Parameter Optimization: Unconstrained

We will begin our study by developing some tools and concepts associated with the *general* optimization process applied to problems that are *independent of time*

$\Rightarrow$ These are known as *parameter optimization* problems

- We shall utilize a useful class of algorithms known as *iterative* methods
  - Iterative methods generate a sequence of points, $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \ldots$, or more compactly $\left\{\boldsymbol{x}^{(k)}\right\}$, that converge to a fixed point $\boldsymbol{x}^*$ which is the solution to a given problem
- For example, let us define a *line* as a set of points $\boldsymbol{x}(\alpha) = \boldsymbol{x}' + \alpha \boldsymbol{s}$ where $\boldsymbol{x}'$ is a fixed point and $\boldsymbol{s}$ is the *direction* of the line (see a 2-D representation in Figure 3.1)
- An iterative scheme might systematically choose new directions $\boldsymbol{s}$ at each step and then minimize function values along those directions to generate a sequence of solution points $\left\{\boldsymbol{x}^{(k)}\right\}$

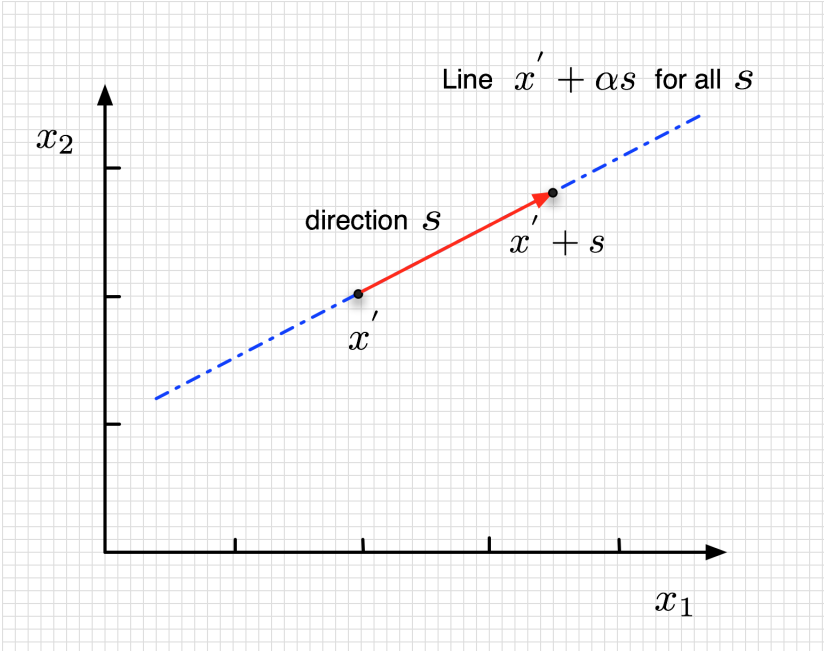


Figure 3.1 A line in two dimensions

- An acceptable iterative optimization algorithm exhibits the following properties:
  - iterations $\boldsymbol{x}^{(k)}$ move steadily toward the neighborhood of a local minimizer $\boldsymbol{x}^*$
  - iterations converge rapidly to the point $\boldsymbol{x}^*$, i.e., for $\boldsymbol{h}^{(k)} = \boldsymbol{x}^{(k)} - \boldsymbol{x}^*$, $\boldsymbol{h}^{(k)} \rightarrow \mathbf{0}$ for some appropriate measure of $\boldsymbol{h}^{(k)}$
  - rate of convergence is an important measure of goodness of the algorithm
- A method is usually based on a *model* – an approximation of the objective function – which enables an estimate of the local minimizer to be made
  - most successful have been quadratic models

3.1: Unconstrained Optimization: The Basics

- To begin, we must first define the goals we hope to achieve through optimization
  - We introduce an *index of performance*, or *objective function*, that captures the natue of our optimization goal – we'll call this function L
    - In general, L will be a function of one, two or many variables; i.e., $L = f\,(u_1,\, u_2, \ldots,\, u_m)$, where the u_i are scalar parameters
    - NOTE: it's also customary to use J to denote an objective function and x_k for the independent variables; e,g., $J = f\left(\; x_1,\;\; x_2,\;\; \ldots,\;\; x_m \;\right)$
  - SImply put, our main task will be to select the decision variables $\{u_1,\, u_2, \ldots,\, u_m\}$ such that L *is minimized*
    - Recall here that maximization can be achieved by simply switching the sign on a minimization problem
  - But what exacly do we mean by a *minimum*? We generally consider two definitions:

absolute (or global) minimum

$\Rightarrow \quad L(u_1^* + \triangle u_1,\, u_2^* + \triangle u_2, \ldots,\, u_m^* + \triangle u_m) > L(u_1^*,\, u_2^*, \ldots,\, u_m^*)$
for *all* changes $\triangle u_1,\, \triangle u_2,\, \ldots,\, \triangle u_m$

local minimum

$\Rightarrow \quad L(u_1^* + \triangle u_1,\, u_2^* + \triangle u_2, \ldots,\, u_m^* + \triangle u_m) > L(u_1^*,\, u_2^*, \ldots,\, u_m^*)$
for all *infinitesimal* changes $\triangle u_1,\, \triangle u_2,\, \ldots,\, \triangle u_m$, where values u^* denote the optimal (minimizing) values of u

– An optimization problem usually assumes that an optimum solution $\boldsymbol{u}^*$ exists, is unique and can be found, but this ideal situation may not hold for a number of reasons:
  - $L(\boldsymbol{u})$ is unbounded below
  - $L(\boldsymbol{u})$ is bounded below
  - $\boldsymbol{u}^*$ is not unique
  - local minimum exists that is not a global minimum
  - local minimum exists although $L(\boldsymbol{u})$ is unbounded below (see Figure 3.2)

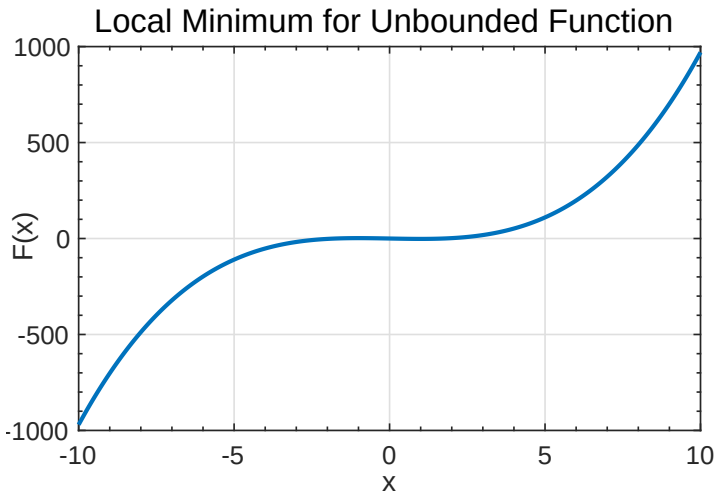


Figure 3.2 $f(x) = x^3 - 3x$

– The conditions for a local minimum are considerably easier to solve than for a global minimum; we'll address the local minimum problem in this course

– NOTE: We will focus on *minimizing* performance indices (or objective functions). The problem of maximizing an objective function fits easily within this framework by simply letting $\acute{L} = -L$

Conditions for Local Minima

- Along any line $\boldsymbol{u}(\alpha) = \boldsymbol{u}^* + \alpha \boldsymbol{s}$ through $\boldsymbol{u}^*$, $L\left[\boldsymbol{u}(\alpha)\right]$ has both zero slope and non-negative curvature at $\boldsymbol{u}^*$ (see Figure 3.3)
- This is the usual condition derived from a Taylor series for a local minimum of a function of one variable

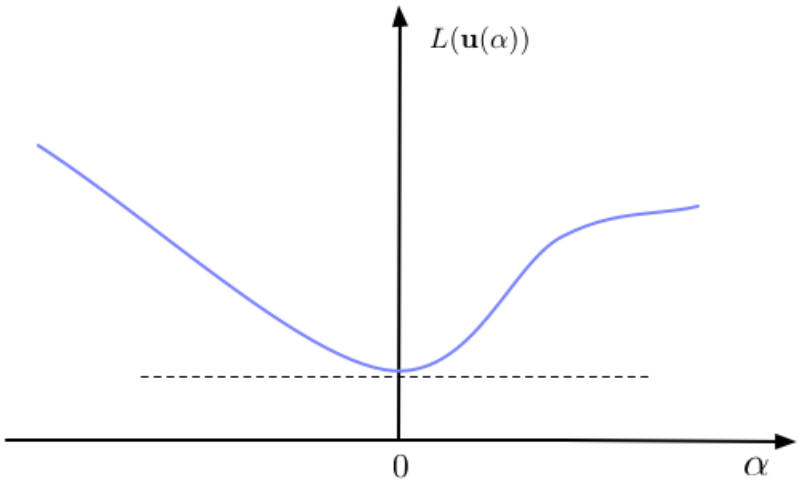


Figure 3.3 Zero slope and non-negative curvature at $\alpha = 0$

3.2: Unconstrained Optimization: One and Two Parameters

Single Parameter Problem

- Consider the function: $L(u) = (u-1)^2$

  How do we find the minimum?

$$\frac{dL}{du} = 0 = 2(u-1) \quad \Rightarrow \quad u = 1$$

$$\frac{d^2L}{du^2} = 2 > 0$$

- Why does this work?
  - if we let u^* denote a local minimum of $L(u)$, then L can be expanded in a Taylor series about u^*:

$$L(u) = L(u^*) + \left.\frac{dL}{du}\right|_{u^*} \triangle u + \frac{1}{2}\left.\frac{d^2L}{du^2}\right|_{u^*} \triangle u^2 + \cdots$$

  or

$$\triangle L = L(u) - L(u^*) = \left.\frac{dL}{du}\right|_{u^*} \triangle u + \frac{1}{2}\left.\frac{d^2L}{du^2}\right|_{u^*} \triangle u^2 + \cdots$$

  - Since u^* is a local minimum, we know two things:
    1. $L(u) - L(u^*) > 0$ for all u in a neighborhood of u^*
    2. $\triangle u$ is an arbitrary, but infinitesimal change in u away from $u^* \Rightarrow$ higher order terms in Taylor series expansion are insignificant:

$$\Rightarrow \quad \triangle L \approx \left.\frac{dL}{du}\right|_{u^*} \triangle u$$

  But since $\triangle u$ is arbitrary, $\left.\frac{dL}{du}\right|_{u^*} \neq 0 \quad \Rightarrow \quad \triangle L < 0$ for some $\triangle u$, and by deduction,

$$\Rightarrow \quad u^* \text{ can only be a minimum if } \left.\frac{dL}{du}\right|_{u^*} = 0$$

If $\left.\frac{dL}{du}\right|_{u^*} = 0$,

$$\triangle L \approx \left.\frac{d^2L}{du^2}\right|_{u^*} \triangle u^2$$

but $\triangle u^2 > 0$ for all $\triangle u$, so $\triangle L > 0$ if $\left.\frac{d^2L}{du^2}\right|_{u^*} > 0$

$$\Rightarrow \quad u^* \text{ will be a minimum if } \left.\frac{d^2L}{du^2}\right|_{u^*} > 0$$

Sufficient Conditions For a Local Minimum

$$\left.\frac{dL}{du}\right|_{u^*} = 0; \qquad \left.\frac{d^2L}{du^2}\right|_{u^*} > 0$$

- What if $\left.\frac{d^2L}{du^2}\right|_{u^*} = 0$?

- Must go to higher order derivatives (odd derivatives must be zero, 1st even derivatives must be positive)

Necessary Conditions For a Local Minimum

$$\left.\frac{dL}{du}\right|_{u^*} = 0; \qquad \left.\frac{d^2L}{du^2}\right|_{u^*} \geq 0$$

QUESTION: What is the difference between necessary and sufficient conditions?

Two-Parameter Problem

- Consider the function $L(u_1,\, u_2)$ where $L(u_1^*,\, u_2^*)$ is a local minimum
- We'll use the same Taylor series arguments as above to develop conditions for a minimum, but now the Taylor series is more complicated:

$$L(u_1,\, u_2) = L(u_1^*,\, u_2^*) + \left.\frac{\partial L}{\partial u_1}\right|_{u_1^*,u_2^*} \triangle u_1 + \left.\frac{\partial L}{\partial u_2}\right|_{u_1^*,u_2^*} \triangle u_2$$
$$+ \frac{1}{2}\left\{ \left.\frac{\partial^2 L}{\partial u_1^2}\right|_* \triangle u_1^2 + 2\left.\frac{\partial^2 L}{\partial u_1 \partial u_2}\right|_* \triangle u_1 \triangle u_2 + \left.\frac{\partial^2 L}{\partial u_2^2}\right|_* \triangle u_2^2 \right\} + \cdots$$

- Clearly, $\left(u_1^*,\, u_2^*\right)$ can only be a minimum if the following stationarity condition is attained:

$$\left.\frac{\partial L}{\partial u_1}\right|_* = \left.\frac{\partial L}{\partial u_2}\right|_* = 0$$

  – If these conditions are satisfied, then the second-order term in the Taylor series expansion must be greater than or equal to to zero for $\left(\, u_1^*,\ u_2^* \,\right)$ to be a minimizer

- Let's re-write the 2nd-order term to see how we can validate this condition:

$$\frac{1}{2}\left\{ \left.\frac{\partial^2 L}{\partial u_1^2}\right|_* \triangle u_1^2 + 2\left.\frac{\partial^2 L}{\partial u_1 \partial u_2}\right|_* \triangle u_1 \triangle u_2 + \left.\frac{\partial^2 L}{\partial u_2^2}\right|_* \triangle u_2^2 \right\}$$

$$= \frac{1}{2}\begin{bmatrix} \triangle u_1 & \triangle u_2 \end{bmatrix} \begin{bmatrix} \left.\dfrac{\partial^2 L}{\partial u_1^2}\right|_* & \left.\dfrac{\partial^2 L}{\partial u_1 \partial u_2}\right|_* \\ \left.\dfrac{\partial^2 L}{\partial u_2 \partial u_1}\right|_* & \left.\dfrac{\partial^2 L}{\partial u_2^2}\right|_* \end{bmatrix} \begin{bmatrix} \triangle u_1 \\ \triangle u_2 \end{bmatrix}$$

$$= \frac{1}{2} \triangle \boldsymbol{u}^T \left[\frac{\partial^2 L}{\partial \boldsymbol{u}^2} \right] \triangle \boldsymbol{u}$$

– NOTE: $\dfrac{\partial^2 L}{\partial \boldsymbol{u}^2}$ is the *Hessian* of L

- This result clearly indicates that the 2nd-order term in the Taylor series expansion will be greater than or equal to zero if

$$\frac{\partial^2 L}{\partial \boldsymbol{u}^2} \text{ is positive semidefinite}$$

Sufficient Conditions For a Local Minimum

$$\frac{\partial L}{\partial \boldsymbol{u}} = \mathbf{0} \qquad \frac{\partial^2 L}{\partial \boldsymbol{u}^2} \quad \text{positive definite}$$

Necessary Conditions For a Local Minimum

$$\frac{\partial L}{\partial \boldsymbol{u}} = \mathbf{0} \qquad \frac{\partial^2 L}{\partial \boldsymbol{u}^2} \quad \text{positive semidefinite}$$

N-Parameter Problem

- The vector notation introduced in the 2-parameter problem above is ideally suited to the N-parameter problem and leads to precisely the same necessary and sufficient conditions as those stated above

Example 1

- Consider the following four cases:

1. $f(\boldsymbol{x}) = x_1^2 + x_2^2$

$$\frac{\partial f}{\partial \boldsymbol{x}} = \begin{bmatrix} 2x_1 & 2x_2 \end{bmatrix} = \mathbf{0}$$

$$\Rightarrow x_1 = x_2 = 0$$

$$\frac{\partial^2 f}{\partial \boldsymbol{x}^2} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} > 0$$

2. $f(\boldsymbol{x}) = -x_1^2 + -x_2^2$

$$\frac{\partial f}{\partial \boldsymbol{x}} = \begin{bmatrix} -2x_1 & -2x_2 \end{bmatrix} = \mathbf{0}$$

$$\Rightarrow x_1 = x_2 = 0$$

$$\frac{\partial^2 f}{\partial \boldsymbol{x}^2} = \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix} < 0$$

3. $f(\boldsymbol{x}) = x_1^2 - x_2^2$

$$\frac{\partial f}{\partial \boldsymbol{x}} = \begin{bmatrix} 2x_1 & -2x_2 \end{bmatrix} = \mathbf{0}$$

$$\Rightarrow x_1 = x_2 = 0$$

$$\frac{\partial^2 f}{\partial \boldsymbol{x}^2} = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix} \equiv \text{indeterminate}$$

4. $f(\boldsymbol{x}) = -x_1^2 + x_2^2$

$$\frac{\partial f}{\partial \boldsymbol{x}} = \begin{bmatrix} -2x_1 & 2x_2 \end{bmatrix} = \mathbf{0}$$

$$\Rightarrow x_1 = x_2 = 0$$

$$\frac{\partial^2 f}{\partial \boldsymbol{x}^2} = \begin{bmatrix} -2 & 0 \\ 0 & 2 \end{bmatrix} \equiv \text{indeterminate}$$

- Corresponding function surface graphs are depicted in the following figures

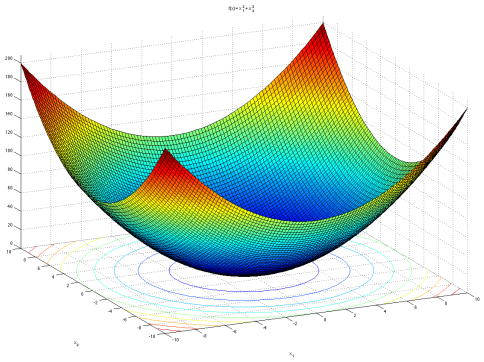


Figure 3.4 $f(\boldsymbol{x}) = x_1^2 + x_2^2$

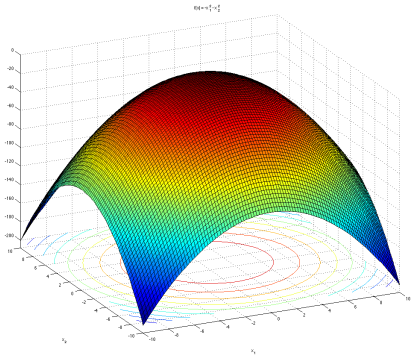


Figure 3.5 $f(\boldsymbol{x}) = -x_1^2 - x_2^2$

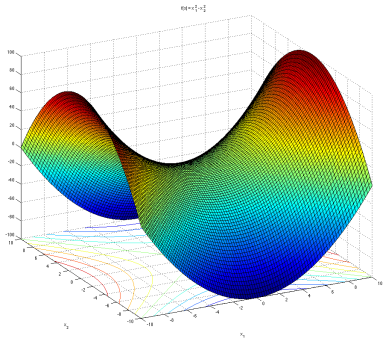


Figure 3.6 $f(\boldsymbol{x}) = x_1^2 - x_2^2$

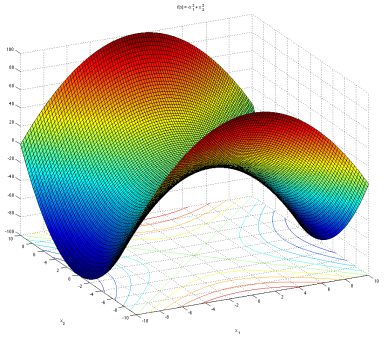


Figure 3.7 $f(\boldsymbol{x}) = -x_1^2 + x_2^2$

Example 2

- Consider the objective function given by:

$$f(\boldsymbol{x}) = (x_1 - x_2 + 2)^2 + (x_1 + x_2 - 4)^4$$

$$\left[\frac{\partial f}{\partial \boldsymbol{x}}\right]^T = \begin{bmatrix} 2(x_1 - x_2 + 2) + 4(x_1 + x_2 - 4)^3 \\ -2(x_1 - x_2 + 2) + 4(x_1 + x_2 - 4)^3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$4(x_1 - x_2 + 2) = 0$$
$$x_1 - x_2 = -2$$

$$x_1 + x_2 - 4 = 0$$
$$x_1 + x_2 = 4$$

$\Rightarrow$

$$\boldsymbol{x} = \begin{bmatrix} 1 & 3 \end{bmatrix}^T$$

$$\frac{\partial^2 f}{\partial \boldsymbol{x}^2} = \begin{bmatrix} 2 + 12(x_1 + x_2 - 4)^2 & -2 + 12(x_1 + x_2 - 4)^2 \\ -2 + 12(x_1 + x_2 - 4)^2 & 2 + 12(x_1 + x_2 - 4)^2 \end{bmatrix}$$

$$\frac{\partial^2 f}{\partial \boldsymbol{x}^2} = \left[\begin{array}{c|c} 2 + 12(x_1 + x_2 - 4)^2 & -2 + 12(x_1 + x_2 - 4)^2 \\ \hline -2 + 12(x_1 + x_2 - 4)^2 & 2 + 12(x_1 + x_2 - 4)^2 \end{array}\right]$$

$$= \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix} \qquad \lambda_1 = 4,\ \lambda_2 = 0$$

Necessary conditions are satisfied; sufficient conditions are not.

$$f(\boldsymbol{x}) \geq 0 \ \forall\ (x_1,\ x_2)$$
$$f(\boldsymbol{x}) = 0 \text{ for } (1,\ 3)$$

$$\Rightarrow \quad f(1,\ 3) \text{ is a local minimum}$$

- For many multi-parameter optimization problems, the necessary condition

$$\frac{\partial L}{\partial \boldsymbol{u}} = 0$$

generates a set of equations that are too difficult to solve analytically.

- So what do we do? Compute numerically!

3.3: Line Search Methods for Unconstrained Optimization

- Here we seek an iterative method for unconstrained optimization, i.e., one that iterates $\boldsymbol{u}^{(k)}$ so that it moves rapidly toward the neighborhood of a local minimizer $\boldsymbol{u}^*$ and converges rapidly to the point $\boldsymbol{u}^*$ itself
  - Order of convergence is a useful measure of algorithm behavior
    - Define the error vector,
    $$\boldsymbol{h}^{(k)} = \boldsymbol{u}^{(k)} - \boldsymbol{u}^*$$
    - Then if $\boldsymbol{h}^{(k)} \to 0$ (convergence), it may be possible to give *local convergence* results:
    $$\frac{\left\|\boldsymbol{h}^{(k+1)}\right\|}{\left\|\boldsymbol{h}^{(k)}\right\|^p} \to a$$
    where $a > 0$ implies the order of convergence is p^{th} order.
  - Here the notation $\|\bullet\|$ denotes a vector norm and,
    - $p = 1$ $\Rightarrow$ first order or linear convergence
    - $p = 2$ $\Rightarrow$ second order or quadratic convergence

Line Search Algorithms

- The basic idea is to search for a minimum function value along coordinate directions, or in more general directions
- First we generate an initial estimate $\boldsymbol{u}^{(1)}$, then for each k^{th} iteration,
  1. Determine a direction of search $\boldsymbol{s}^{(k)}$
  2. Find $\alpha^{(k)}$ to minimize $L(\boldsymbol{u}^{(k)} + \alpha \boldsymbol{s}^{(k)})$ with respect to α
  3. Set $\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + \alpha^{(k)} \boldsymbol{s}^{(k)}$

- Different methods correspond to different ways of choosing $\boldsymbol{s}^{(k)}$ in step 1
- Step 2 is the line search subproblem and involves sampling $L(\boldsymbol{u})$ (and possibly its derivatives) along the line
  - Ideally, an exact minimizing value of $\alpha^{(k)}$ is required, but this is not practical in a finite number of steps
- It is apparent that the slope of $dL/d\alpha$ at $\alpha^{(k)}$ must be zero, which gives

$$\nabla L^{(k+1)T} \boldsymbol{s}^{(k)} = 0$$

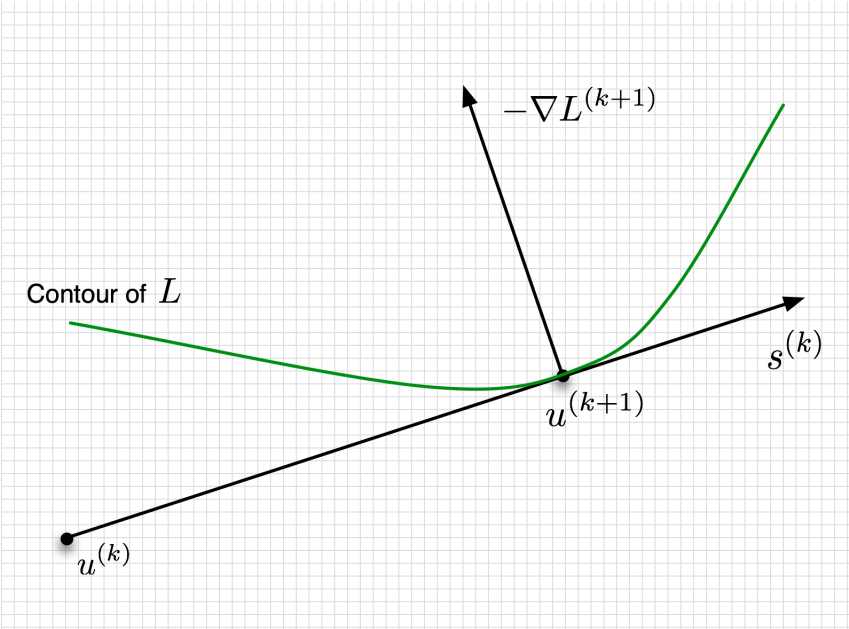


Figure 3.8 Exact line search

- Generally, *inexact* or *approximate* line searches are used to satisfy this minimizing condition
- Requirement that $L^{(k+1)} < L^{(k)}$ is unsatisfactory by itself because reductions in L might be negligible
- Aim of a line search is to:

  - find a step $\alpha^{(k)}$ which gives a significant reduction in L on each iteration
  - ensure points are not near the extremes of the interval $[0,\ \bar{\alpha}^{(k)}]$, where $\bar{\alpha}^{(k)}$ denotes the least positive value of α for which $L(\boldsymbol{u}^{(k)} + \alpha \boldsymbol{s}^{(k)}) = L(\boldsymbol{u}^{(k)})$

- *Goldstein Conditions* meet the above requirements:

  - $f(\alpha) \leq f(0) + \alpha\rho f'(0)$
  - $f(\alpha) \geq f(0) + \alpha(1-\rho) f'(0)$

  $\rho \in \left(0,\ \frac{1}{2}\right)$ is a fixed parameter; the geometry is illustrated in accompanying Figure 3.9 .

- The second of these conditions might exclude the minimizing point of $f(\alpha)$, so an alternate condition is often used:

$$|f'(\alpha)| \leq -\sigma f'(0)$$

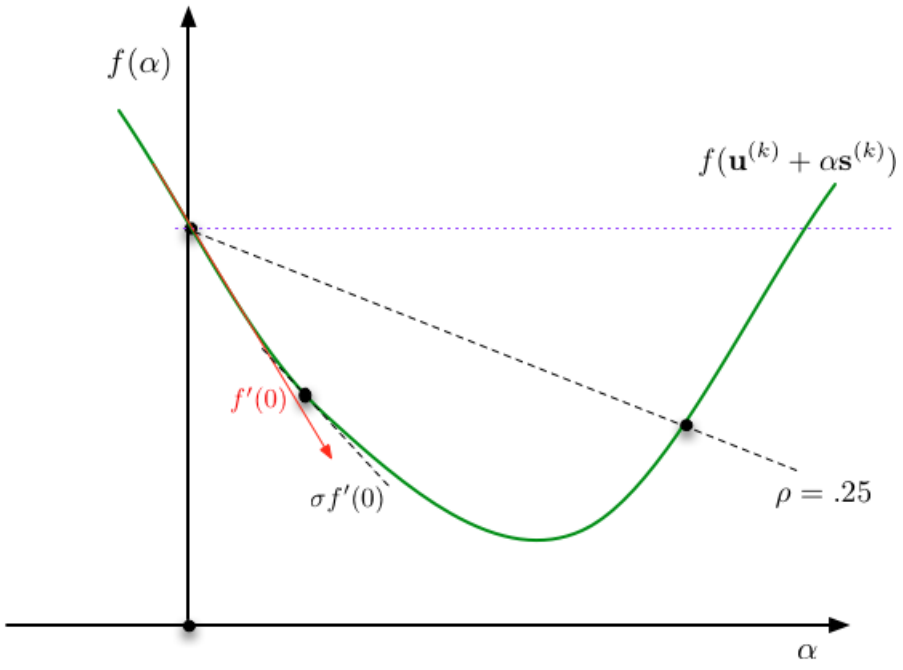


Figure 3.9 Line search geometry

- If $\hat{\alpha}$ is the least value of $\alpha > 0$ at which the $f(\alpha)$ curve intersects the ρ-line, and $\sigma > \rho$, then it can be shown there exists an interal of acceptable points satisfying the Goldstein conditions (proof omitted).
- In practice, it is customary to use $\sigma = 0.1$ and $\rho = 0.01$, though the behavior is not really too sensitive to choice of ρ
- Line search algorithm comprises two phases: *bracketing* and *sectioning:*
  - Bracketing: iterates α_i move out to the right in increasingly large jumps until an acceptable interval is located
  - Sectioning: generates a sequence of brackets $\left[a_j,\ b_j\right]$ whose lengths tend toward zero

3.4: Line Search Algorithm: Bracketing

Bracketing Algorithm

For $i = 1, 2, \ldots$

1. evaluate $f(\alpha_i)$
2. if $f(\alpha_i) \leq f_{min} \Rightarrow$ terminate line search
3. if $f(\alpha_i) > f(0) + \alpha\rho f'(0)$ or $f(\alpha_i) \geq f(\alpha_{i-1})$
   (a) $a_i = \alpha_{i-1}$
   (b) $b_i = \alpha_i$
   $\Rightarrow$ terminate bracket
4. evaluate $f'(\alpha_i)$
5. if $|f'(\alpha_i)| \leq -\sigma f'(0) \Rightarrow$ terminate line search
6. if $f'(\alpha_i) \geq 0$
   (a) $a_i = \alpha_i$
   (b) $b_i = \alpha_{i-1}$
   $\Rightarrow$ terminate bracket
7. if $\mu \leq 2\alpha_i - \alpha_{i-1}$
   (a) $\alpha_{i+1} = \mu$
8. else
   (a) choose $\alpha_{i+1} \in [2\alpha_i - \alpha_{i-1},\ \min(\mu,\ \alpha_i + \tau_1(\alpha_i - \alpha_{i-1}))]$

end

- Parameter τ_1 is preset and governs the size of the jumps; $\tau_1 = 9$ is a reasonable choice
- Choice of α_{i+1} can be made in any way, but a sensible choice is to minimize a cubic polynomial interpolating $f(\alpha_i)$, $f'(\alpha_i)$, $f(\alpha_{i-1})$, and $f'(\alpha_{i-1})$.

Example: Bracketing

- Consider the quadratic function

$$f(\alpha) = 0.5 + 2(\alpha - 3)^2$$

Since this is a quadratic, it's somewhat of a special case. For this example, we choose the following parameters for the start of the line search:

$$\alpha_0 = 0 \quad \alpha_1 = 1 \quad \rho = 0.25 \quad \sigma = 0.5$$

  - For simplicity, we select $\bar{f} = 0$ as an absolute lower bound (although it's obviously bounded by 0.5)

- We begin the first iteration of the bracketing algorithm ($i = 1$)

1. $f(\alpha_1) = 8.5$
2. Test: $f(\alpha_1) \leq \bar{f}$ No
3. Test: $f(\alpha_1) > f(0) + \alpha_i \rho f'(0)$

$$8.5 > 18.5 + (1)(0.25)(-12) = 15.5 \quad \text{No}$$

4. $f'(\alpha_1) = -8$
5. Test: $|f'(\alpha_1)| \leq -\sigma f'(0)$

$$|-8| \leq -(0.5)(-12) = 6 \quad \text{No}$$

6. Test: $f'(\alpha_1) > 0$ No
7. Test: $\mu \leq 2\alpha_1 - \alpha_0$ No

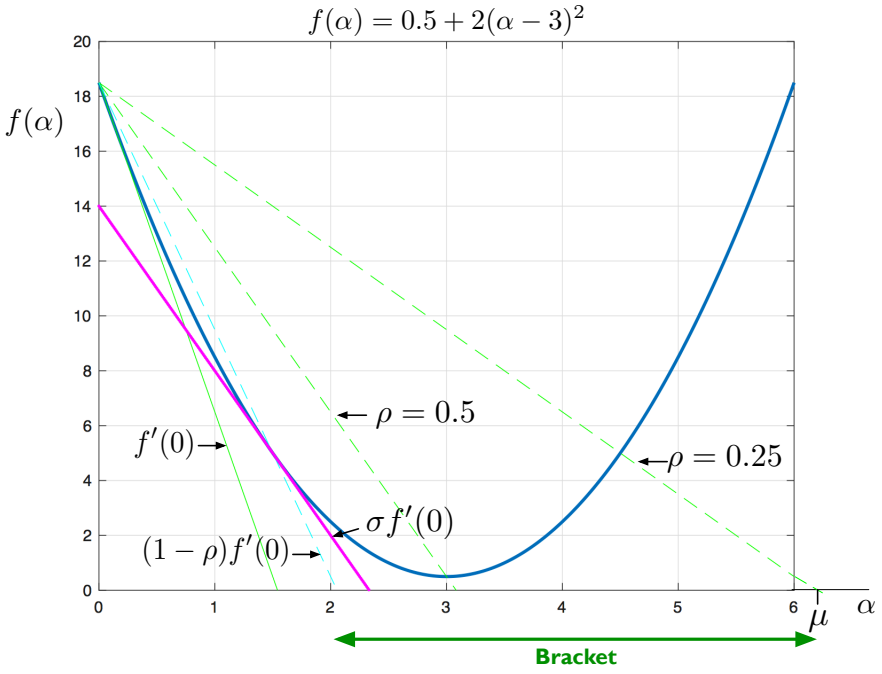
$\Rightarrow$ Therefore, choose the next iterate within the interval

$$\alpha_2 \in \left[\; 2, \;\; \min\left(\; 6.1667, \;\; 1 + \tau_1 \left(\alpha_1 - \alpha_0 \right) \; \right) \; \right]$$

Subsituting values,

$$\alpha_2 \in \left[\; 2, \;\; \min\left(\; 6.1667, \;\; 1 + 9 \left(1 - 0 \right) \; \right) \; \right] = \left[\; 2, \;\; 6.1667 \; \right]$$

- Quadratic interpolation over this interval will give $\alpha_2 = 3$ as the next iterate; this will terminate the line search in the next bracket iteration at Step 5
  - The bracketing sequence is depicted in the plot below



3.5: Line Search Algorithm: Sectioning

Sectioning Algorithm

For $j = i,\ i+1, \ldots$

1. choose $\alpha_j \in \left[a_j + \tau_2\left(b_j - a_j\right),\quad b_j - \tau_3\left(b_j - a_j\right)\right]$
2. evaluate $f(\alpha_j)$
3. if $f(\alpha_j) > f(0) + \rho\alpha_j f'(0)$ or $f(\alpha_j) \geq f(a_j)$
   (a) $a_{j+1} = a_j$
   (b) $b_{j+1} = \alpha_j$
4. else
   (a) evaluate $f'(\alpha_j)$
   (b) if $\left|f'(\alpha_j)\right| \leq -\sigma f'(0) \quad \Rightarrow$ terminate line search
      i. $a_{j+1} = \alpha_j$
   (c) if $\left(b_j - a_j\right) f'(\alpha_j) \geq 0$
      i. $b_{j+1} = a_j$
   (d) else
      i. $b_{j+1} = b_j$
   (e) end

end

- Parameters τ_2 and τ_3 are preset and restrict α_j from getting too close to the extremes of the interval $\left[a_j,\ b_j\right]$:

$$0 < \tau_2 < \tau_3 \leq \frac{1}{2}$$

- Typical values are: $\tau_2 = 0.1$ and $\tau_3 = 0.5$

Polynomial Interpolation

- For the quadratic case, we can define the 2^{nd}-order polynomial

$$p_{\mathrm{q}}(z) = p_2 z^2 + p_1 z + p_0$$

Considering the normalized interval $z = [0,\ 1]$ corresponding to $\left[a_j,\ b_j\right]$ allows us to write the interpolation conditions:

$$\begin{aligned} p_{\mathrm{q}}(0) &= f(a_j) \\ p_{\mathrm{q}}(1) &= f(b_j) \\ p'_{\mathrm{q}}(0) &= f'_z(a_j) \\ p'_{\mathrm{q}}(1) &= f'_z(b_j) \end{aligned}$$

Assuming we can compute the values $f(a_j)$, $f'_z(a_j)$, and $f(b_j)$, substituting for z allows us to write

$$\begin{aligned} p_q\,(0) &= p_0 = f(a_j) \\ p'_q\,(0) &= p_1 = f'_z(a_j) \\ p_q\,(1) &= p_2 + p_1 + p_0 = f(b_j) \end{aligned}$$

or, assembling in matrix-vector form,

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_2 \\ p_1 \\ p_0 \end{bmatrix} = \begin{bmatrix} f(b_j) \\ f'_z(a_j) \\ f(a_j) \end{bmatrix}$$

– Solving this system of equations yields

$$\begin{aligned} p_0 &= f(a_j) \\ p_1 &= f'_z(a_j) \\ p_2 &= f(b_j) - f(a_j) - f'_z(a_j) \end{aligned}$$

giving the interpolating polynomial:

$$p_q(z) = \left[f(b_j) - f(a_j) - f_z'(a_j)\right] z^2 + \left[f_z'(a_j)\right] z + f(a_j)$$

– Note that the mapping transformation is given by

$$\alpha = a + z\,(b-a)$$

where by the chain rule we have

$$f_z' = \frac{df}{dz} = \frac{df}{d\alpha} \cdot \frac{d\alpha}{dz} = (b-a)\,\frac{df}{d\alpha}$$

which relates the derivatives of the mapped variables.

– The inverse mapping is,

$$z = \frac{1}{b-a}\,(\alpha - a)$$

- If in addition $f'(b_j)$ is available, we can find the cubic interpolating polynomial:

$$p_c(z) = p_3 z^3 + p_2 z^2 + p_1 z + p_0$$

where we assemble the interpolation equations:

$$\begin{aligned} p_c(1) &= p_3 + p_2 + p_1 + p_0 = f\left(b_j\right) \\ p_c'(1) &= 3p_3 + 2p_2 + p_1 = f_z'\left(b_j\right) \\ p_c'(0) &= p_1 = f_z'\left(a_j\right) \\ p_c(0) &= p_0 = f\left(a_j\right) \end{aligned}$$

or in matrix form,

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 3 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_3 \\ p_2 \\ p_1 \\ p_0 \end{bmatrix} = \begin{bmatrix} f\left(b_j\right) \\ f_z'\left(b_j\right) \\ f_z'\left(a_j\right) \\ f\left(a_j\right) \end{bmatrix}$$

Thus giving the solution:

$$\begin{aligned} p_0 &= f(a_j) \\ p_1 &= f_z'(a_j) \\ p_2 &= 3\left(f(b_j) - f(a_j)\right) - 2f_z'(a_j) - f_z'(b_j) \\ p_3 &= f_z'(a_j) + f_z'(b_j) - 2\left(f(b_j) - f(a_j)\right) \end{aligned}$$

Example 3

- Consider the function,

$$f(\boldsymbol{x}) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$$

and let

$$\boldsymbol{x}^{(k)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{s}^{(k)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

- We then have,

$$\begin{aligned} f(\alpha) &= 100\alpha^4 + (1 - \alpha)^2 \\ f'(\alpha) &= 400\alpha^3 - 2(1 - \alpha) \end{aligned}$$

- Choosing parameters,

$$\begin{aligned} \sigma &= 0.1 \\ \rho &= 0.01 \\ \tau_1 &= 9 \\ \tau_2 &= 0.1 \\ \tau_3 &= 0.5 \end{aligned}$$

gives the following results for the cases $\alpha_1 = 0.1$ and $\alpha_1 = 1$:

Iteration	0	1	2	3	4
$\alpha_1 = 0.1$					
α	0	0.1	0.2	0.160948	
$f(\alpha)$	1	0.82	0.8	0.771111	
$f'(\alpha)$	-2	-1.4	1.6	-0.010423	
$\alpha_1 = 1$					
α	0	1	0.1	0.19	0.160922
$f(\alpha)$	1	100	0.82	0.786421	0.771112
$f'(\alpha)$	-2		-1.4	1.1236	-0.011269

Table 3.1 LINE SEARCH EXAMPLE

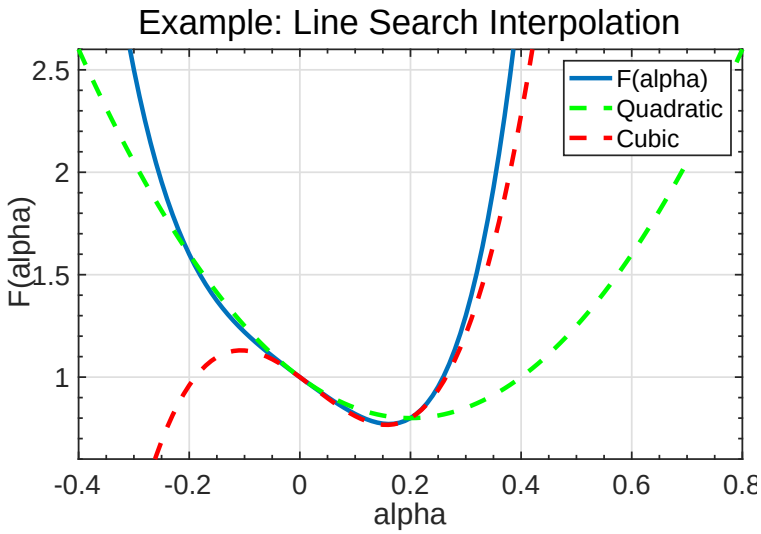


Figure 3.10 Example 3: $f(\alpha) = 100\alpha^4 + (1-\alpha)^2$

3.6: Descent Methods

- Descent methods are line search methods where the the search direction satisfies the *descent property*:

$$\boldsymbol{s}^{(k)T}\boldsymbol{g}^{(k)} < 0$$

  where

$$\boldsymbol{g}^{(k)} = \nabla L(\boldsymbol{u})$$

- This condition ensures
  - The slope of $dL/d\alpha$ is always negative at $\alpha = 0$ (unless $\boldsymbol{u}^{(k)}$ is a stationary point)
  - The function $L(\boldsymbol{u})$ can be reduced in the line search for some $\alpha^{(k)} > 0$

Steepest Descent Methods

Steepest descent is defined by the condition:

$$\boldsymbol{s}^{(k)} = -\boldsymbol{g}^{(k)}$$

for all k .

- This condition ensures that $L(\boldsymbol{u})$ decreases most rapidly local to $\boldsymbol{u}^{(k)}$
- Although appealing, the steepest descent method is not suited for practical use, largely because:
  - it usually exhibits oscillatory behavior
  - it usually terminates far from the exact solution due to round-off errors

- Inadequacy of steepest descent is due mostly to the model: the steepest descent property along the line holds only at $\alpha = 0$ (not for all α)
- An exception occurs for *quadratic* models, which we'll investigate in more detail later

Convergence

- It is important to be able to determine when an algorithm has converged to an acceptable solution
- A useful test would be: $L^{(k)} - L^* \leq \epsilon$ or $\left| \boldsymbol{x}_i^{(k)} - \boldsymbol{x}_i^* \right| \leq \epsilon_i$, but these are not practical because they require the solution!
- A practical alternative is: $\left\| \boldsymbol{g}^{(k)} \right\| \leq \epsilon$, though in practice it's hard to choose an appropriate ϵ
- Far more practical are tests of the following form:

$$\left| x_i^{(k+1)} - x_i^{(k)} \right| \leq \epsilon_i \qquad \forall i$$

or

$$L^{(k+1)} - L^{(k)} \leq \epsilon$$

3.7: Newton's Method

- It was shown previously that there is a great advantage to deriving a method based on a quadratic model
- Newton's method is the most straightforward such technique
- The key to this algorithm is that the values of $\boldsymbol{u}$ which minimize L are the same as the ones which satisfy

$$\frac{\partial L}{\partial \boldsymbol{u}} = \mathbf{0}$$

- So, we'll set up an algorithm which searches for a solution to this problem
- Writing the truncated Taylor series expansion of $L(\boldsymbol{u})$ about $\boldsymbol{u}^{(k)}$:

$$L(\boldsymbol{u}^{k)} + \boldsymbol{\delta}) \approx q^{(k)}(\boldsymbol{\delta}) = L^{(k)} + \boldsymbol{g}^{(k)T}\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^T G^{(k)}\boldsymbol{\delta}$$

  where $\boldsymbol{\delta} = \boldsymbol{u} - \boldsymbol{u}^{(k)}$, and $q^{(k)}(\boldsymbol{\delta})$ is the resulting quadratic approximation for iteration k.
- Iteration $\boldsymbol{u}^{(k+1)}$ is computed as $\boldsymbol{u}^{(k)} + \boldsymbol{\delta}^{(k)}$ where the correction $\boldsymbol{\delta}^{(k)}$ minimizes $q^{(k)}(\boldsymbol{\delta})$
- Method requires the zero, first and second order derivatives of $L(\boldsymbol{u})$
- The basic algorithm can be written:
  - solve for $G^{(k)}\boldsymbol{\delta} = -\boldsymbol{g}^{(k)}$ for $\boldsymbol{\delta} = \boldsymbol{\delta}^{(k)}$
  - set $\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + \boldsymbol{\delta}^{(k)}$
- Newton's method exhibits second-order convergence

Complete Algorithm $\Rightarrow$

1. Compute the general functions $\boldsymbol{g}^{(k)} = \dfrac{\partial L}{\partial \boldsymbol{u}}$ and $G^{(k)} = \dfrac{\partial^2 L}{\partial \boldsymbol{u}^2}$ a priori
2. Choose starting value $\boldsymbol{u}^{(1)}$
3. Evaluate $\boldsymbol{g}^{(1)}$ and $G^{(1)}$ at $\boldsymbol{u}^{(1)}$
4. Solve for $\boldsymbol{\delta}^{(1)}$ (solve the set of simultaneous equations $G^{(1)}\boldsymbol{\delta}^{(1)} = -\boldsymbol{g}^{(1)}$)
5. Compute $\boldsymbol{u}^{(2)} = \boldsymbol{u}^{(1)} + \boldsymbol{\delta}^{(1)}$
6. Repeat steps (3) - (5) for increasing values of k until convergence condition is satisfied

- The biggest problem with this algorithm is that the calculation of the Hessian $\dfrac{\partial^2 L}{\partial \boldsymbol{u}^2}$ may be extremely tedious

Example 4

Let

$$L(\boldsymbol{u}) = u_1^4 + u_1 u_2 + (1 + u_2)^2$$

where

$$\boldsymbol{u}^{(1)} = \begin{bmatrix} 1.25 \\ -0.2 \end{bmatrix}$$

Implementing Newton's method for values of k from 1 to 7 gives the results summarized in the table below; a graphical representation is

shown in Figure 3.11. Here we use the (idealized) definition of $\boldsymbol{h}^{(k)} = \boldsymbol{u}^k - \boldsymbol{u}^*$.

k	1	2	3	4	5	6	7
$u_1^{(k)}$	1.25	0.9110	0.7451	0.69932	0.6959029	0.6958844	0.6958843
$u_2^{(k)}$	−0.2	−1.455	−1.3726	−1.34966	−1.347951	−1.3479422	−1.3479422
$g_1^{(k)}$	7.6125	1.5683	0.2823	−0.018382	0.0000982235	−0.0000000028559	0
$g_2^{(k)}$	2.8500	0	0	0	0	0	0
$L^{(k)}$	2.8314	−0.4298	−0.5757	−0.582414	−0.5824452	−0.5824452	−0.5824452
$\left\|\boldsymbol{h}^{(k)}\right\|$	1.2727	0.24046	0.0550691	0.00384881	0.0000206765	0.00000000064497	0

Table 3.2 NEWTON'S METHOD EXAMPLE

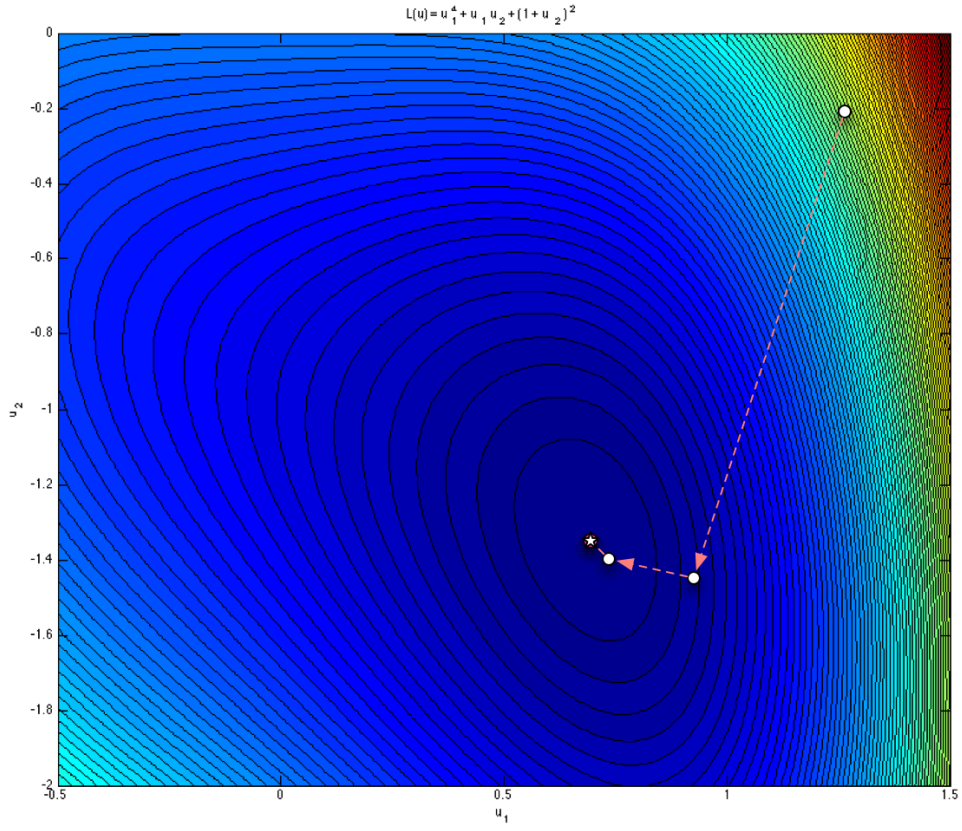


Figure 3.11 Newton's method example

From the above, it can be shown that the ratio $\|\boldsymbol{h}^{(k+1)}\|/\|\boldsymbol{h}^{(k)}\|^2 \rightarrow a$ where $a \approx 1.4$, indicating second-order convergence.

- Basic Newton method is not suitable for a general purpose algorithm:
  - $G^{(k)}$ may not be be positive definite when $\boldsymbol{x}^{(k)}$ is far from the solution
  - even if $G^{(k)}$ is positive definite, algorithm still may not converge
  - convergence can be addressed by using *Newton's method with line search*:

$$\boldsymbol{s}^{(k)} = -\boldsymbol{G}^{(k)-1}\boldsymbol{g}^{(k)} \Rightarrow \text{search direction}$$

Example 5

- Returning to the function of Example 4,

$$L(\boldsymbol{u}) = u_1^4 + u_1 u_2 + (1+u_2)^2$$

If we choose

$$\boldsymbol{x}^{(1)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

then we have,

$$\boldsymbol{g}^{(1)} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \quad \boldsymbol{G}^{(1)} = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}, \quad \boldsymbol{s}^{(1)} = \begin{bmatrix} -2 \\ 0 \end{bmatrix}$$

- A search along $\pm\boldsymbol{s}^{(1)}$ changes only the x_1 component of $\boldsymbol{x}^{(1)}$ which finds $x_1 = 0$ as the minimizing value in the search $\Rightarrow$ algorithm fails to make progress!
- Reason for this is: $\boldsymbol{s}^{(1)T}\boldsymbol{g}^{(1)} = 0 \Rightarrow$ directions are not downhill!
  - This stems from the fact that $\lambda(G^{(1)}) = 2.1412,\ -0.4142 \Rightarrow G^{(1)}$ not positive definite

- But function $L(\boldsymbol{u})$ has a well defined minimum which can be found by searching along steepest descent direction

3.8 Modifications to Newton's Method

- Clearly, some modification is required to make the Newton algorithm generally applicable
- Modification 1: Revert to steepest descent direction $\boldsymbol{s}^{(k)} = -\boldsymbol{g}^{(k)}$ whenever $G^{(k)}$ is not positive definite
  - Unfortunately, this exhibits slow oscillatory behavior since the method ignores information in the model quadratic function
- Modification 2: Adjust the Newton search direction by giving it a bias towards the steepest descent vector, $-\boldsymbol{g}^{(k)}$:

$$\left(G^{(k)} + \nu I\right) \boldsymbol{s}^{(k)} = -\boldsymbol{g}^{(k)}$$

  - Method adds factor ν to the eigenvalues of $G^{(k)}$ to (hopefully) make it positive definite
  - Takes into account more of the function's quadratic information (except in the vicinity of a saddle point)
- Other modifications exist, but they are beyond the scope of this course

1st-order Gradient Methods (a simplified approach)

- Instead of using knowledge about the "curvature" of L to help us find $\boldsymbol{\delta}$, let's simply step by some amount (???) in the direction of decreasing L until we reach a minimum

- the solution of the linear equation $G\boldsymbol{\delta} = -\boldsymbol{g}$ can be expressed as

$$\boldsymbol{\delta} = -G^{-1}\boldsymbol{g}$$

(though in practice this is not how we would solve it)

- we now replace G^{-1} by a positive scalar constant K so that

$$\boldsymbol{\delta} = -K\boldsymbol{g}$$

- we can now perform the same iterative algorithm outlined above

- Can I convince you that this will work?
  - remember, $L(\boldsymbol{u} + \boldsymbol{\delta}) = L(\boldsymbol{u}) + \dfrac{\partial L}{\partial \boldsymbol{u}}\boldsymbol{\delta} + \mathscr{O}(2)$
  - if $\boldsymbol{\delta} = -K\dfrac{\partial L}{\partial \boldsymbol{u}}$, then

$$L(\boldsymbol{u} + \boldsymbol{\delta}) - L(\boldsymbol{u}) \approx -K\left|\frac{\partial L}{\partial \boldsymbol{u}}\right|^2 < 0$$

  - so, *to first order*, we are moving in the right direction!

- This verification also provides some insight into the problem of selecting K :
  - if K is too big, the 2nd-order term in the Taylor series may become significant and the algorithm may overshoot the stationary point and not converge at all.
  - if K is too small, the higher-order-terms will be truly insignificant but it may take forever to get to the solution
  - how is it done in practice? Vary K during the iteration process

3.9: Quasi-Newton Methods

- Main disadvantage of Newton's method is that the user must supply explicit formulae to compute the second derivative matrix G
- But methods very similar to Newton's method can be derived when only first derivative formulae are available
- One straighforward approach is the *Finite Difference Newton Method:*
  - estimate $G^{(k)}$ by using finite differences in the gradient vectors, i.e., the $(i,\ j)$ element of estimate $\hat{G}^{(1)}$ is computed as:

$$\hat{G}_{ij} = \frac{\left(\boldsymbol{g}_j(\boldsymbol{x}^{(k)} + h_i \boldsymbol{e}_i) - \boldsymbol{g}_j^{(k)}\right)}{h_i}$$

    where h_i is an increment length in the coordinate direction, $\boldsymbol{e}_i$.
  - make $\hat{G}$ symmetric by computing

$$\hat{G}_s = \frac{1}{2}\left(\hat{G} + \hat{G}^T\right)$$

  - use $\hat{G}_s$ in place of $G^{(k)}$ in Newton's method
- The method can be useful, but has some disadvantages:
  - $\hat{G}_s$ may not be positive definite
  - n gradient evaluations are required to estimate $G^{(k)}$
  - a set of linear equations must be solved at each iteration

Quasi-Newton Methods

- *Quasi-Newton methods* avoid some of the disadvantages outlined above by –
  - employing Newton's method with line search

  - approximating $G^{(k)-1}$ by a symmetric positive definite matrix $H^{(k)}$ which is updated at each iteration

- Basic Algorithm:
  - initialize $H^{(1)}$ to any positive definite matrix ($H^{(k)} = I$ is a good choice)
  - set search direction $\boldsymbol{s}^{(k)} = -H^{(k)}\boldsymbol{g}^{(k)}$
  - perform line search along $\boldsymbol{s}^{(k)}$ giving $\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + \alpha^{(k)}\boldsymbol{s}^{(k)}$
  - update $H^{(k)}$ giving $H^{(k+1)}$
- Advantages to this method:
  - only first derivatives are required
  - positive definite $H^{(k)}$ implies the descent property
  - order of n^2 multiplications per iteration
- New aspect is the update calculation of $H^{(k+1)}$ from $H^{(k)}$
  - attempts to augment $H^{(k)}$ with second derivative information gained from k^{th} iteration
  - ideally, want the update to change $H^{(1)}$ into a close approximation of $G^{(k)-1}$
  - one method of doing this involves defining the differences:

$$\boldsymbol{\delta}^{(k)} = \alpha^{(k)}\boldsymbol{s}^{(k)} = \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}$$
$$\boldsymbol{\gamma}^{(k)} = \boldsymbol{g}^{(k+1)} - \boldsymbol{g}^{(k)}$$

    then the Taylor series of the gradient $\boldsymbol{g}^{(k)}$ gives

$$\boldsymbol{\gamma}^{(k)} = G^{(k)}\boldsymbol{\delta}^{(k)} + o\left(\left\|\boldsymbol{\delta}^{(k)}\right\|\right)$$

    where higher order terms can be neglected.

  - since $\boldsymbol{\delta}^{(k)}$ and $\boldsymbol{\gamma}^{(k)}$ can only be calculated after the line search, $H^{(k)}$ does not usually relate them correctly
  - thus, $H^{(k+1)}$ is chosen to correctly relate the differences (*quasi-Newton condition*):

$$H^{(k+1)}\boldsymbol{\gamma}^{(k)} = \boldsymbol{\delta}^{(k)}$$

- Computationally, one approach is to introduce a recursive form:

$$H^{(k+1)} = H^{(k)} + E^{(k)}$$

  - let $E^{(k)}$ be the rank one symmetric matrix $a\boldsymbol{v}\boldsymbol{v}^T$
  - satisfying the quasi-Newton condition requires:

$$H^{(k)}\boldsymbol{\gamma}^{(k)} + a\boldsymbol{v}\boldsymbol{v}^T\boldsymbol{\gamma}^{(k)} = \boldsymbol{\delta}^{(k)}$$

  - which gives rise to the *rank one formula*:

$$H^{(k+1)} = H + \frac{(\boldsymbol{\delta} - H\boldsymbol{\gamma})(\boldsymbol{\delta} - H\boldsymbol{\gamma})^T}{(\boldsymbol{\delta} - H\boldsymbol{\gamma})^T\boldsymbol{\gamma}}$$

Example 6

- Consider the quadratic function:

$$\begin{aligned} L(\boldsymbol{u}) &= 10u_1^2 + u_2^2 \\ &= \boldsymbol{u}^T \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{u} \end{aligned}$$

  where the initial point is given by

$$\boldsymbol{u}^{(1)} = \begin{bmatrix} 0.1 \\ 1 \end{bmatrix}$$

- Gradient:

$$\boldsymbol{g}(\boldsymbol{u}) = \begin{bmatrix} 20u_1 \\ 2u_2 \end{bmatrix}$$

- Hessian:

$$G(\boldsymbol{u}) = \begin{bmatrix} 20 & 0 \\ 0 & 2 \end{bmatrix}$$

Iteration $k = 1$

$$\boldsymbol{g}^{(1)} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \quad H^{(1)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \boldsymbol{s}^{(1)} = \begin{bmatrix} -2 \\ -2 \end{bmatrix} \quad \alpha^{(1)} = 0.0909$$

Iteration $k = 2$

$$\Rightarrow \boldsymbol{u}^{(2)} = \boldsymbol{u}^{(1)} + \alpha^{(1)}\boldsymbol{s}^{(1)} = \begin{bmatrix} .1 \\ 1 \end{bmatrix} + (0.0909)\begin{bmatrix} -2 \\ -2 \end{bmatrix} = \begin{bmatrix} -0.0818 \\ 0.8182 \end{bmatrix}$$

$$\boldsymbol{g}^{(2)} = \begin{bmatrix} -1.6364 \\ 1.6364 \end{bmatrix}$$

$$\boldsymbol{\delta}^{(1)} = \boldsymbol{u}^{(2)} - \boldsymbol{u}^{(1)} = \begin{bmatrix} -0.0818 \\ 0.8182 \end{bmatrix} - \begin{bmatrix} 0.1 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.1818 \\ 0.1818 \end{bmatrix}$$

$$\boldsymbol{\gamma}^{(1)} = \boldsymbol{g}^{(2)} - \boldsymbol{g}^{(1)} = \begin{bmatrix} -1.6364 \\ 1.6364 \end{bmatrix} - \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} -3.6363 \\ -0.3636 \end{bmatrix}$$

$$\boldsymbol{v}^{(1)} = \boldsymbol{\delta}^{(1)} - H^{(1)}\boldsymbol{\gamma}^{(1)} = \begin{bmatrix} -0.1818 \\ 0.1818 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} -3.6363 \\ -.3636 \end{bmatrix} = \begin{bmatrix} 3.4545 \\ 0.1818 \end{bmatrix}$$

$$H^{(2)} = H^{(1)} + \frac{\left(\boldsymbol{\delta}^{(1)} - H^{(1)}\boldsymbol{\gamma}^{(1)}\right)\left(\boldsymbol{\delta}^{(1)} - H^{(1)}\boldsymbol{\gamma}^{(1)}\right)^T}{\left(\boldsymbol{\delta}^{(1)} - H^{(1)}\boldsymbol{\gamma}^{(1)}\right)^T \boldsymbol{\gamma}^{(1)}} = \begin{bmatrix} 0.0550 & -0.0497 \\ -0.0497 & 0.9974 \end{bmatrix}$$

$$\boldsymbol{s}^{(2)} = -H^{(2)}\boldsymbol{g}^{(2)} = \begin{bmatrix} 0.0550 & -0.0497 \\ -0.0497 & 0.9974 \end{bmatrix}\begin{bmatrix} -1.6364 \\ 1.6364 \end{bmatrix} = \begin{bmatrix} 0.1713 \\ -1.7135 \end{bmatrix}$$

$$\alpha^{(2)} = 0.4775$$

Iteration $k = 3$

$$\Rightarrow \boldsymbol{u}^{(3)} = \boldsymbol{u}^{(2)} + \alpha^{(2)}\boldsymbol{s}^{(2)} = \begin{bmatrix} -0.0818 \\ 0.8182 \end{bmatrix} + (0.4775)\begin{bmatrix} 0.1713 \\ -1.7135 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{g}^{(3)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{\delta}^{(2)} = \boldsymbol{u}^{(3)} - \boldsymbol{u}^{(2)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} -0.0818 \\ 0.8182 \end{bmatrix} = \begin{bmatrix} 0.0818 \\ -0.8182 \end{bmatrix}$$

$$\boldsymbol{\gamma}^{(2)} = \boldsymbol{g}^{(3)} - \boldsymbol{g}^{(2)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} -1.6364 \\ 1.6364 \end{bmatrix} = \begin{bmatrix} 1.6364 \\ -1.6364 \end{bmatrix}$$

$$\boldsymbol{v}^{(2)} = \boldsymbol{\delta}^{(2)} - H^{(2)}\boldsymbol{\gamma}^{(2)} = \begin{bmatrix} -0.0895 \\ 0.8953 \end{bmatrix}$$

$$H^{(3)} = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.5 \end{bmatrix}$$

- Note that the algorithm terminates with $\boldsymbol{g}^* = \boldsymbol{0}$ and $H^* = G^{-1}$
- It can be proven that under some mild conditions, the method terminates on a quadratic function in at most $n + 1$ steps, with $H^{(n+1)} = G^{-1}$

- Two other well-known quasi-Newton algorithms are:
  - DAVIDON-FLETCHER-POWELL (DFP):

$$H_{DFP}^{(k+1)} = H + \frac{\boldsymbol{\delta}\boldsymbol{\delta}^T}{\boldsymbol{\delta}^T\boldsymbol{\gamma}} - \frac{H\boldsymbol{\gamma}\boldsymbol{\gamma}^T}{\boldsymbol{\gamma}^T H\boldsymbol{\gamma}}$$

  - BROYDEN-FLETCHER-GOLDFARB-SHANNO (BFGS):

$$H_{BFGS}^{(k+1)} = H + \left(1 + \frac{\boldsymbol{\gamma}^T H\boldsymbol{\gamma}}{\boldsymbol{\delta}^T\boldsymbol{\gamma}}\right)\frac{\boldsymbol{\delta}\boldsymbol{\delta}^T}{\boldsymbol{\delta}^T\boldsymbol{\gamma}} - \left(\frac{\boldsymbol{\delta}\boldsymbol{\gamma}^T H + H\boldsymbol{\gamma}\boldsymbol{\delta}^T}{\boldsymbol{\delta}^T\boldsymbol{\gamma}}\right)$$

    - The BFGS algorithm is perhaps the most widely used Quasi-Newton numerical algorithm and works well with low accuracy line searches

(mostly blank)

(mostly blank)