
4. $q = 2$	$\alpha_2 = \sum_{j=0}^{2} \delta(1.618)^j = 2.618034 \leftarrow \alpha_u$	5.236610

Iteration No.	$\alpha_l; [f(\alpha_l)]$	$\alpha_a; [f(\alpha_a)]$	$\alpha_b; [f(\alpha_b)]$	$\alpha_u; [f(\alpha_u)]$	$I = \alpha_u - \alpha_l$
Phase II: Reducing Interval of Uncertainty					
1	0.500000 [1.648721] ↓	1.309017 [0.466 464] ↘	**1.809017** **[0.868376]** ↘	2.618034 [5.236610]	2.118034
2	0.500000 [1.648721]	**1.000000** ↙ **[0.718282]**	1.309017 ↙ [0.466464]	1.809017 [0.868376] ↓	1.309017
3	1.000000 [0.718282]	1.309017 [0.466 464]	**1.500000** **[0.481689]**	1.809017 [0.868376]	0.809017
—	—	—	—	—	—
—	—	—	—	—	—
16	1.385438 [0.454824]	1.386031 [0.454823]	**1.386398** **[0.454823]**	1.386991 [0.454824]	0.001553
17	1.386031 [0.454823]	1.386398 [0.454823]	**1.386624** **[0.454823]**	1.386991 [0.454823]	0.000960

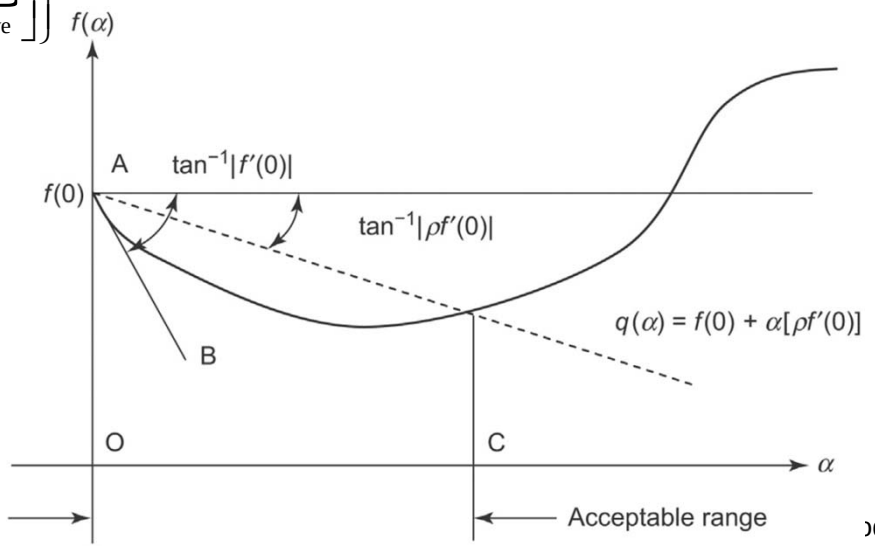
$\alpha^* = 0.5(1.386398 + 1.386624) = 1.386511; f(\alpha) = 0.454823.$

Note: New calculation for each iteration is shown as boldfaced and shaded; the arrows indicate direction of data transfer to the subsequent row/iteration.

Inexact Line Search: Armijo's Rule

- Line search termination criterion: $\nabla f\left(\mathbf{x}^{(k+1)}\right)\cdot\mathbf{d}^{(k)}=0$
- Basic concept
  - step size should not be too large or too small
  - sufficient decrease in the cost function value along the search direction

$$\left.\begin{aligned} f(\alpha) &= f\left(\mathbf{x}^{(k)}+\alpha\mathbf{d}^{(k)}\right)\\ q(\alpha) &= f(0)+\alpha\left[\rho\underbrace{f'(0)}_{\text{negative}}\right]\end{aligned}\right\}\xrightarrow[\eta>1(\eta=2)]{0<\rho<1(\rho=0.2)}\begin{cases}\text{not too large: } f(\alpha)\le q(\alpha)\left[\text{sufficient-decrease condition}\right]\\ \text{not too small: } f(\eta\alpha)>q(\eta\alpha)\end{cases}$$



Inexact Line Search: Wolfe Conditions

- Sufficient-decrease condition + curvature condition
  - Small values for α

$$f(\alpha) = f\left(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}\right)$$

$$q(\alpha) = f(0) + \alpha\left[\rho \underbrace{f'(0)}_{\text{negative}}\right]$$

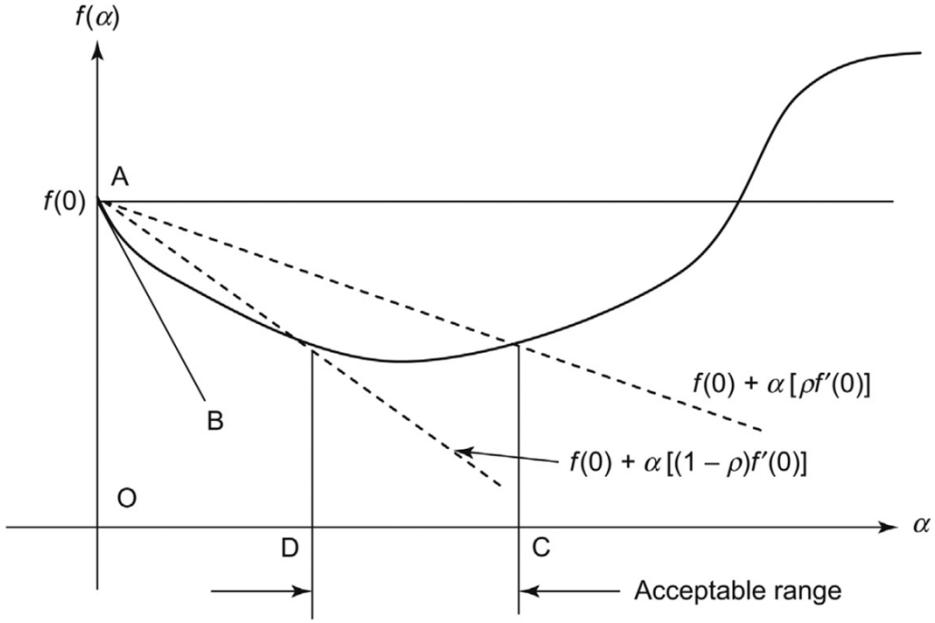
$$\rightarrow \begin{cases} \text{sufficient-decrease condition: } f(\alpha) \le q(\alpha) \\ \text{curvature condition:} \\ \xrightarrow[0<\rho<0.5]{\rho<\beta<1} f'(\alpha) \ge \beta f'(0) \\ \xrightarrow[\rho=10^{-4}\sim 10^{-3}]{\beta=0.9} \left|f'(\alpha)\right| \le \beta \left|f'(0)\right| \rightarrow \dfrac{\left|f(\alpha) - f(v)\right|}{\alpha - v} \le \beta \left|f'(0)\right|,\ 0 \le v \le \alpha \\ \text{smaller } \beta, \text{ more accurate } \alpha \rightarrow \begin{cases} \beta = 0.9: \text{ Newton and quasi-Newton method} \\ \beta = 0.1: \text{ conjugate gradient method} \end{cases} \end{cases}$$

Inexact Line Search: Goldstein Test

- somewhat similar to Armijo's rule
- Newton-type methods, but not quasi-Newton methods

$$\left.\begin{aligned} f(\alpha) &= f\left(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}\right) \\ q(\alpha) &= f(0) + \alpha\left[\rho \underbrace{f'(0)}_{\text{negative}}\right] \end{aligned}\right\} \xrightarrow{0<\rho<0.5} \begin{cases} \text{not too large: } f(\alpha) \le q(\alpha)\left[= f(0) + \alpha\rho f'(0)\right] \\ \text{not too small: } f(\alpha) \ge f(0) + \alpha\left[(1-\rho) f'(0)\right] \end{cases}$$

$$\rightarrow \rho \le \frac{f(\alpha) - f(0)}{\alpha f'(0)} \le (1-\rho)$$

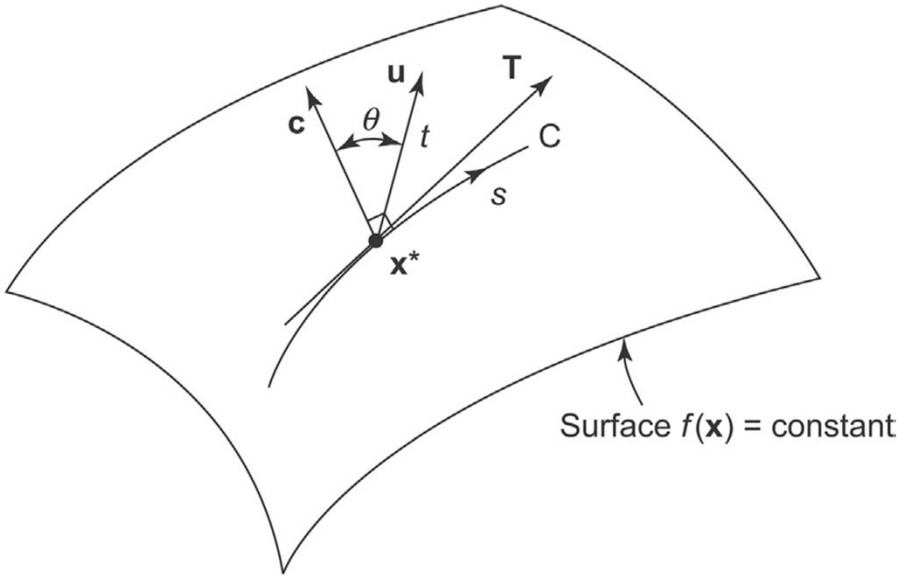


Multidimensional Unconstrained Optimization

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

Properties of Gradient Vector

- The gradient vector of a function *f* @ the point x^* is orthogonal (normal) to the tangent plane for the surface *f*=const.
- Gradient represents a direction of maximum rate of increase for the function *f* at the point x^* .
- The maximum rate of change of *f(x)* at any point x^* is the magnitude of the gradient vector



$$\mathbf{T} = \left[\frac{\partial x_1}{\partial s} \quad \frac{\partial x_2}{\partial s} \quad \cdots \quad \frac{\partial x_n}{\partial s} \right]$$

$$\frac{df}{ds} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial s} + \cdots + \frac{\partial f}{\partial x_n}\frac{\partial x_n}{\partial s} = 0$$

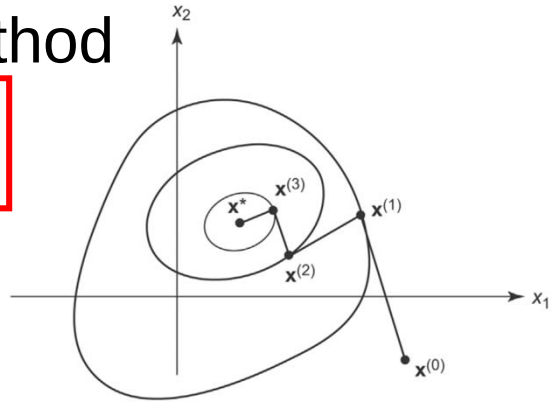
$$\frac{df}{dt} = \lim_{\varepsilon \to 0} \frac{f\left(\mathbf{x} + \varepsilon \mathbf{u}\right) - f\left(\mathbf{x}\right)}{\varepsilon}$$

$$= \sum_{i=1}^{n} u_i \frac{\partial f}{\partial x_i} = \left(\mathbf{c} \cdot \mathbf{u}\right) = \mathbf{c}^T \mathbf{u} = \left\|\mathbf{c}\right\| \left\|\mathbf{u}\right\| \cos\theta$$

$$\max \left| \frac{df}{dt} \right| = \left\|\mathbf{c}\right\|$$

Steepest Descent Method (1)

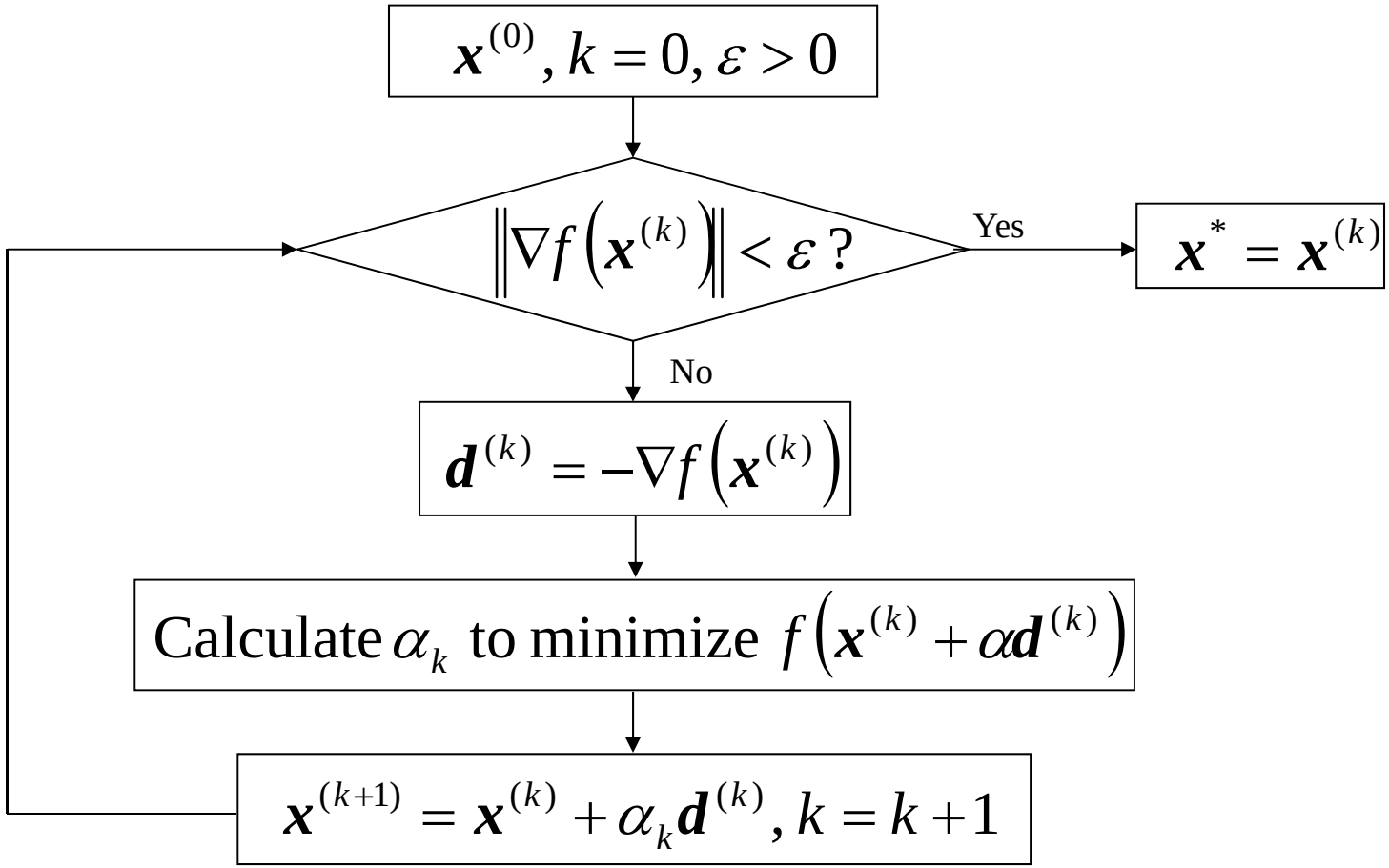
- Cauchy's Method (1847), first-order method
- Steepest descent direction : $\boldsymbol{d} = -\nabla f = -\dfrac{\partial f}{\partial x}$
  - satisfy $\nabla f\left(\boldsymbol{x}^{(k)}\right)\cdot \boldsymbol{d}^{(k)} < 0 \rightarrow -\left\|\nabla f\left(\boldsymbol{x}^{(k)}\right)\right\|^2 < 0$



- One-dimensional search : $\dfrac{df\left(\boldsymbol{x}^{(k+1)}\right)}{d\alpha} = \nabla f\left(\boldsymbol{x}^{(k+1)}\right)\cdot \nabla f\left(\boldsymbol{x}^{(k)}\right) = 0$

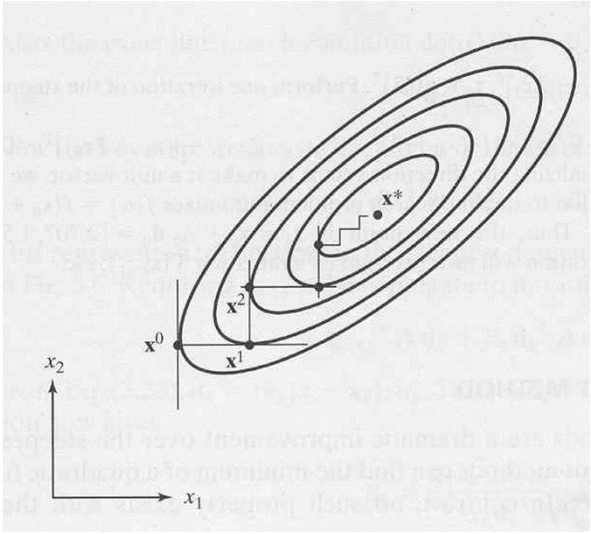
- ☺ simple & robust (convergence guaranteed)
- ☹ slow rate of convergence
  - Orthogonal path, condition # of the Hessian
- ☹ inefficient: no previous information is used

Steepest Descent Algorithm



Steepest Descent Method (2)

- Scaling of design variables
  - Accelerate the rate of convergence
  - For a p.d. quadratic function with a unit condition number
    - Converge in only one iteration
  - Unscale the transformed design variables



$$f = x_1^{\,2} + a x_2^{\,2} \rightarrow \nabla^2 f = \begin{bmatrix} 2 & 0 \\ 0 & 2a \end{bmatrix}$$

$$\rightarrow cond\left(\nabla^2 f\right) = a$$

$$f = f(\boldsymbol{x}) \xrightarrow{\boldsymbol{x}=\boldsymbol{T}\boldsymbol{y}} g(\boldsymbol{y}) = f(\boldsymbol{T}\boldsymbol{y})$$

$$\rightarrow \nabla^2 g = \boldsymbol{T}^T \nabla^2 f \boldsymbol{T}$$

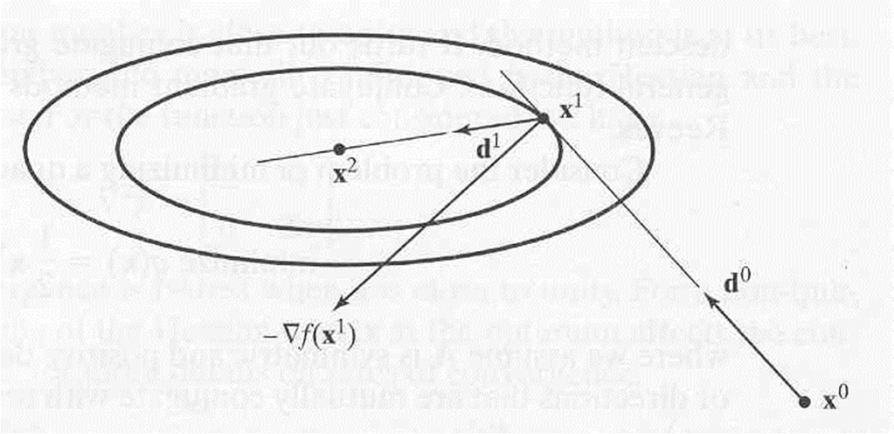
$$\rightarrow cond\left(\nabla^2 g\right) \approx 1$$

Conjugate Gradient Method (1)

- Conjugate Gradient Direction: Modification of SDM
  - Not orthogonal to each other
  - Cut diagonally through the orthogonal steepest descent directions
  - Orthogonal w.r.t. a symmetric and p.d. A :

$$\boldsymbol{d}^{(i)T}\boldsymbol{A}\boldsymbol{d}^{(j)} = \begin{cases} a_{jj} & (i=j) \\ 0 & (i\neq j) \end{cases}$$

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



  - Satisfies descent condition

$$\alpha_k\left(\nabla f\left(\boldsymbol{x}^{(k)}\right)\cdot\boldsymbol{d}^{(k)}\right)<0 \rightarrow -\alpha_k\nabla f\left(\boldsymbol{x}^{(k)}\right)\cdot\nabla f\left(\boldsymbol{x}^{(k)}\right)+\alpha_k\beta_k\underbrace{\nabla f\left(\boldsymbol{x}^{(k)}\right)\cdot\boldsymbol{d}^{(k-1)}}_{=0\text{ (step size condition)}}<0$$

Conjugate Gradient Method (2)

- Assumption: exact line search
  - Quadratic function: Fletcher & Reeves(1964)
  - More general function: Polak-Rebiere (1969)
- ☺ Simple & Fast Convergent Rate
- *n* iterations for *n* design variables (p.d. quadratic form)

$$\boldsymbol{x}^{(0)} \xrightarrow{\boldsymbol{d}^{(0)}} \boldsymbol{x}^{(1)} \xrightarrow{\boldsymbol{d}^{(1)}} \boldsymbol{x}^{(2)} \cdots \xrightarrow{\boldsymbol{d}^{(n-1)}} \boldsymbol{x}^{(n)}$$

- Restart every (*n*+1) iterations for computational stability

Conjugate Gradient Method (3)

- Consider the problem of minimizing a quadratic function

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} \rightarrow \nabla f\left(\boldsymbol{x}^{(k)}\right) = \boldsymbol{A}\boldsymbol{x}^{(k)} + \boldsymbol{c}$$

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

$$\frac{\partial f(\alpha)}{\partial \alpha} = \boldsymbol{d}^{(k)^T} \nabla f\left(\boldsymbol{x}^{(k+1)}\right) = 0$$

$$\boldsymbol{d}^{(k)^T}\left[\boldsymbol{A}\left(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}\right) + \boldsymbol{c}\right] = 0 \rightarrow \alpha_k = -\frac{\boldsymbol{d}^{(k)^T}\nabla f\left(\boldsymbol{x}^{(k)}\right)}{\boldsymbol{d}^{(k)^T}\boldsymbol{A}\boldsymbol{d}^{(k)}}$$

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

$$\boldsymbol{d}^{(k+1)^T}\boldsymbol{A}\boldsymbol{d}^{(k)} = -\nabla f\left(\boldsymbol{x}^{(k+1)}\right)^T \boldsymbol{A}\boldsymbol{d}^{(k)} + \beta_k \boldsymbol{d}^{(k)^T}\boldsymbol{A}\boldsymbol{d}^{(k)} = 0 \rightarrow \beta_k = \frac{\nabla f\left(\boldsymbol{x}^{(k+1)}\right)^T \boldsymbol{A}\boldsymbol{d}^{(k)}}{\boldsymbol{d}^{(k)^T}\boldsymbol{A}\boldsymbol{d}^{(k)}}$$

Polak-Rebiere (1969)

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)} \rightarrow \mathbf{d}^{(k)} = \frac{\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}}{\alpha_k} \rightarrow \mathbf{A}\mathbf{d}^{(k)} = \frac{\mathbf{A}\mathbf{x}^{(k+1)} - \mathbf{A}\mathbf{x}^{(k)}}{\alpha_k} = \frac{\nabla f\left(\mathbf{x}^{(k+1)}\right) - \nabla f\left(\mathbf{x}^{(k)}\right)}{\alpha_k}$$

$$\left.\begin{aligned} &\mathbf{d}^{(k)T} \nabla f\left(\mathbf{x}^{(k+1)}\right) = 0 \rightarrow \mathbf{d}^{(k-1)T} \nabla f\left(\mathbf{x}^{(k)}\right) = 0 \\ &\mathbf{d}^{(k)} = -\nabla f\left(\mathbf{x}^{(k)}\right) + \beta_{k-1} \mathbf{d}^{(k-1)} \rightarrow \mathbf{d}^{(k-1)} = \frac{\mathbf{d}^{(k)} + \nabla f\left(\mathbf{x}^{(k)}\right)}{\beta_{k-1}} \end{aligned}\right\} \rightarrow \frac{\left[\mathbf{d}^{(k)} + \nabla f\left(\mathbf{x}^{(k)}\right)\right]^T \nabla f\left(\mathbf{x}^{(k)}\right)}{\beta_{k-1}} = 0$$

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

$$\alpha_k = -\frac{\mathbf{d}^{(k)T} \nabla f\left(\mathbf{x}^{(k)}\right)}{\mathbf{d}^{(k)T} \mathbf{A}\mathbf{d}^{(k)}} \rightarrow \alpha_k = \frac{\nabla f\left(\mathbf{x}^{(k)}\right)^T \nabla f\left(\mathbf{x}^{(k)}\right)}{\mathbf{d}^{(k)T} \mathbf{A}\mathbf{d}^{(k)}}$$

$$\beta_k = \frac{\nabla f\left(\mathbf{x}^{(k+1)}\right)^T \mathbf{A}\mathbf{d}^{(k)}}{\mathbf{d}^{(k)T} \mathbf{A}\mathbf{d}^{(k)}} \rightarrow \beta_k = \frac{\nabla f\left(\mathbf{x}^{(k+1)}\right)^T \left[\nabla f\left(\mathbf{x}^{(k+1)}\right) - \nabla f\left(\mathbf{x}^{(k)}\right)\right]}{\alpha_k \mathbf{d}^{(k)T} \mathbf{A}\mathbf{d}^{(k)}}$$

$$\rightarrow \boxed{\beta_k = \frac{\nabla f\left(\mathbf{x}^{(k+1)}\right)^T \left[\nabla f\left(\mathbf{x}^{(k+1)}\right) - \nabla f\left(\mathbf{x}^{(k)}\right)\right]}{\nabla f\left(\mathbf{x}^{(k)}\right)^T \nabla f\left(\mathbf{x}^{(k)}\right)}}$$

for more general objective model

Fletcher-Reeves (1964)

for a quadratic function $\left[\nabla f\left(\mathbf{x}^{(k+1)}\right)-\nabla f\left(\mathbf{x}^{(k)}\right)=\mathbf{A}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{(k)}\right)=\alpha_k\mathbf{A}\mathbf{d}^{(k)}\right]$,

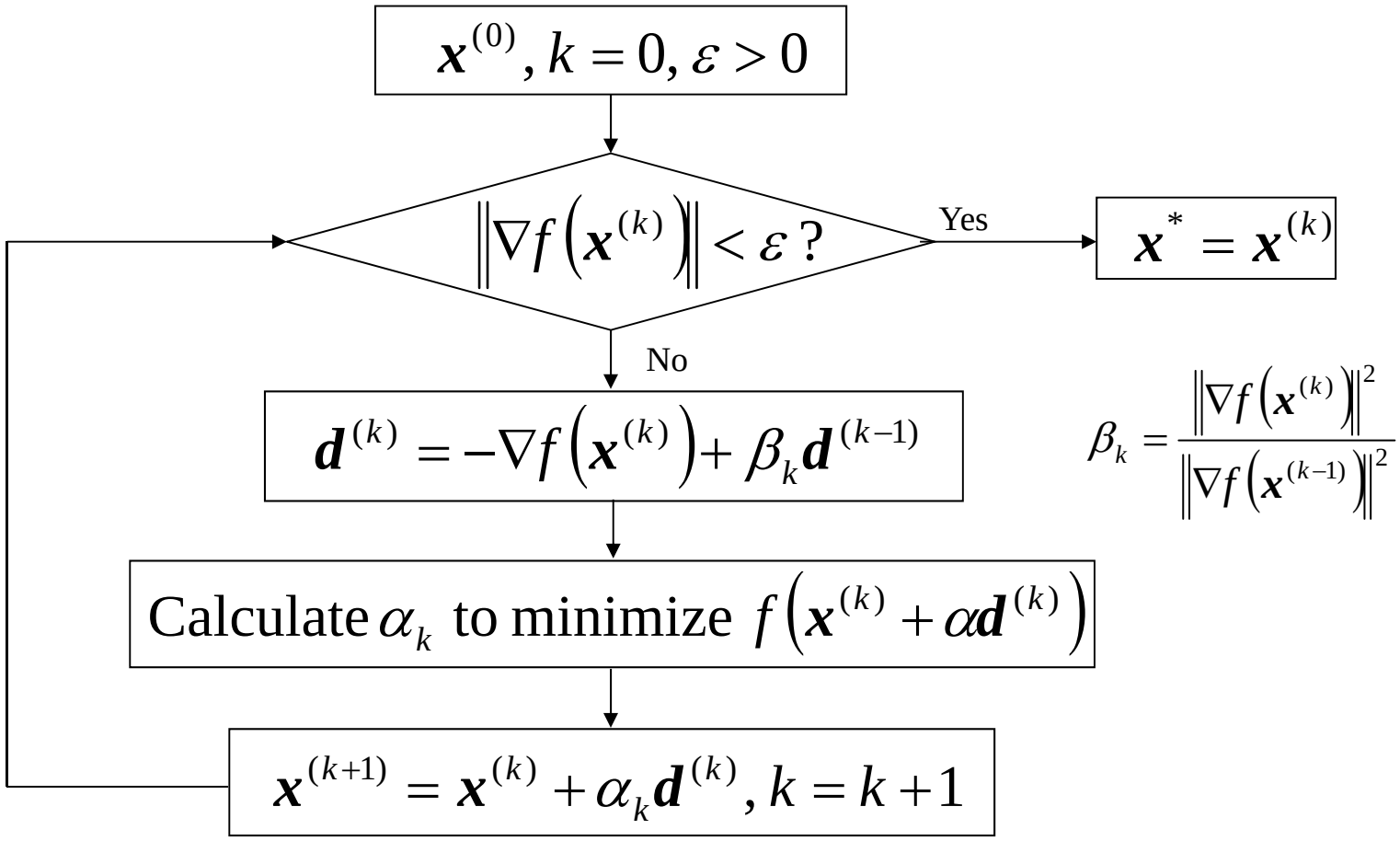
$$\nabla f\left(\mathbf{x}^{(k+1)}\right)^T\nabla f\left(\mathbf{x}^{(k)}\right)=\nabla f\left(\mathbf{x}^{(k+1)}\right)^T\left[-\mathbf{d}^{(k)}+\beta_{k-1}\mathbf{d}^{(k-1)}\right]=-\underbrace{\nabla f\left(\mathbf{x}^{(k+1)}\right)^T\mathbf{d}^{(k)}}_{=0\ \text{(exact line search)}}+\nabla f\left(\mathbf{x}^{(k+1)}\right)^T\beta_{k-1}\mathbf{d}^{(k-1)}$$

$$=\beta_{k-1}\left[\nabla f\left(\mathbf{x}^{(k)}\right)^T+\alpha_k\mathbf{d}^{(k)T}\mathbf{A}\right]\mathbf{d}^{(k-1)}=\beta_{k-1}\left[\underbrace{\nabla f\left(\mathbf{x}^{(k)}\right)^T\mathbf{d}^{(k-1)}}_{=0\ \text{(exact line search)}}+\alpha_k\underbrace{\mathbf{d}^{(k)T}\mathbf{A}\mathbf{d}^{(k-1)}}_{=0\ \text{(conjugacy)}}\right]=0$$

$$\beta_k=\frac{\nabla f\left(\mathbf{x}^{(k+1)}\right)^T\left[\nabla f\left(\mathbf{x}^{(k+1)}\right)-\nabla f\left(\mathbf{x}^{(k)}\right)\right]}{\nabla f\left(\mathbf{x}^{(k)}\right)^T\nabla f\left(\mathbf{x}^{(k)}\right)}\rightarrow\beta_k=\frac{\nabla f\left(\mathbf{x}^{(k+1)}\right)^T\nabla f\left(\mathbf{x}^{(k+1)}\right)}{\nabla f\left(\mathbf{x}^{(k)}\right)^T\nabla f\left(\mathbf{x}^{(k)}\right)}$$

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

Conjugate Gradient Algorithm



Example 10.5+6

$$f\left(x_1, x_2, x_3\right) = x_1^{\,2} + 2x_2^{\,2} + 2x_3^{\,2} + 2x_1x_2 + 2x_2x_3$$

$$\mathbf{x}^{(0)} = \left(2, 4, 10\right),\ \varepsilon = 0.005$$

line search by golden section: $\delta = 0.05,\ \varepsilon = 0.0001$

TABLE 10.2 Optimum Solution for Example 10.5 with the Steepest–Descent Method

$f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3$	
Starting values of design variables	2, 4, 10
Optimum design variables	8.04787E-03, −6.81319E-03, 3.42174E-03
Optimum cost function value	2.47347E-05
Norm of gradient of the cost function at optimum	4.97071E-03
Number of iterations	40
Total number of function evaluations	753

TABLE 10.3 Optimum Solution for Example 10.6 with the Conjugate Gradient Method

$f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3$	
Starting values of design variables	2, 4, 10
Optimum design variables	1.01E-07, −1.70E-07, 1.04E-09
Optimum cost function value	−4.0E-14
Norm of gradient at optimum	5.20E-07
Number of iterations	4

Newton's Method

- Optimality criteria for second-order Taylor series

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

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

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

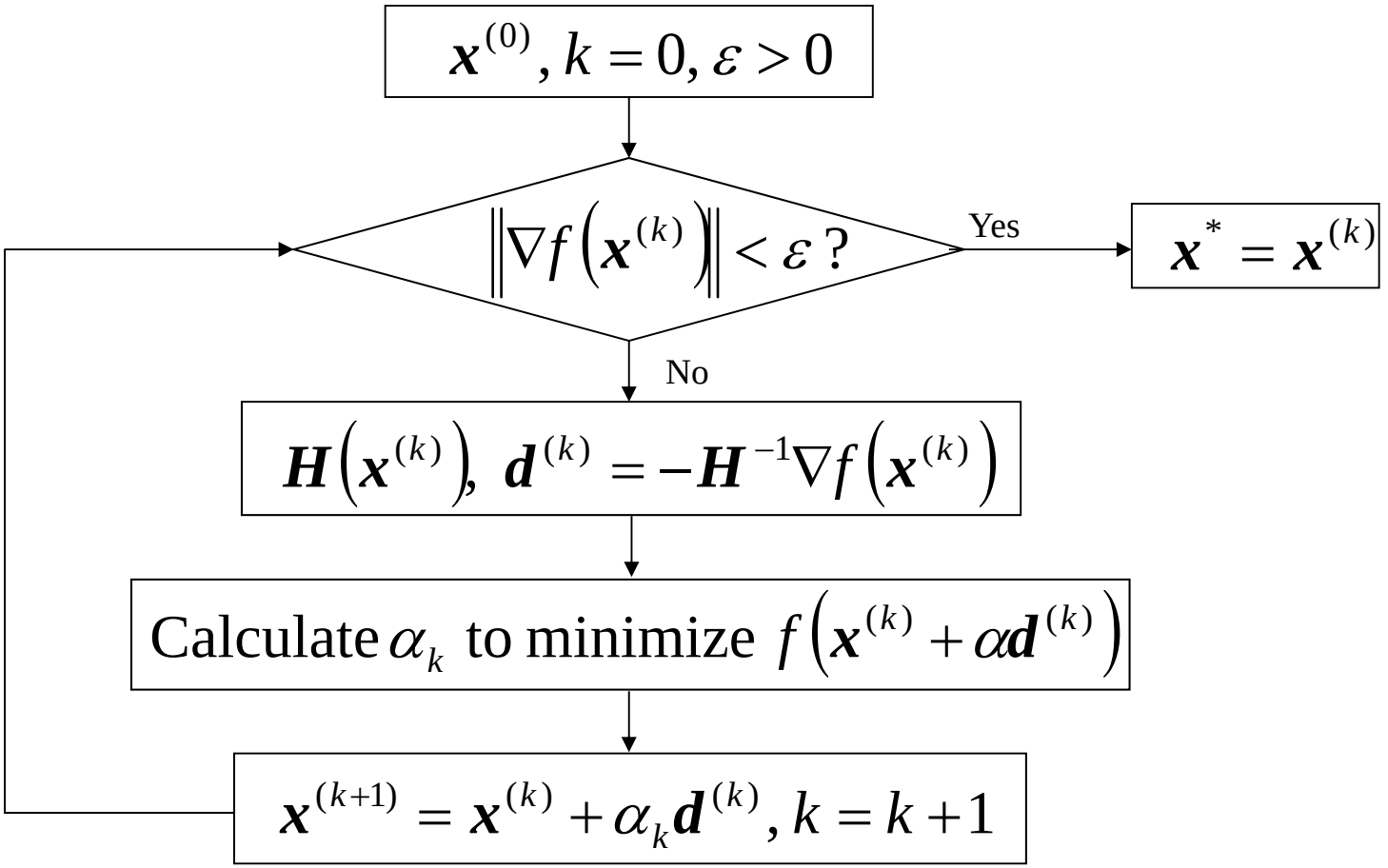
- Descent direction?

$$\alpha_k\left(\nabla f\left(\boldsymbol{x}^{(k)}\right)\cdot\boldsymbol{d}^{(k)}\right) < 0 \rightarrow -\alpha_k\nabla f\left(\boldsymbol{x}^{(k)}\right)\cdot\boldsymbol{H}^{-1}\nabla f\left(\boldsymbol{x}^{(k)}\right) < 0$$

- Quadratic rate of convergence ☺
- Second-order derivative : high cost ☹
- No guaranteed convergence unless ***H*** is p.d. ☹
- Memoryless method (no use of previous data) ☹

Modified Newton's Algorithm

- Introduce the step length parameter



Example 11.8

$$f(\boldsymbol{x}) = 50\left(x_2 - x_1^2\right)^2 + \left(2 - x_1\right)^2 \quad \text{from } \boldsymbol{x}_0 = (5, -5)$$

$$\delta_0 = 0.05$$

$$\varepsilon = \begin{cases} 0.00001 \ (\text{steepest descent method}) \\ 0.0001 \ (\text{modified Newton method}) \end{cases}$$

TABLE 11.3 Comparative Evaluation of Three Methods for Example 11.8

$f(x) = 50\left(x_2 - x_1^2\right)^2 + (2 - x_1)^2$

	Steepest-descent	Conjugate gradient	Modified Newton
x_1	1.9941E+00	2.0000E+00	2.0000E+00
x_2	3.9765E+00	3.9998E+00	3.9999E+00
f	3.4564E−05	1.0239E−08	2.5054E−10
$\|\mathbf{c}\|$	3.3236E−03	1.2860E−04	9.0357E−04
Number of function evaluations	138236	65	349
Number of iterations	9670	22	13

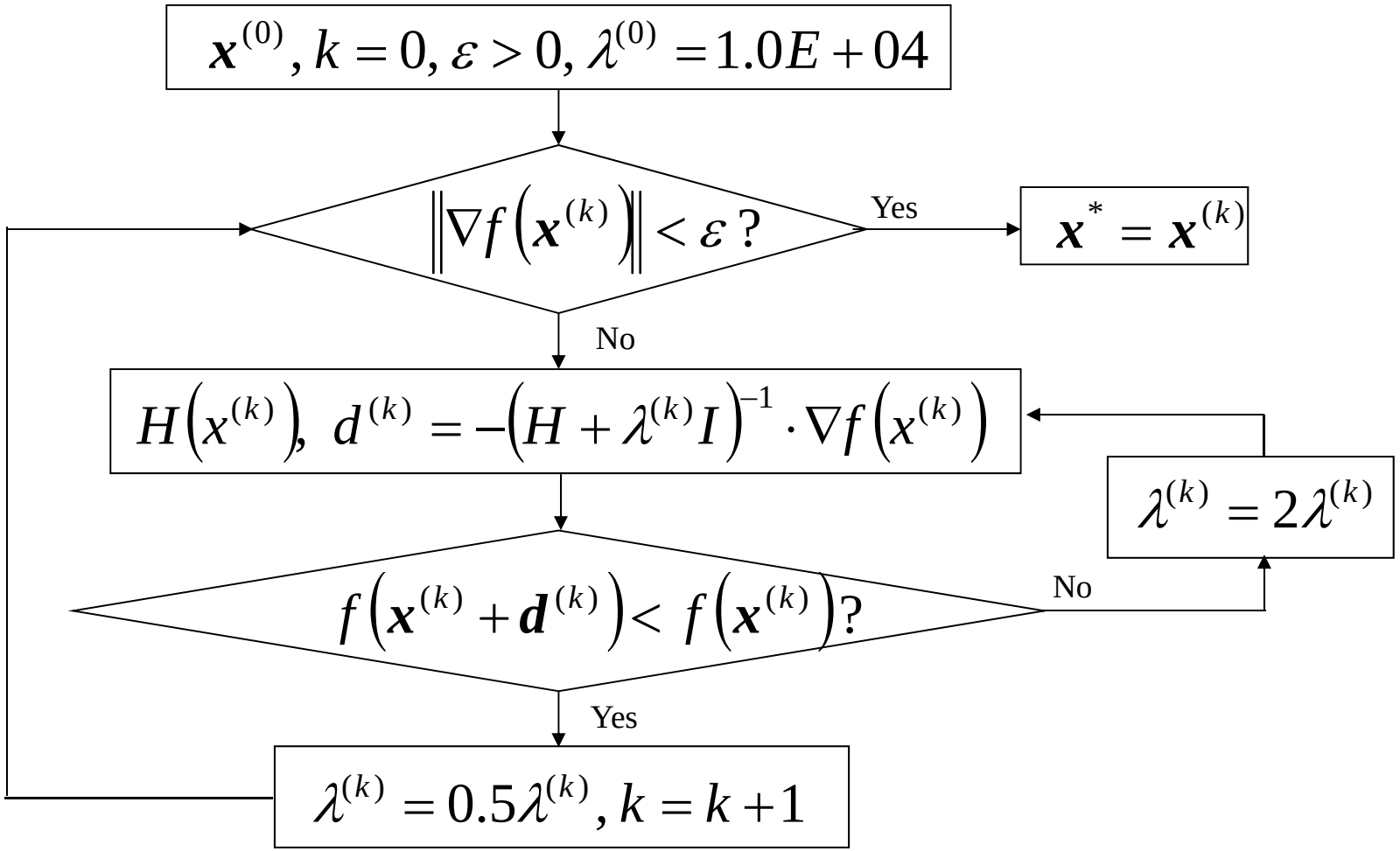
Marquardt Modification

- Marquardt (1963)
  - Steepest descent method: far away from the solution
  - Newton method: near the solution point

$$\boldsymbol{d}^{(k)} = -\left(\boldsymbol{H} + \lambda \boldsymbol{I}\right)^{-1} \nabla f\left(\boldsymbol{x}\right)$$

  - When λ is large,
    - steepest descent direction: $\boldsymbol{d}^{(k)} = \left(-1/\lambda\right)\nabla f\left(\boldsymbol{x}\right)$
  - As λ is reduced (step size is increased),
    - Newton direction: $\boldsymbol{d}^{(k)} = -\boldsymbol{H}^{-1}\nabla f\left(\boldsymbol{x}\right)$
  - If the direction does not reduce the cost function, then λ is increased (step size is reduced)

Marquardt's Algorithm (1)



Marquardt's Algorithm (2)

- Advantages
  - Simplicity
  - Descent property
  - Excellent convergence rate near $\boldsymbol{x}^*$
  - Absence of a line search
- Disadvantages
  - Need to calculate $\boldsymbol{H}^{(k)}$
  - Solve the linear equation set

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

Quasi-Newton Method

- No learning process
  - Steepest Descent Method: Convergent, but Slow
  - Newton's Method: Fast, but Expensive
- Use of previous information, speed up the convergence !
  - Approximate Hessian (or its inverse) matrix by 1st-order derivatives preserving symmetry and positive definiteness
- Variable Metric Method

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

$$\boldsymbol{A}^{(k+1)} = \boldsymbol{A}^{(k)} + \boldsymbol{A}_c^{(k)} \xrightarrow{\text{as } k \to \infty} \boldsymbol{H}^{-1}$$

DFP Method (1)

- Davidon (1959) $\rightarrow$ Fletcher and Powell (1963)
  - Approximate inverse of Hessian matrix
    - For a quadratic objective, directions of the conjugate gradient method
  - One of the most powerful algorithm for general function
  - ☹ need to store the $N \times N$ matrix A

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

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

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

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

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

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

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

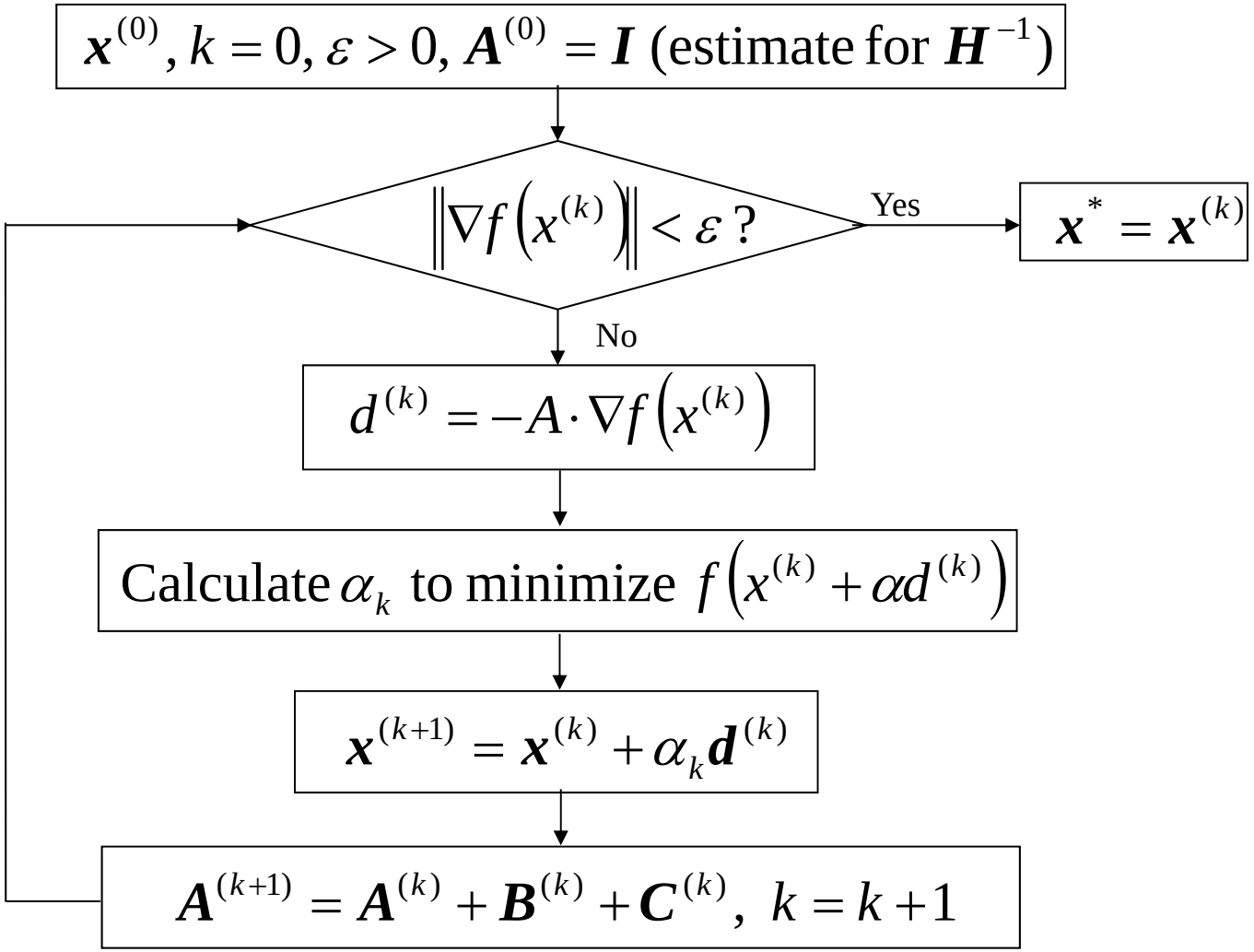
$$\rightarrow \Delta x^{(k)} = -\alpha_k A^{(k)} \nabla f\left(x^{(k)}\right)$$

$$c^{(k)} = \nabla f\left(x^{(k)}\right)$$

$$f\left(x^{(k+1)}\right) = f\left(x^{(k)}\right) + c^{(k)^T} \Delta x^{(k)}$$

$$\Delta f\left(x^{(k)}\right) = -\alpha^{(k)} c^{(k)^T} A^{(k)} c^{(k)} < 0$$

DFP Method (2)


$\boldsymbol{x}^{(0)}, k = 0, \varepsilon > 0, \boldsymbol{A}^{(0)} = \boldsymbol{I}$ (estimate for $\boldsymbol{H}^{-1}$)
$\left\|\nabla f\left(x^{(k)}\right)\right\| < \varepsilon$?
Yes
$\boldsymbol{x}^{*} = \boldsymbol{x}^{(k)}$
No
$d^{(k)} = -A \cdot \nabla f\left(x^{(k)}\right)$
Calculate α_k to minimize $f\left(x^{(k)} + \alpha d^{(k)}\right)$
$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}$
$\boldsymbol{A}^{(k+1)} = \boldsymbol{A}^{(k)} + \boldsymbol{B}^{(k)} + \boldsymbol{C}^{(k)}, \; k = k + 1$

BFGS Method (1)

- Broyden-Fletcher-Goldfarb-Shanno (1981)
  - Direct update the Hessian matrix
  - Most popular and effective

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

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

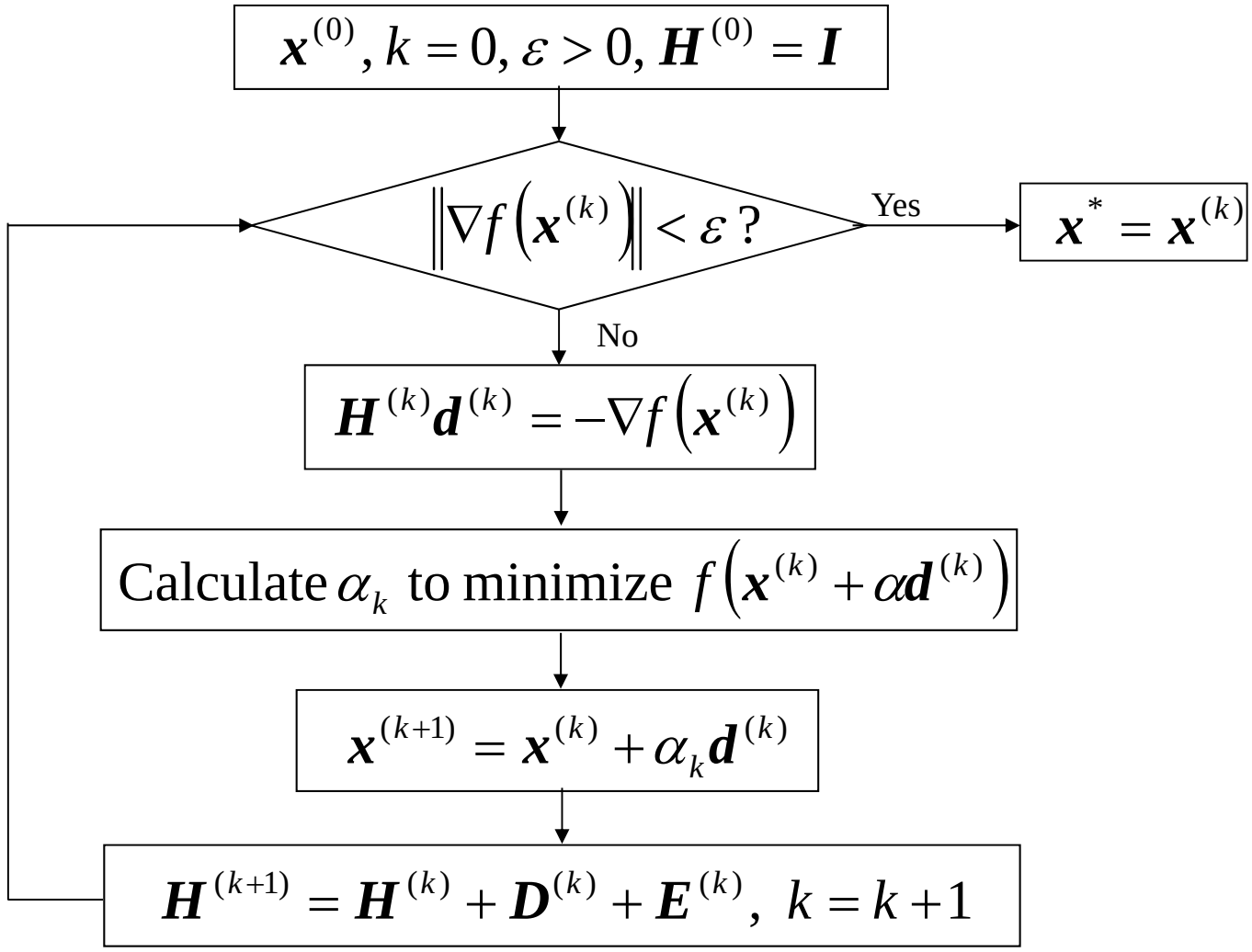
$$H^{(k+1)} = H^{(k)} + \frac{y^{(k)} y^{(k)^T}}{y^{(k)^T} s^{(k)}} - \frac{H^{(k)} s^{(k)} s^{(k)^T} H^{(k)}}{s^{(k)^T} H^{(k)} s^{(k)}}$$

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

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

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

BFGS Method (2)



Remarks

- DFP and BFGS methods have theoretical properties that guarantee
  - Superlinear (fast) convergence rate
  - Global convergence under certain conditions
- Duals of each other: $\boldsymbol{s} \leftrightarrow \boldsymbol{y}$, $\boldsymbol{H} \leftrightarrow \boldsymbol{A}$
- Both methods could fail for nonlinear problems,
  - DFP is highly sensitive to inaccuracies in line searches.
  - Both methods can get stuck on a saddle point.
  - Update of Hessian becomes “corrupted” by round-off and other inaccuracies.
- All kinds of “tricks” such as scaling and preconditioning exist to boost the performance of the methods

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 $\beta_k = \left\|\nabla f\left(\boldsymbol{x}^{(k)}\right)\right\|^2 \Big/ \left\|\nabla f\left(\boldsymbol{x}^{(k-1)}\right)\right\|^2$
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 $H^{(k+1)} = H^{(k)} + \frac{y^{(k)}y^{(k)^T}}{y^{(k)^T}s^{(k)}} + \frac{c^{(k)}c^{(k)^T}}{c^{(k)^T}d^{(k)}}$

Rate/Order of Convergence

- Algorithm has *p* order of convergence if

$$0 \le \beta = \lim_{k \to \infty} \frac{\left\| x^{(k+1)} - x^* \right\|}{\left\| x^{(k)} - x^* \right\|^p} < \infty$$

$$\beta : \text{ convergence ratio } \left(\text{asymptotic error constant}\right)$$

  - Linear convergence rate: $p = 1$, $\beta \le 1$
    - Steepest Descent Method
  - Superlinear convergence: $p = 1$, $\beta = 0$
    - Quasi-Newton Method
  - Quadratic convergence: $p = 2$
    - Newton's Method