Lecture 11

Optimization

Ye Ding (丁烨)

Email: y.ding@sjtu.edu.cn

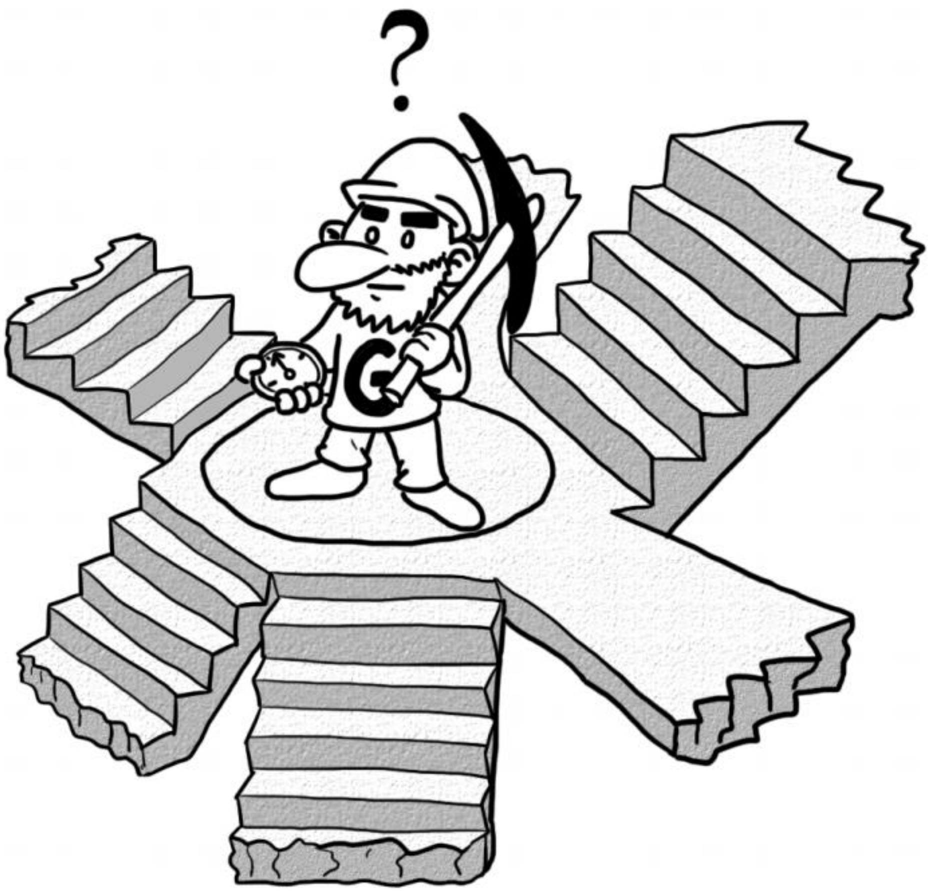
School of Mechanical Engineering

Shanghai Jiao Tong University

Optimization

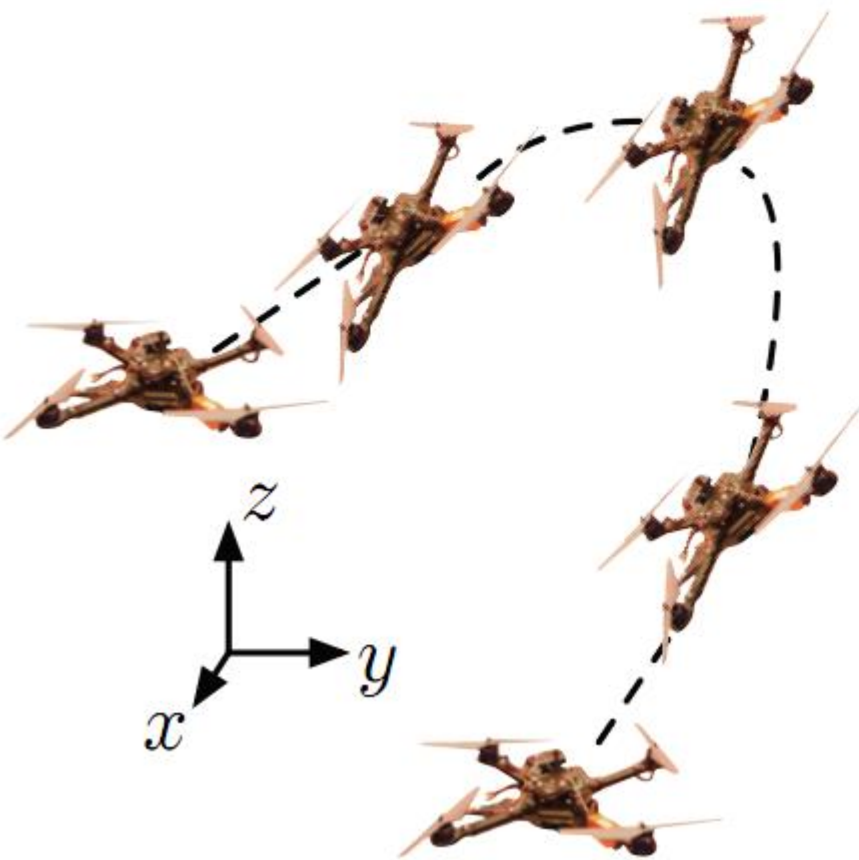
- **Motivation: On Choices**

***R. Battiti*: Everybody carries on his shoulders the responsibility of his choices. It is a heavy weight.**



Optimization

- **Motivation: from Robotics**

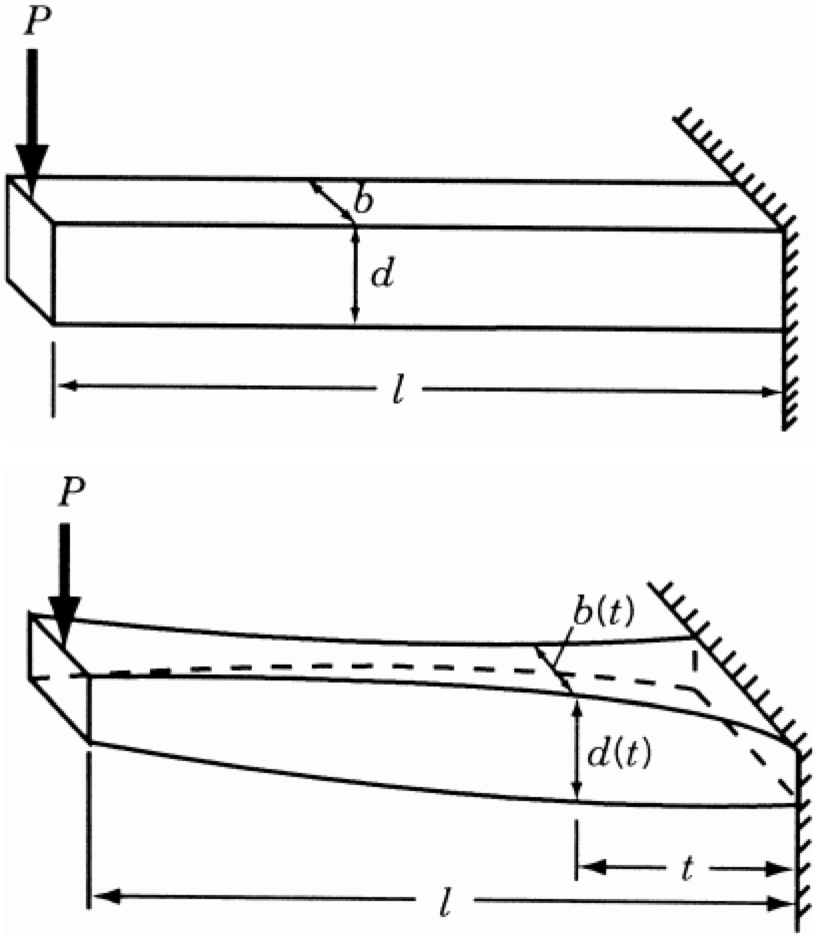


$$\min_{\mathbf{v}(t)} \int_0^{T} L(\mathbf{z})\, dt, \quad \text{s.t. } g(\mathbf{z}) \leq 0.$$

where $L(\mathbf{z})$ is the square of the norm of the input vector.

Optimization

Motivation: from Mechanical Design



Find $\mathbf{X}(t) = \begin{Bmatrix} b(t) \\ d(t) \end{Bmatrix}$ which minimizes

$$f[\mathbf{X}(t)] = \rho \int_0^l b(t)\ d(t)\, dt$$

subject to the constraints

$$\delta_{\text{tip}}[\mathbf{X}(t)] \le \delta_{\max}, \qquad 0 \le t \le l$$

$$b(t) \ge 0, \qquad 0 \le t \le l$$

$$d(t) \ge 0, \qquad 0 \le t \le l$$

Optimization

- **References for Optimization**

[1] J. Nocedal, and S. J. Wright, Numerical Optimization, 2nd ed., New York: Springer, 2006. Chapters 2, 3, 6

[2] Timothy Sauer, Numerical analysis, 2nd ed., Pearson Education, 2012. Chapter 13

[3] Singiresu S. Rao, Engineering Optimization: Theory and Practice, 4th ed., John Wiley & Sons, Inc., 2009. Chapters 5, 6

[4] 袁亚湘, 非线性优化计算方法, 科学出版社, 2008.

Optimization

General Mathematical Programming Problem

$$
\begin{array}{lll}
\text{minimize} & f(\mathbf{x}) & \\
\text{subject to} & h_i(\mathbf{x}) = 0, & i = 1, 2, \ldots, m \\
& g_j(\mathbf{x}) \leq 0, & j = 1, 2, \ldots r \\
& \mathbf{x} \in S. &
\end{array}
$$

- $\mathbf{x}$ is an n-dimensional vector of unknowns; f, h_i , $i = 1, 2, \ldots, m$, and g_j , $j = 1, 2, \ldots, r$, are real-valued functions of $\mathbf{x}$;
- f is the ***objective function***; the equations, inequalities, and set restrictions are ***constraints***.

Optimization

- **One-Dimensional Minimization**
  - **Golden Section Search**
  - **Newton's Method**
  - **Inaccurate Line Search**
- **Unconstrained Optimization**
  - **Steepest Descent Method**
  - **Newton's Method**
  - **Quasi-Newton Method**
- **Linear Programming**
  - **Geometry of Linear Programming**
  - **The Simplex Method**

One-Dimensional Optimization

- **General One-Dimensional Optimization Problem**

$$\min_{x\in\mathbb{R}} \quad f(x)$$

where $\quad x \in \mathbb{R},\ f(x) \in \mathbb{R}$

$$\min_{\alpha\in\mathbb{R}} \quad f(\mathbf{x}+\alpha\mathbf{d})$$

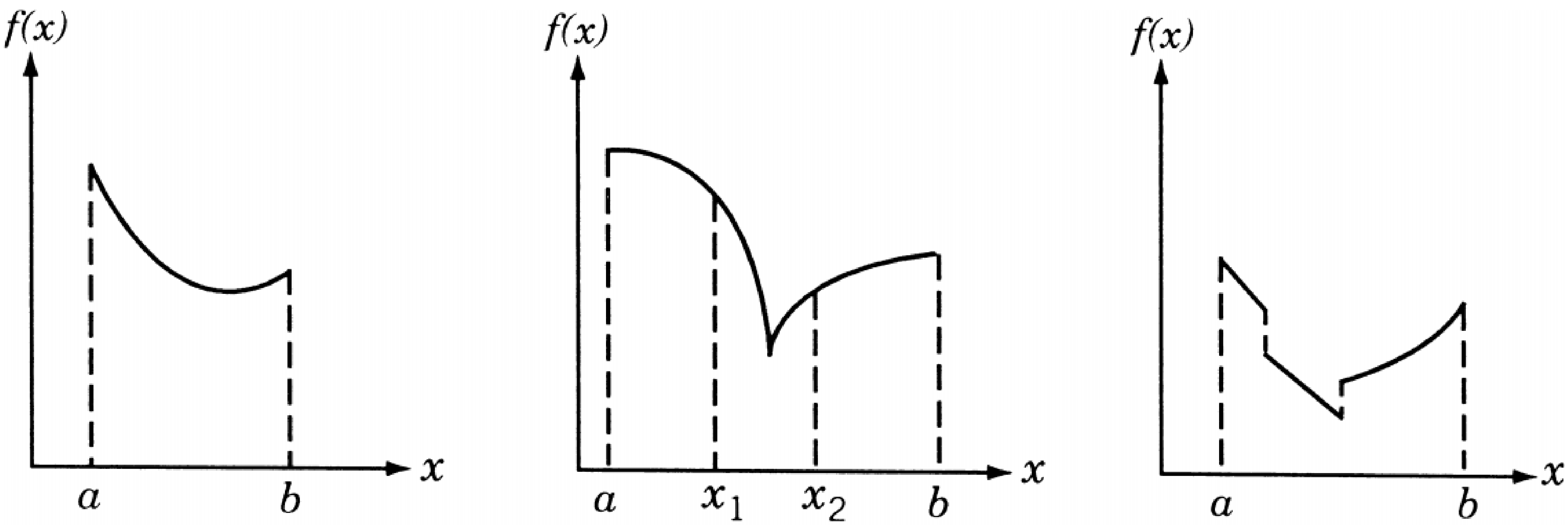
where $\quad \mathbf{x} \in \mathbb{R}^n, \mathbf{d} \in \mathbb{R}^n,\ f(\mathbf{x}) \in \mathbb{R}$

One-Dimensional Optimization

Unimodal Function: Definition

A *unimodal function* is one that has only one valley (minimum) in a given interval.

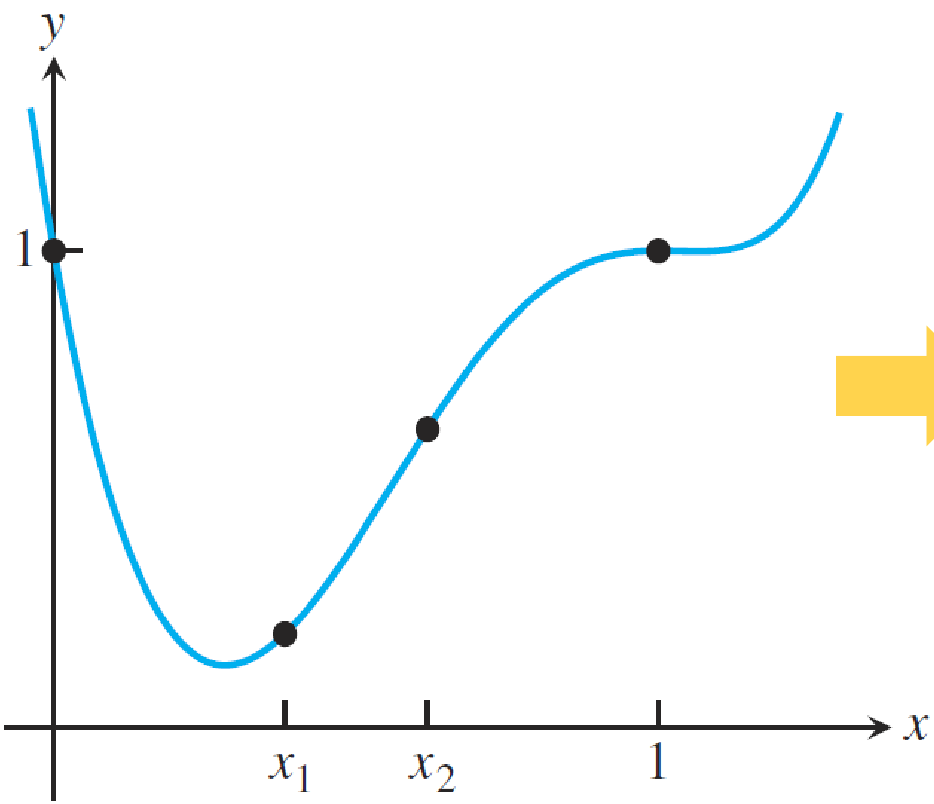
There is unique $x^* \in [a, b]$ such that $f(x^*)$ is minimum of f on [a, b], and f is strictly decreasing for $x \le x^*$, strictly increasing for $x^* \le x$.

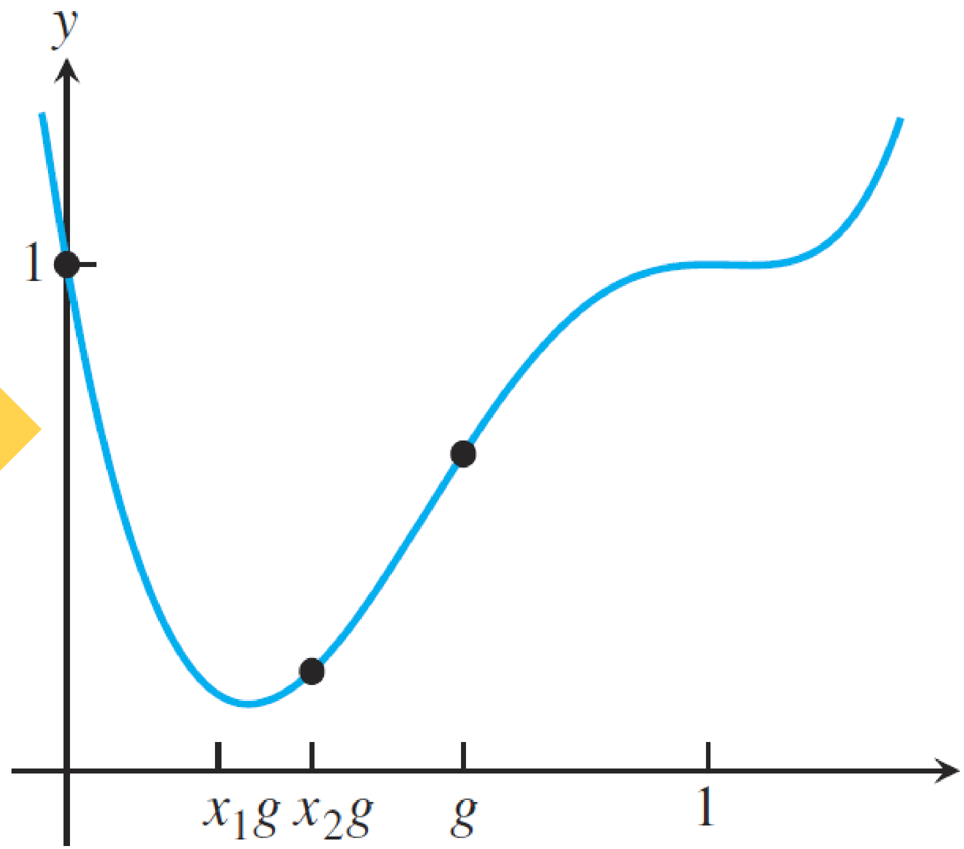


One-Dimensional Optimization

Golden Section Search: Geometric View

Given *f* unimodal with minimum in [0,1]

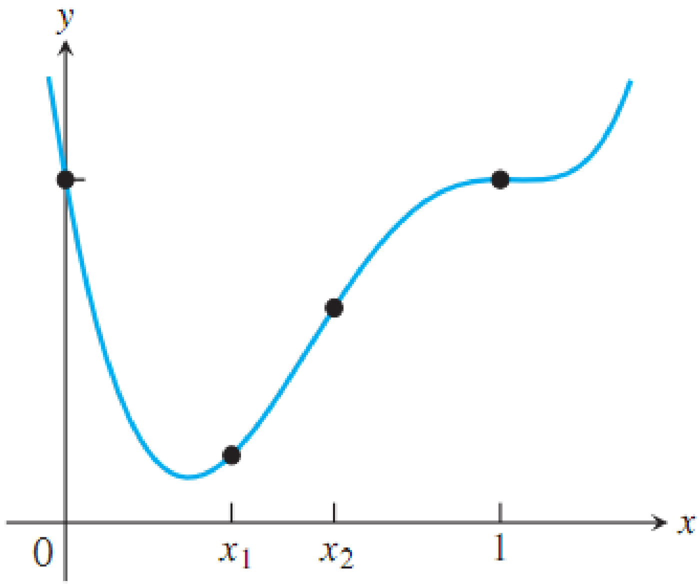


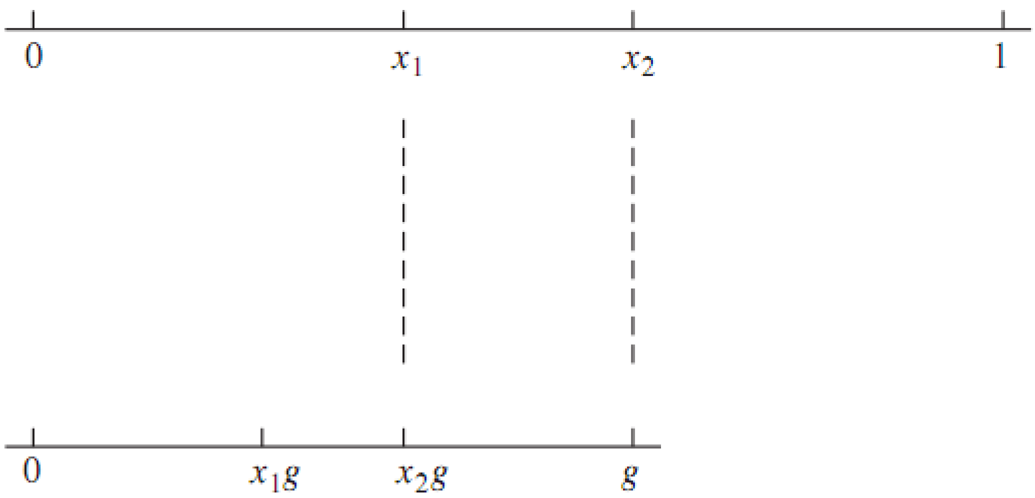


One-Dimensional Optimization

Golden Section Search: Basic Idea

Given f unimodal with minimum in [0,1]





(a) Make them symmetric with respect to the interval $x_1 = 1 - x_2$

(b) No matter which choice is made for the new interval, both x_1 and x_2 are used in the next step $x_1 = x_2^2$

$$x_2^2 + x_2 - 1 = 0 \Rightarrow x_2 = g = (\sqrt{5} - 1)/2$$

One-Dimensional Optimization

Golden Section Search: Pseudo-code

Given f unimodal with minimum in $[a, b]$

for $i = 1, 2, 3, \ldots$
  $g = (\sqrt{5} - 1)/2$
  if $f(a + (1 - g)(b - a)) < f(a + g(b - a))$
    $b = a + g(b - a)$
  else
    $a = a + (1 - g)(b - a)$
  end
end

The final interval $[a,b]$ contains a minimum.

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

One-Dimensional Optimization

Golden Section Search: Pseudo-code

Given $f(x)$ unimodal with minimum in $[a,b]$

g = (sqrt(5)-1)/2;

$x_1 = a + (1 - g)(b - a)$; $f_1 = f(x_1)$;

$x_2 = a + g(b - a)$; $f_2 = f(x_2)$;

while (b - a) > TOL

    if $(f_1 > f_2)$

        $a = x_1$; $x_1 = x_2$; $f_1 = f_2$;

        $x_2 = a + g(b - a)$; $f_2 = f(x_2)$;

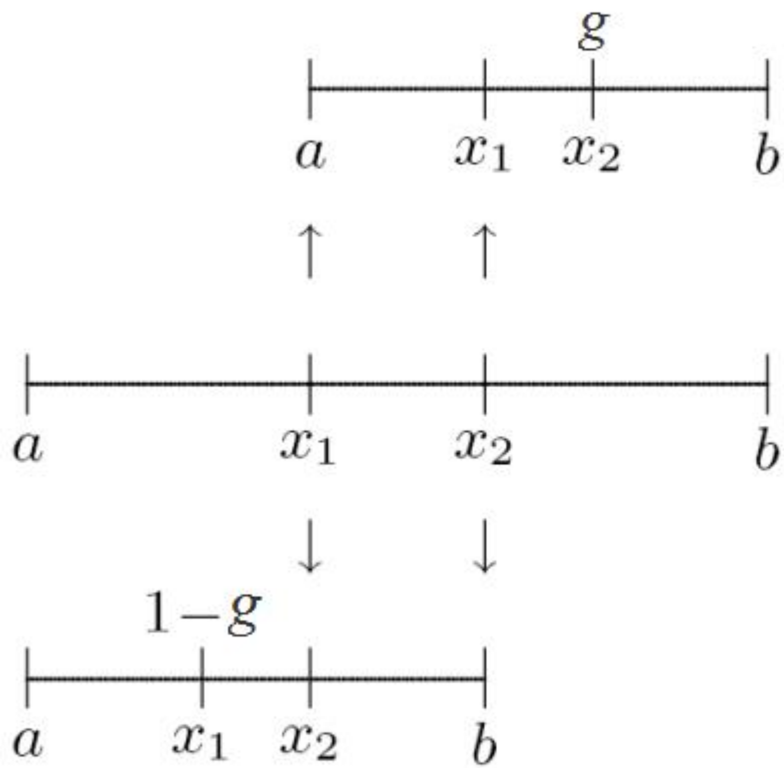
    else

        $b = x_2$; $x_2 = x_1$; $f_2 = f_1$;

        $x_1 = a + (1 - g)(b - a)$; $f_1 = f(x_1)$;

    end

end



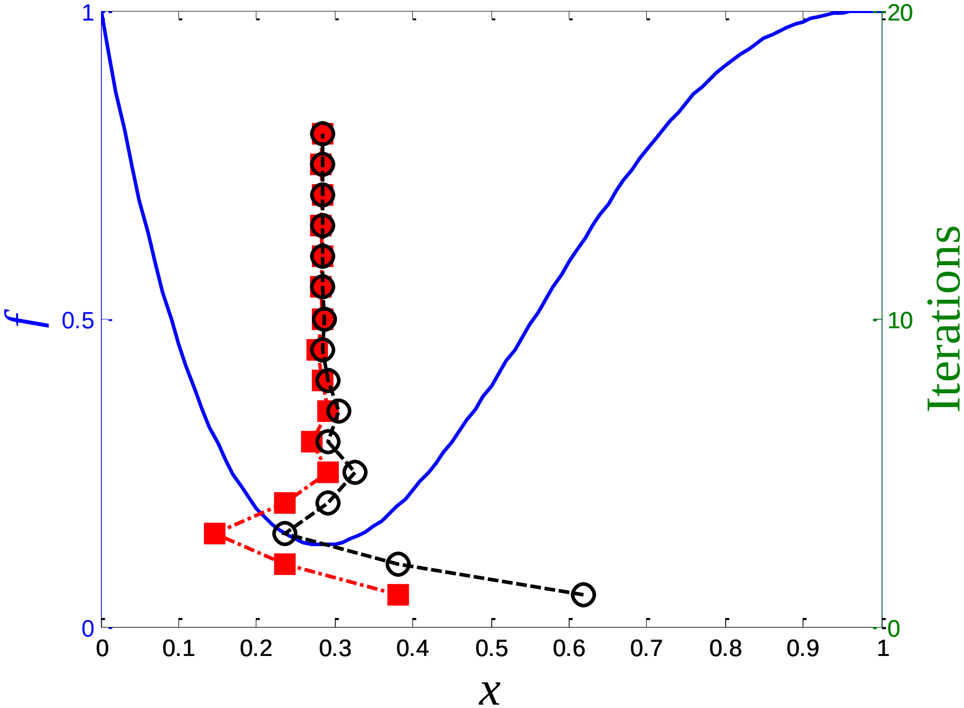
One-Dimensional Optimization

Golden Section Search: Example 1

Find the minimum of

$$f(x) = x^6 - 11x^3 + 17x^2 - 7x + 1$$

on the interval [0,1].



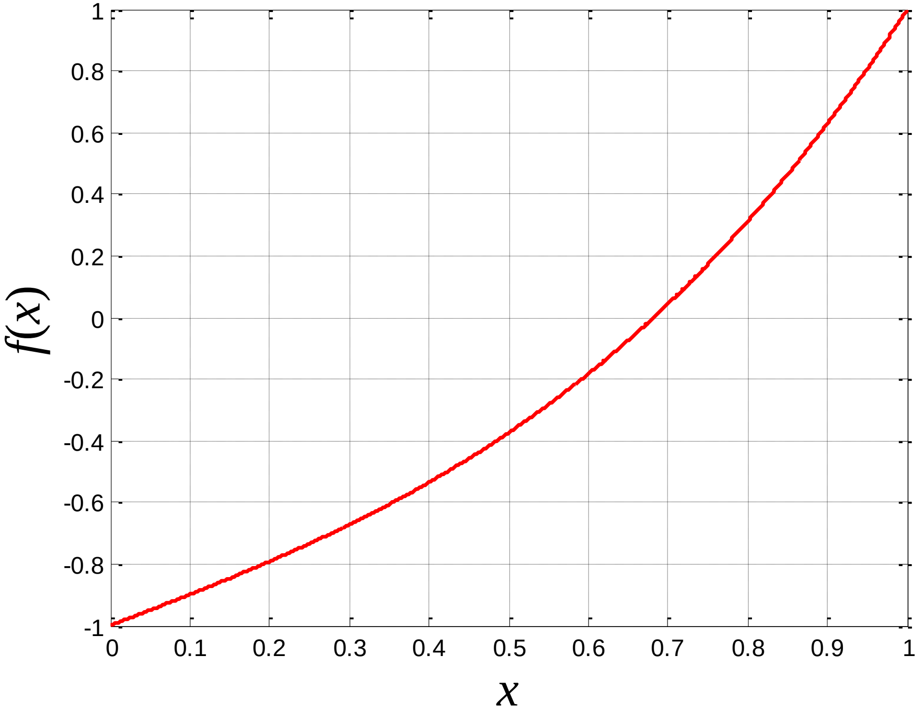
step	a	x_1	x_2	b
0	0.0000	0.3820	0.6180	1.0000
1	0.0000	0.2361	0.3820	0.6180
2	0.0000	0.1459	0.2361	0.3820
3	0.1459	0.2361	0.2918	0.3820
4	0.2361	0.2918	0.3262	0.3820
5	0.2361	0.2705	0.2918	0.3262
6	0.2705	0.2918	0.3050	0.3262
7	0.2705	0.2837	0.2918	0.3050
8	0.2705	0.2786	0.2837	0.2918
9	0.2786	0.2837	0.2868	0.2918
10	0.2786	0.2817	0.2837	0.2868
11	0.2817	0.2837	0.2849	0.2868
12	0.2817	0.2829	0.2837	0.2849
13	0.2829	0.2837	0.2841	0.2849
14	0.2829	0.2834	0.2837	0.2841
15	0.2834	0.2837	0.2838	0.2841

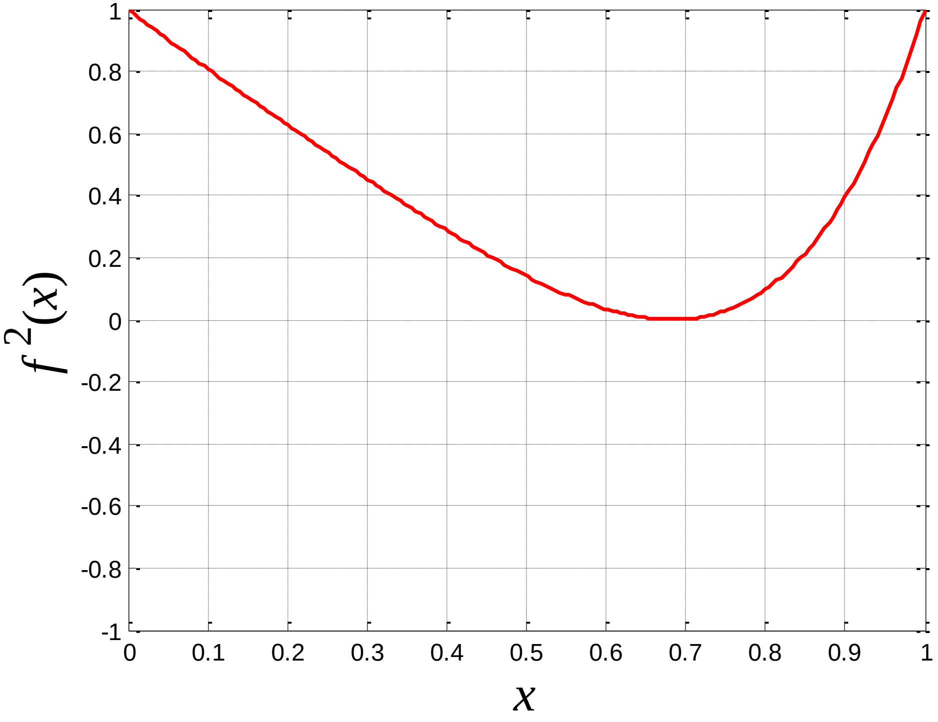
One-Dimensional Optimization

Golden Section Search: Example 2

Find the root of (Example 1 of Lecture 3 Revisited)

$$f(x) = x^3 + x - 1 = 0,\ x \in [0,1]$$





One-Dimensional Optimization

Golden Section Search: Example 2

Find the root of (Example 1 of Lecture 3 Revisited)

$$f(x) = x^3 + x - 1 = 0,\ x \in [0,1]$$

Exact solution x* = 0.682327803828019

Bisection Method: **k = 14;** **xc =** **0.6823425**	**Golden Section:** **k = 14;** **x_gss =** **0.6824858**

Optimization

- One-Dimensional Minimization
  - Golden Section Search
  - Newton's Method
  - Inaccurate Line Search
- Unconstrained Optimization
  - Steepest Descent Method
  - Newton's Method
  - Quasi-Newton Method
- Linear Programming
  - Geometry of Linear Programming
  - The Simplex Method

One-Dimensional Optimization

Newton's Method: Review

An Equation: $f(x) = 0$

Newton's Method: $x_{i+1} = x_i - \dfrac{f(x_i)}{f'(x_i)}$

Taylor Expansion:

$$f(x) \approx f(x_i) + f'(x_i)(x - x_i) = 0$$

One-Dimensional Optimization

Newton's Method for Optimization

$$\min_{x \in \mathbb{R}} \quad f(x)$$

First-Order Necessary Condition

$$f'(x) = 0$$

Newton's Method

$$\boxed{x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}}$$

One-Dimensional Optimization

Newton's Method for Optimization: Pseudo-code

Step 1. Given $x_0 \in$ R, $\varepsilon > 0$, $k := 0$;

Step 2. Caculate $f'(x_k)$, $f''(x_k)$;

Step 3. If $|f'(x_k)| < \varepsilon$, stop;

Solve

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Step 4. $k := k + 1$, goto Step 2.

One-Dimensional Optimization

- **Newton's Method: Example 3**

$$\min_{x\in\mathbb{R}} \quad f(x) = 0.65 - \frac{0.75}{1+x^2} - 0.65x\tan^{-1}\frac{1}{x}$$

$$f'(x) = \frac{1.5x}{\left(1+x^2\right)^2} + \frac{0.65x}{1+x^2} - 0.65\tan^{-1}\frac{1}{x}$$

$$f''(x) = \frac{2.8-3.2x^2}{\left(1+x^2\right)^3}$$

Use $x_0 = 0.1$, $\varepsilon = 0.01$

```
xList =
   0.100000000000000
   0.377240355518724
   0.465119791648128
   0.480408724516480
```

One-Dimensional Optimization

- **Quasi-Newton's Method for Optimization**

$$\min_{x \in \mathbb{R}} \quad f(x)$$

Newton's Method

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Finite difference formulas

One-Dimensional Optimization

Finite Difference Formulas: Review

Three-point centered-difference formula

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6} f'''(c)$$

where $x - h < c < x + h$.

Three-point centered-difference formula for second derivative

$$f''(x) = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} - \frac{h^2}{12} f^{(\mathrm{iv})}(c)$$

for some c between $x - h$ and $x + h$.

One-Dimensional Optimization

Quasi-Newton's Method for Optimization

$$\min_{x\in\mathbb{R}} \quad f(x)$$

Newton's Method

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Finite difference formulas

$$x_{k+1} = x_k - \frac{\Delta x[f(x_k+\Delta x) - f(x_k-\Delta x)]}{2[f(x_k+\Delta x) - 2f(x_k) + f(x_k-\Delta x)]}$$

One-Dimensional Optimization

Quasi-Newton's Method: Example 4

$$\min_{x\in\mathbb{R}} \quad f(x) = 0.65 - \frac{0.75}{1+x^2} - 0.65x\tan^{-1}\frac{1}{x}$$

Newton's Method:

Use $x_0 = 0.1$, $\varepsilon = 0.01$

```
xList =
   0.100000000000000
   0.377240355518724
   0.465119791648128
   0.480408724516480
```

Quasi-Newton's Method:

Use $x_0 = 0.1$, $\varepsilon = 0.01$,

$\Delta x = 0.01$

```
xList =
   0.100000000000000
   0.377271453664973
   0.465177230088857
   0.480473052168382
```

One-Dimensional Optimization

Quasi-Newton's Method: Example 5

Find the minimum of

$$f(x) = x^6 - 11x^3 + 17x^2 - 7x + 1$$

with $x_0 = 0.1$, $\Delta x = 0.01$.

```
xStar =
   0.283716504125462
xList =
   0.100000000000000
   0.243452676259928
   0.280842873920272
   0.283699846068886
   0.283716504125462
```

Golden Section Search

step	a	x_1	x_2	b
0	0.0000	0.3820	0.6180	1.0000
1	0.0000	0.2361	0.3820	0.6180
2	0.0000	0.1459	0.2361	0.3820
3	0.1459	0.2361	0.2918	0.3820
4	0.2361	0.2918	0.3262	0.3820
5	0.2361	0.2705	0.2918	0.3262
6	0.2705	0.2918	0.3050	0.3262
7	0.2705	0.2837	0.2918	0.3050
8	0.2705	0.2786	0.2837	0.2918
9	0.2786	0.2837	0.2868	0.2918
10	0.2786	0.2817	0.2837	0.2868
11	0.2817	0.2837	0.2849	0.2868
12	0.2817	0.2829	0.2837	0.2849
13	0.2829	0.2837	0.2841	0.2849
14	0.2829	0.2834	0.2837	0.2841
15	0.2834	0.2837	0.2838	0.2841

Optimization

- **One-Dimensional Minimization**
  - **Golden Section Search**
  - **Newton's Method**
  - **Inaccurate Line Search**
- **Unconstrained Optimization**
  - **Steepest Descent Method**
  - **Newton's Method**
  - **Quasi-Newton Method**
- **Linear Programming**
  - **Geometry of Linear Programming**
  - **The Simplex Method**

One-Dimensional Optimization

Inaccurate Line Search: Motivation

$$\min_{\alpha \in \mathbb{R}} \quad f(\mathbf{x}_k + \alpha \mathbf{d}_k)$$

where $\mathbf{x}_k \in \mathbb{R}^n, \mathbf{d}_k \in \mathbb{R}^n, \ f(\mathbf{x}_k) \in \mathbb{R}$

$$\min_{\mathbf{x} \in \mathbb{R}^n} \quad f(\mathbf{x})$$

Beginning at $\mathbf{x}_0$, **to generate a sequence of iterates**

$\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{k-1}, \mathbf{x}_k, \cdots, \mathbf{x}^*$ **, such that**

$$f(\mathbf{x}_0) > f(\mathbf{x}_1) > \cdots > f(\mathbf{x}_k) > \cdots > f(\mathbf{x}^*)$$

One-Dimensional Optimization

Inaccurate Line Search: Motivation

$$\min_{\alpha \in \mathbb{R}} \quad f(\mathbf{x}_k + \alpha \mathbf{d}_k)$$

where $\mathbf{x}_k \in \mathbb{R}^n$, $\mathbf{d}_k \in \mathbb{R}^n$, $f(\mathbf{x}_k) \in \mathbb{R}$

***k* = *0*, *x*$_0$**

for ***k*** **= 1, 2, 3, …**

(1) determine d$_k$ at x$_k$

(2) calculate the step α_k from $\min_{\alpha \in \mathbb{R}} \quad f(\mathbf{x}_k + \alpha \mathbf{d}_k)$

(3) $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$

(4) check the termination criterion

end

The exact line search is expensive!

One-Dimensional Optimization

Inaccurate Line Search

Define the function

$$\phi(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{d}_k)$$

The essential idea is that the rule should first guarantee that the selected α is not too large, and next it should not be too small.

$\alpha = 0$: $\phi(0) = f(x_k),\ \phi'(0) = \nabla f(x_k)^T d_k < \mathbf{0}$

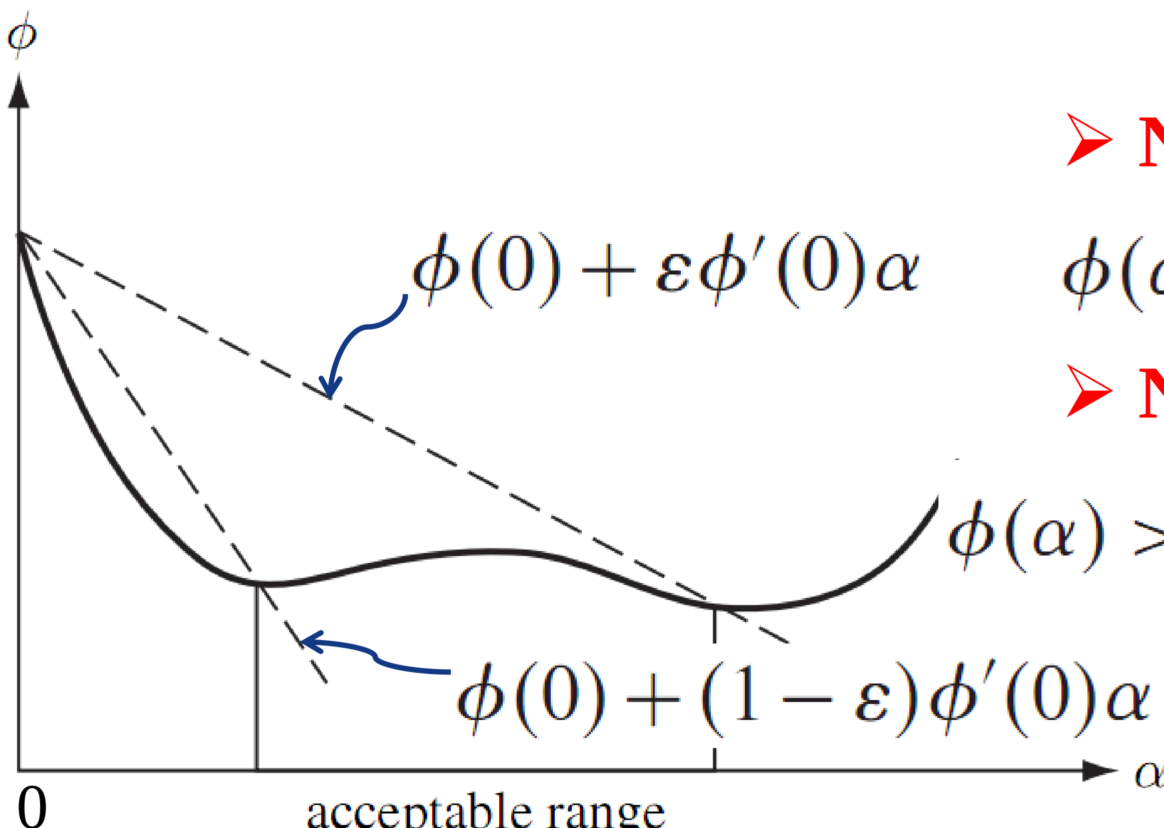
$\alpha = 1$: $\phi(1) = f(x_k + d_k)$

One-Dimensional Optimization

- **Inaccurate Line Search: Armijo-Goldstein Rule**

Define the function

$$\phi(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{d}_k)$$



- **Not too large**

$$\phi(\alpha) \leqslant \phi(0) + \varepsilon\phi'(0)\alpha$$

- **Not too small**

$$\phi(\alpha) > \phi(0) + (1-\varepsilon)\phi'(0)\alpha$$

$$0 < \varepsilon < (1/2)$$

One-Dimensional Optimization

- **Inaccurate Line Search: backtracking approach**

Backtracking Line Search

Step 1. **Given $0 < \varepsilon < 0.5$, $0 < L < U < 1$, set $\alpha = 1$.**

Step 2. **Test**

$$f(\mathbf{x}_k + \alpha \mathbf{d}_k) \le f(\mathbf{x}_k) + \varepsilon \alpha \nabla f(\mathbf{x}_k)^T \mathbf{d}_k$$

If it is satisfied, $\alpha_k = \alpha$ and $x_{k+1} = x_k + \alpha_k d_k$.

otherwise, go to Step 3.

Step 3. **Set $\alpha = w\alpha$, $w \in [L,U]$, go to Step 2.**

Generally, $\varepsilon \in [10^{-4}, 0.2]$, $w = 0.5$, $L = 0.15$, $U = 0.85$.

One-Dimensional Optimization

- **Inaccurate Line Search: Quadratic interpolation**

In *Step 3* of Backtracking Line Search

$$\phi(\alpha) = f(x_k + \alpha d_k)$$

$\alpha = 0$: $\phi(0) = f(x_k),\ \phi'(0) = \nabla f(x_k)^T d_k$

$\alpha = 1$: $\phi(1) = f(x_k + d_k)$

$\phi(\alpha)$ *is approximated by*

$$m(\alpha) = [\phi(1) - \phi(0) - \phi'(0)]\alpha^2 + \phi'(0)\alpha + \phi(0)$$

$$\Downarrow \; m'(\alpha) = 0$$

$$\hat{\alpha} = -\frac{\phi'(0)}{2[\phi(1) - \phi(0) - \phi'(0)]}$$

One-Dimensional Optimization

- **Inaccurate Line Search: Quadratic interpolation**

Suppose that the initial guess α_0 is given, on $[0, \alpha_0]$

$$\phi_q(\alpha) = \left(\frac{\phi(\alpha_0) - \phi(0) - \alpha_0 \phi'(0)}{\alpha_0^2} \right) \alpha^2 + \phi'(0)\alpha + \phi(0)$$

satisfying the interpolation conditions

$$\phi_q(0) = \phi(0), \phi_q'(0) = \phi'(0), \text{and } \phi_q(\alpha_0) = \phi(\alpha_0)$$

$$\Downarrow$$

$$\alpha_1 = -\frac{\phi'(0)\alpha_0^2}{2\left[\phi(\alpha_0) - \phi(0) - \phi'(0)\alpha_0\right]}$$

Optimization

- One-Dimensional Minimization
  - Golden Section Search
  - Newton's Method
  - Inaccurate Line Search
- Unconstrained Optimization
  - Steepest Descent Method
  - Newton's Method
  - Quasi-Newton Method
- Linear Programming
  - Geometry of Linear Programming
  - The Simplex Method

Unconstrained Optimization

Steepest Descent Method

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad f(\mathbf{x})$$

where $f(\mathbf{x}) \in \mathbb{R}$ has continuous first partial derivatives

The gradient $\nabla f(\mathbf{x})$ (or g) is defined as a n-dimensional *column* vector.

$$\underset{n\times 1}{\nabla f} = \begin{bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \\ \vdots \\ \partial f/\partial x_n \end{bmatrix}$$

Unconstrained Optimization

Steepest Descent Method: Basic Idea

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad f(\mathbf{x})$$

Suppose that $f(x)$ is continuously differentiable near x_k and $g(x_k) \neq 0$.

From the Taylor expansion

$$f(x) = f(x_k) + (x - x_k)^T g_k + o(\|x - x_k\|)$$

Let $x - x_k = \alpha d_k$, the direction d_k satisfying $d_k^T g_k < 0$ is called a **descent direction**, such that

$$f(x) < f(x_k)$$

Unconstrained Optimization

Steepest Descent Method: Basic Idea

From the Taylor expansion

$$f(x) = f(x_k) + (x - x_k)^T g_k + o(\|x - x_k\|)$$

Let $x - x_k = \alpha d_k$ **, the direction** d_k **satisfying** $d_k^T g_k < 0$ **is called a descent direction, such that**

$$f(x) < f(x_k)$$

$$|d_k^T g_k| \le \|d_k\| \|g_k\|$$

$d_k^T g_k$ is the smallest $\Longleftrightarrow$ $d_k = -g_k$

the steepest descent direction

Unconstrained Optimization

Steepest Descent Method: Pseudo-code

Step 0. Let $0 < \varepsilon \ll 1$ be the termination tolerance.
Given an initial point $x_0 \in R^n$. Set $k = 0$.

Step 1. If $\|g_k\| \leq \varepsilon$, stop ; otherwise let $d_k = -g_k$.

Step 2. Find the steplength factor α_k, such that

$$f(x_k + \alpha_k d_k) = \min_{\alpha \geq 0} f(x_k + \alpha d_k);$$

Step 3. Compute $x_{k+1} = x_k + \alpha_k d_k$.

Step 4. $k := k + 1$, return to Step 1.

Unconstrained Optimization

Steepest Descent Method: Example 6

Locate the minimum of the function

$$f(x, y) = 5x^4 + 4x^2 y - xy^3 + 4y^4 - x$$

The gradient is

$$\nabla f = (20x^3 + 8xy - y^3 - 1, 4x^2 - 3xy^2 + 16y^3)^T$$

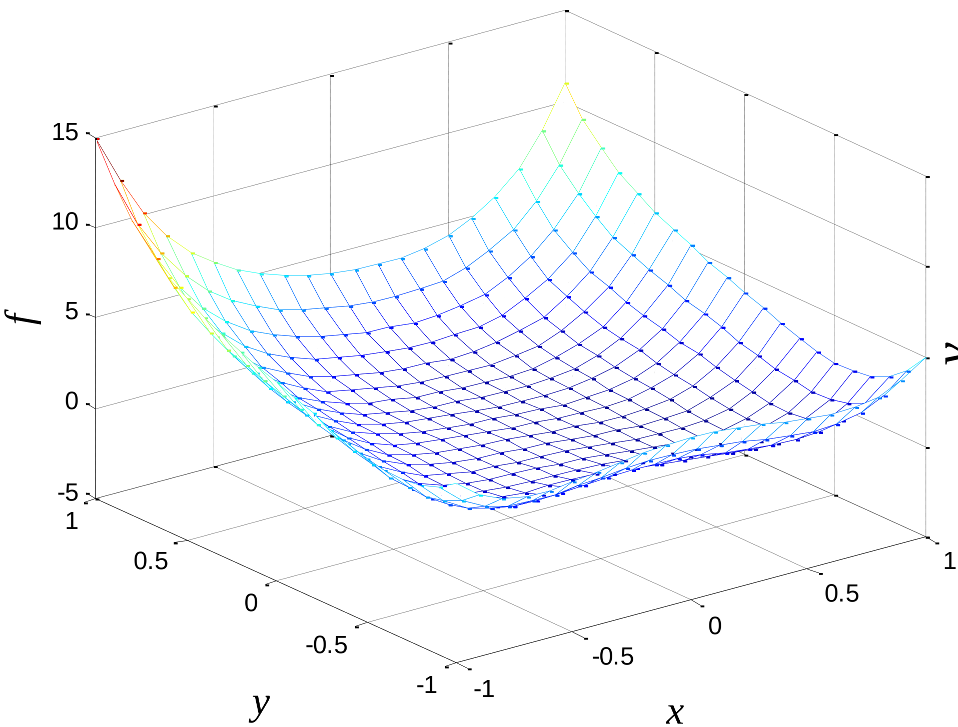
step	x	y	$f(x, y)$
0	1.00000000000000	−1.00000000000000	11.00000000000000
5	0.40314579518113	−0.27992088271756	−0.41964888830651
10	0.49196895085112	−0.36216404374206	−0.45750680523754
15	0.49228284433776	−0.36426635686172	−0.45752161934016
20	0.49230786417532	−0.36428539567277	−0.45752162263389
25	0.49230778262142	−0.36428556578033	−0.45752162263407

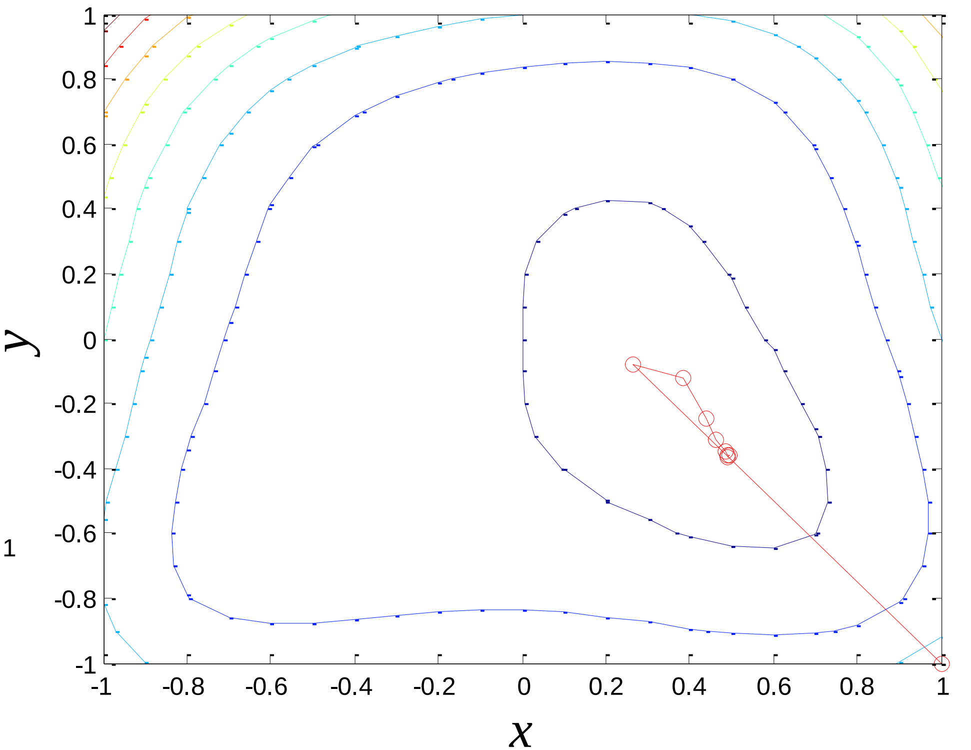
Unconstrained Optimization

Steepest Descent Method: Example 6

Locate the minimum of the function

$$f(x, y) = 5x^4 + 4x^2 y - x y^3 + 4y^4 - x$$





Unconstrained Optimization

Newton's Method: Basic Idea

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad f(\mathbf{x})$$

where $f(\mathbf{x}) \in \mathbb{R}$ has continuous second partial derivatives.

Approximate f by the truncated Taylor series:

$$f(\mathbf{x}) \simeq f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T(\mathbf{x}-\mathbf{x}_k) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_k)^T \underline{\mathbf{F}(\mathbf{x}_k)}(\mathbf{x}-\mathbf{x}_k)$$

the Hessian matrix

The right-hand side is minimized at

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{F}(\mathbf{x}_k)]^{-1}\nabla f(\mathbf{x}_k)$$

Unconstrained Optimization

Newton's Method: Damping Parameter

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad f(\mathbf{x})$$

where $f(\mathbf{x}) \in \mathbb{R}$ has continuous second partial derivatives.

The right-hand side is minimized at

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{F}(\mathbf{x}_k)]^{-1}\nabla f(\mathbf{x}_k)$$

Step-Length Parameter α

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k[\mathbf{F}(\mathbf{x}_k)]^{-1}\nabla f(\mathbf{x}_k)$$

Unconstrained Optimization

Newton's Method: Pseudo-code

Step 1. *Initial step: given* $x_0 \in R^n, \epsilon > 0$, *set* $k := 0$.

Step 2. *Compute* g_k. *If* $||g_k|| \le \epsilon$, *stop and output* x_k; *otherwise go to Step 3.*

Step 3. *Solve* $F_k d = -g_k$ *for* d_k.

Step 4. *Line search step: find* α_k *such that*

$$f(x_k + \alpha_k d_k) = \min_{\alpha \ge 0} f(x_k + \alpha d_k).$$

Step 5. *Set* $x_{k+1} = x_k + \alpha_k d_k$, $k := k + 1$, *go to Step 2.*

Unconstrained Optimization

Newton's Method: Example 7

Locate the minimum of the function

$$f(x, y) = 5x^4 + 4x^2y - xy^3 + 4y^4 - x$$

The gradient is

$$\nabla f = (20x^3 + 8xy - y^3 - 1, 4x^2 - 3xy^2 + 16y^3)^T$$

The Hessian is

$$H_f(x, y) = \begin{bmatrix} 60x^2 + 8y & 8x - 3y^2 \\ 8x - 3y^2 & -6xy + 48y^2 \end{bmatrix}$$

Unconstrained Optimization

Newton's Method: Example 7

Locate the minimum of the function

$$f(x, y) = 5x^4 + 4x^2y - xy^3 + 4y^4 - x$$

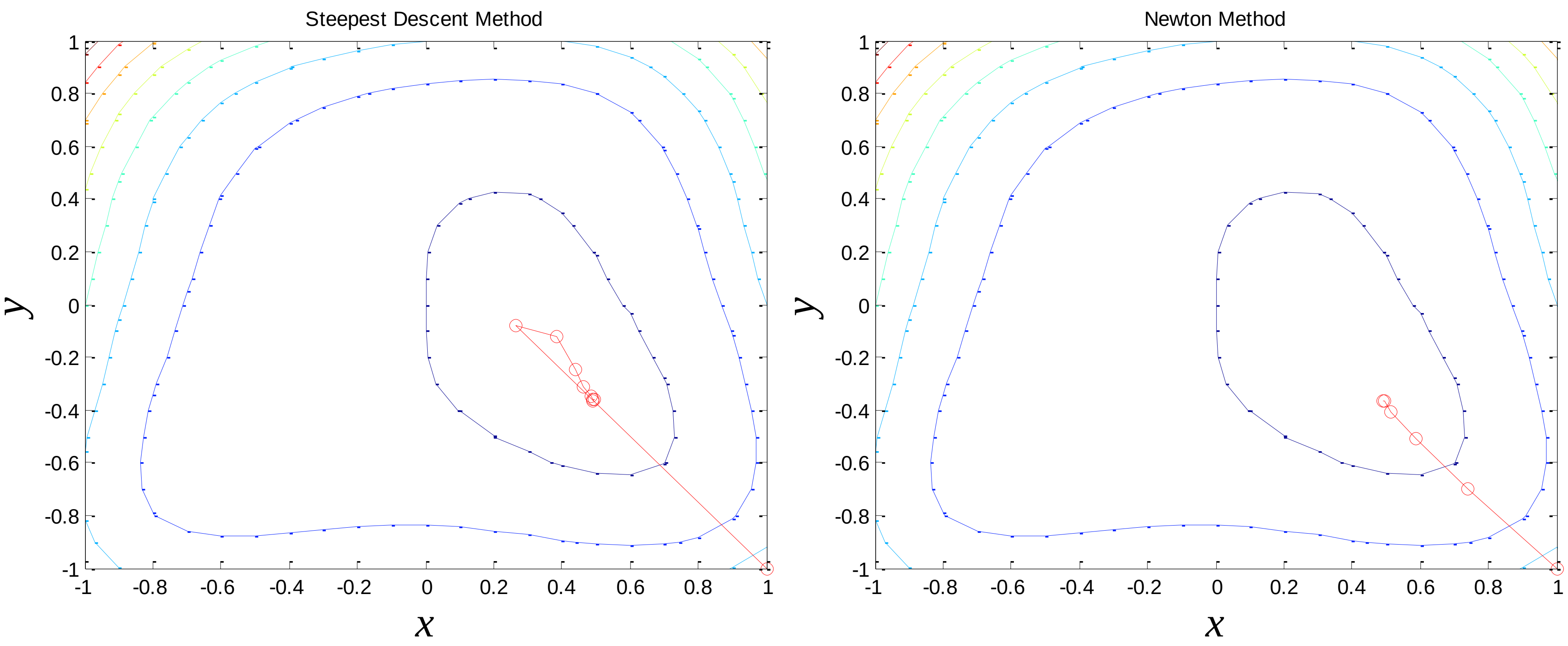
step	x	y	$f(x, y)$
0	1.00000000000000	1.00000000000000	11.0000000000000
1	0.64429530201342	0.63758389261745	1.77001867827422
2	0.43064034542956	0.39233298702231	0.10112006537534
3	0.33877971433352	0.19857714160717	−0.17818585977225
4	0.50009733696780	−0.44771929519763	−0.42964065053918
5	0.49737350571430	−0.37972645728644	−0.45673719664708
6	0.49255000651877	−0.36497753746514	−0.45752009007757
7	0.49230831759106	−0.36428704569173	−0.45752162262701
8	0.49230778672681	−0.36428555993321	−0.45752162263407
9	0.49230778672434	−0.36428555992634	−0.45752162263407
10	0.49230778672434	−0.36428555992634	−0.45752162263407

Unconstrained Optimization

Newton's Method: Example 7

Locate the minimum of the function

$$f(x, y) = 5x^4 + 4x^2 y - x y^3 + 4y^4 - x$$



Unconstrained Optimization

- **Quasi-Newton Method**

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad f(\mathbf{x})$$

***f*(x) is twice continuously differentiable**

$$f(x) \approx f(x_{k+1}) + g_{k+1}^T(x - x_{k+1}) + \frac{1}{2}(x - x_{k+1})^T \underline{G_{k+1}}(x - x_{k+1})$$

the Hessian matrix

min

$$g(x) \approx g_{k+1} + G_{k+1}(x - x_{k+1})$$

$$x = x_k, s_k = x_{k+1} - x_k \text{ and } y_k = g_{k+1} - g_k$$

$$G_{k+1}^{-1} y_k \approx s_k$$

Inverse Hessian approximation?

$$H_{k+1} y_k = s_k$$

Unconstrained Optimization

Quasi-Newton Method

$$H_{k+1} y_k = s_k \qquad H_{k+1} = H_k + E_k$$

Symmetric rank-two update

$$H_{k+1} = H_k + auu^T + bvv^T$$

$$H_k y_k + auu^T y_k + bvv^T y_k = s_k$$

$$u = s_k, \; v = H_k y_k$$

$$a = 1/u^T y_k = 1/s_k^T y_k, \; b = -1/v^T y_k = -1/y_k^T H_k y_k$$

Unconstrained Optimization

Quasi-Newton Method: Pseudo-code

Given $x_0 \in R^n$ *an initial point,* $H_0 \in R^{n\times n}$ *a symmetric and positive definite matrix,* $\epsilon > 0$ *a termination scalar,* $k := 0$.

For $k = 0, 1, \cdots,$

1. *If* $||g_k|| \le \epsilon$, *stop.*
2. *Compute* $d_k = -H_k g_k$.
3. *Compute the step size* α_k.
4. *Set* $s_k = \alpha_k d_k, x_{k+1} = x_k + s_k, y_k = g_{k+1} - g_k$, *and*

$$H_{k+1} = H_k + \frac{s_k s_k^T}{s_k^T y_k} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k}.$$

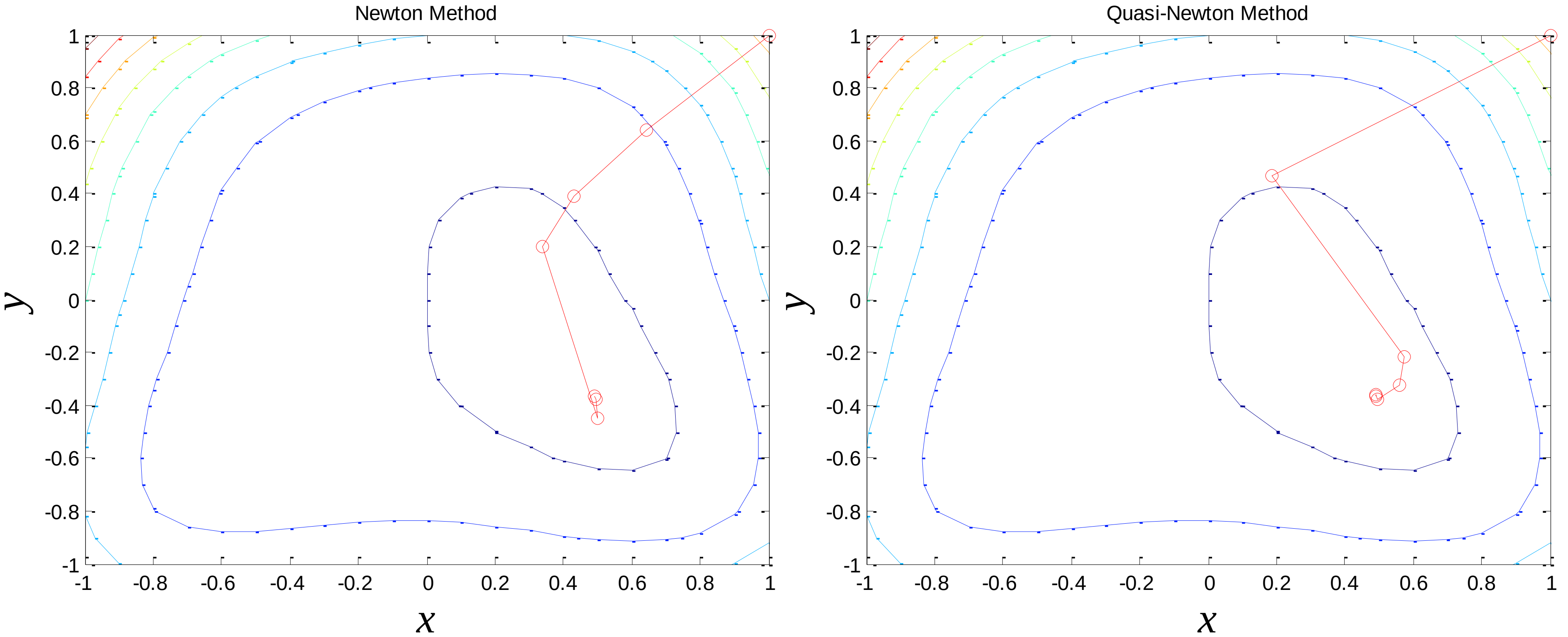
5. $k := k+1$, *go to Step 1.*

Unconstrained Optimization

Quasi-Newton Method: Example 8

Locate the minimum of the function

$$f(x, y) = 5x^4 + 4x^2 y - x y^3 + 4y^4 - x$$



上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Optimization

- One-Dimensional Minimization
  - Golden Section Search
  - Newton's Method
  - Inaccurate Line Search
- Unconstrained Optimization
  - Steepest Descent Method
  - Newton's Method
  - Quasi-Newton Method
- Linear Programming
  - ✓ Geometry of Linear Programming
  - ✓ The Simplex Method

Linear Programming

- **Standard Form**

A linear program (LP) is an optimization problem in which the objective function is linear in the unknowns and the constraints consist of linear equalities and linear inequalities.

$$
\begin{array}{ll}
\text{minimize} & c_1x_1 + c_2x_2 + \ldots + c_nx_n \\
\text{subject to} & a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\
 & a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\
 & \cdot \qquad\qquad\qquad\qquad\qquad\qquad \cdot \\
 & \cdot \qquad\qquad\qquad\qquad\qquad\qquad \cdot \\
 & \cdot \qquad\qquad\qquad\qquad\qquad\qquad \cdot \\
 & a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \\
\text{and} & x_1 \geqslant 0,\ x_2 \geqslant 0,\ \ldots,\ x_n \geqslant 0,
\end{array}
$$

Linear Programming

- **Standard Form: Compact Vector Notation**

$$\text{minimize} \quad \mathbf{c}^T\mathbf{x}$$

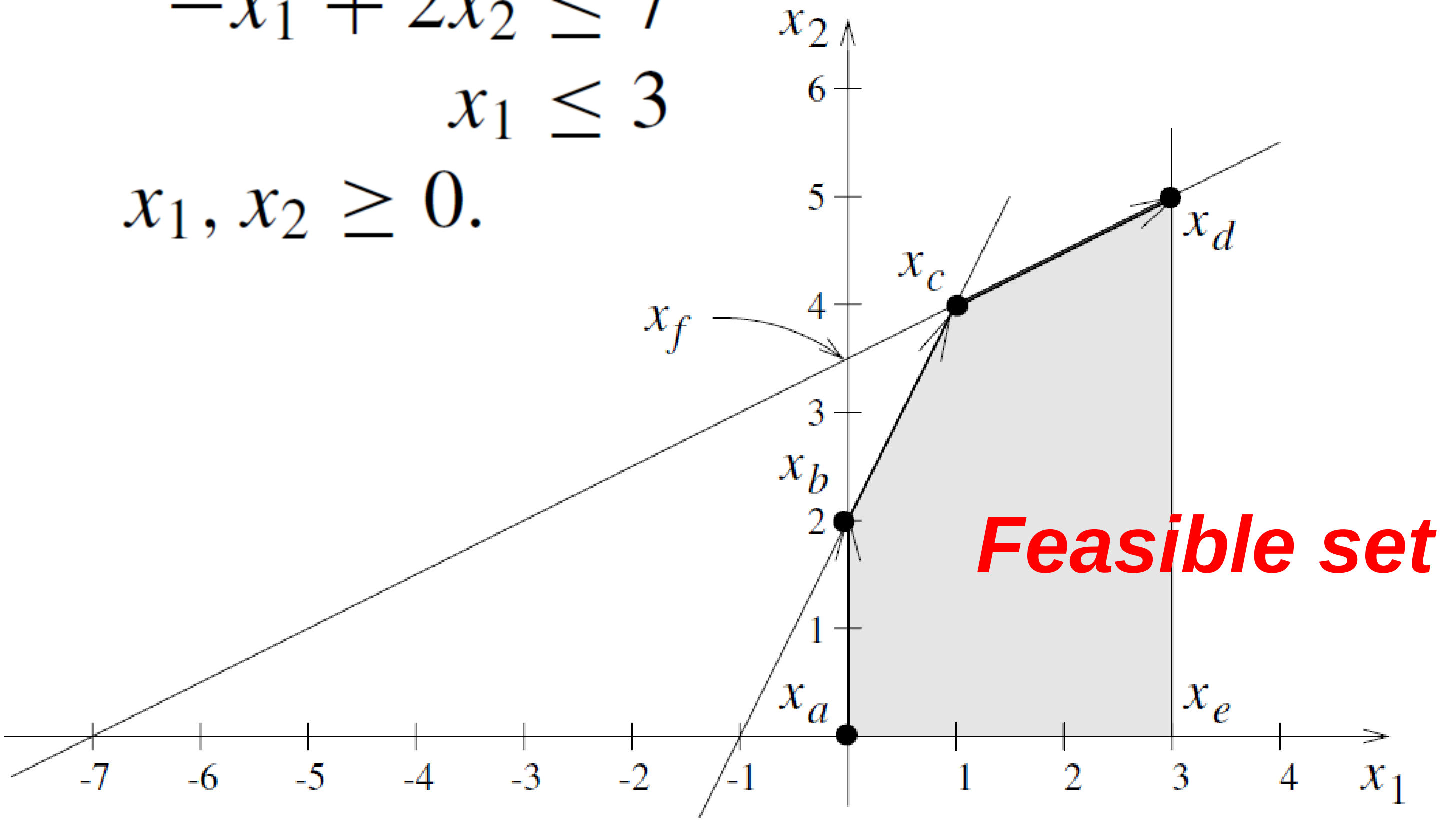
$$\text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \quad \text{and} \quad \mathbf{x} \geqslant \mathbf{0}$$

where x is an *n*-dimensional column vector, $\mathbf{c}^T$ is an *n*-dimensional row vector, A is an $m \times n$ matrix, and b is an *m*-dimensional column vector. The vector inequality $\mathbf{x} \geq 0$ means that each component of x is nonnegative.

Linear Programming

Linear Programming: Example 9

$$\begin{aligned} \text{minimize} \quad & z = -x_1 - 2x_2 \\ \text{subject to} \quad & -2x_1 + x_2 \le 2 \\ & -x_1 + 2x_2 \le 7 \\ & x_1 \le 3 \\ & x_1, x_2 \ge 0. \end{aligned}$$



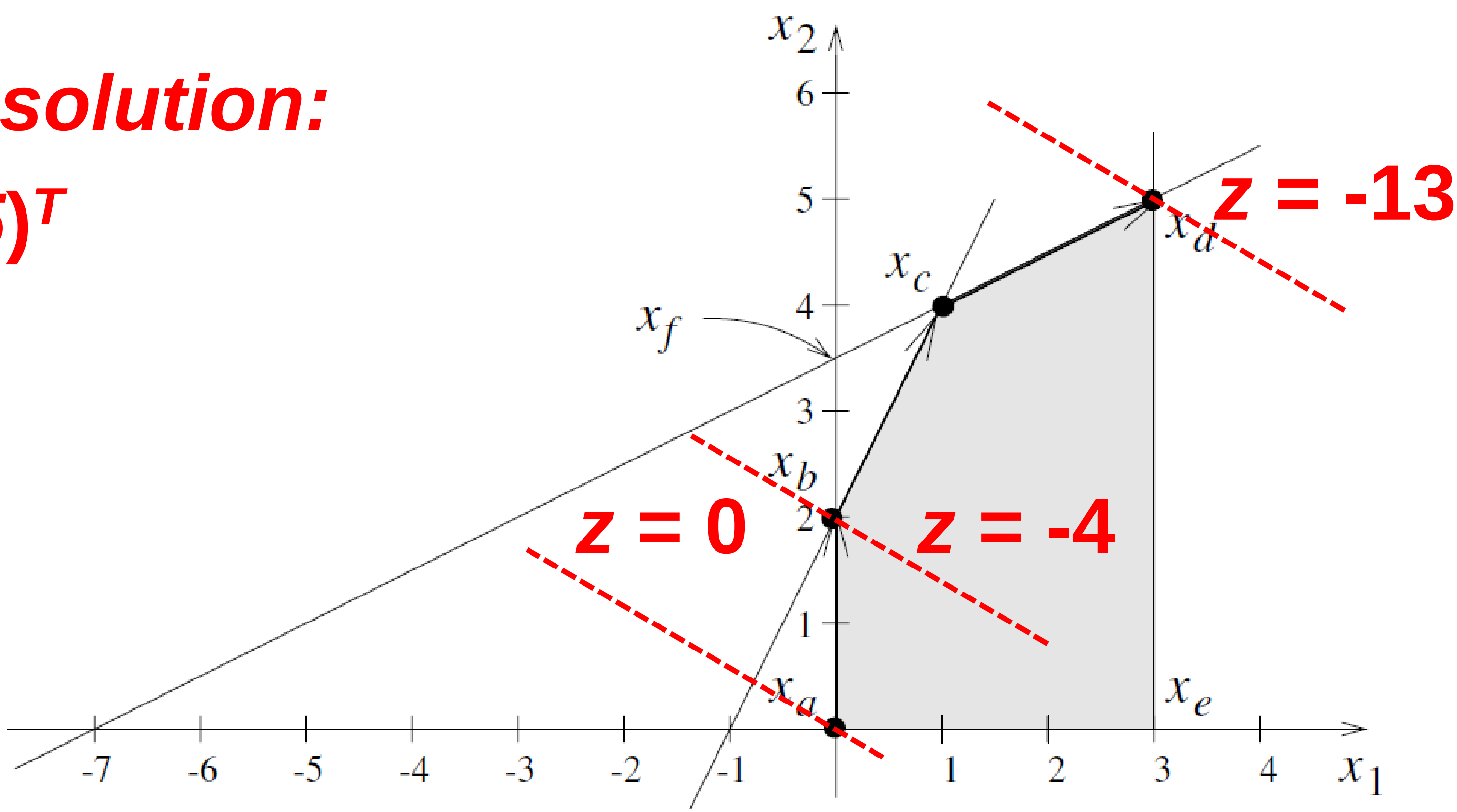
Linear Programming

Linear Programming: Example 9

Graphical solution: $z = -x_1 - 2x_2$

Optimal solution:

$x^* = (3, 5)^T$



Linear Programming

- **Linear Programming: Example 9**

In standard form

$$
\begin{array}{ll}
\text{minimize} & z = -x_1 - 2x_2 \\
\text{subject to} & -2x_1 + x_2 + x_3 = 2 \\
& -x_1 + 2x_2 + x_4 = 7 \\
& x_1 + x_5 = 3 \\
& x_1, x_2, x_3, x_4, x_5 \ge 0.
\end{array}
$$

Linear Programming

- **Linear Programming: Example 9**

$$\begin{aligned} \text{minimize} \quad & z = -x_1 - 2x_2 \\ \text{subject to} \quad & \boxed{\begin{aligned} -2x_1 + x_2 + x_3 &= 2 \\ -x_1 + 2x_2 + x_4 &= 7 \\ x_1 + x_5 &= 3 \\ x_1, x_2, x_3, x_4, x_5 &\geq 0. \end{aligned}} \end{aligned}$$

To find a basic feasible solution

$$x_B = (x_3, x_4, x_5)^T \text{ and } x_N = (x_1, x_2)^T.$$

$$(x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5)^T = (0 \quad 0 \quad 2 \quad 7 \quad 3)^T$$

Linear Programming

Linear Programming: Example 9

For the basic feasible solution [Ite #0]:

$$x_B = (x_3, x_4, x_5)^T \text{ and } x_N = (x_1, x_2)^T$$

$$(x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5)^T = (0 \quad 0 \quad 2 \quad 7 \quad 3)^T$$

the basic variables expressed in terms of the nonbasic variables:

$$\begin{aligned} x_3 &= 2 + 2x_1 - x_2 \\ x_4 &= 7 + x_1 - 2x_2 \\ x_5 &= 3 - x_1 . \end{aligned}$$

Linear Programming

- **Linear Programming: Example 9**

The basic variables expressed in terms of the nonbasic variables:

$$x_3 = 2 + 2x_1 - x_2$$
$$x_4 = 7 + x_1 - 2x_2$$
$$x_5 = 3 - x_1 .$$

To minimize the objective function

$$z = -x_1 - 2x_2$$

- to *increase* a nonbasic variable?

上海交通大学 Shanghai Jiao Tong University

Linear Programming

- **Linear Programming: Example 9**

The basic variables expressed in terms of the nonbasic variables:

$$x_3 = 2 + 2x_1 - x_2$$
$$x_4 = 7 + x_1 - 2x_2$$
$$x_5 = 3 - x_1 .$$

To minimize the objective function

$$z = -x_1 - 2x_2$$

-we choose to increase x_2 rather than x_1

Linear Programming

Linear Programming: Example 9

The basic variables expressed in terms of the nonbasic variables:

$$x_3 = 2 + 2x_1 - x_2$$
$$x_4 = 7 + x_1 - 2x_2$$
$$x_5 = 3 - x_1 .$$

$$\boxed{\begin{aligned} x_3 &= 0 \text{ when } x_2 = 2 \\ x_4 &= 0 \text{ when } x_2 = \tfrac{7}{2} \end{aligned}}$$

To minimize the objective function

$$z = -x_1 - 2x_2$$

-we choose to increase x_2 rather than x_1

-x_2 can only be increased to the value $x_2 = 2$

Linear Programming

Linear Programming: Example 9

The new basic feasible solution is [Ite #1]:

$$(\,x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5\,)^T = (\,0 \quad 2 \quad 0 \quad 3 \quad 3\,)^T$$

where $x_B = (x_2, x_4, x_5)^T$ and $x_N = (x_1, x_3)^T$

The linear program has the new form:

$$\text{minimize } z = -4 - 5x_1 + 2x_3$$

$$\begin{aligned} \text{subject to} \quad x_2 &= 2 + 2x_1 - x_3 \\ x_4 &= 3 - 3x_1 + 2x_3 \\ x_5 &= 3 - x_1 \end{aligned}$$

-we choose to increase x_1 as $x_1 = 1$

Linear Programming

Linear Programming: Example 9

The new basic feasible solution is [Ite #2]:

$$\left(x_1\ x_2\ x_3\ x_4\ x_5\right)^T = \left(1\ 4\ 0\ 0\ 2\right)^T$$

where $x_B = \left(x_1\ x_2\ x_5\right)^T,\ x_N = \left(x_3\ x_4\right)^T$

The linear program has the new form:

$$\begin{aligned} \min \quad & z = -9 - x_3 + x_4 \\ s.t. \quad & x_1 = 1 + \frac{2}{3}x_3 - \frac{1}{3}x_4 \\ & x_2 = 4 + \frac{1}{3}x_3 - \frac{2}{3}x_4 \\ & x_5 = 2 - \frac{2}{3}x_3 + \frac{1}{3}x_4 \end{aligned}$$

-we choose to increase x_3 as $x_3 = 3$

Linear Programming

Linear Programming: Example 9

The new basic feasible solution is [Ite #3]:

$$(x_1\ x_2\ x_3\ x_4\ x_5)^T = (3\,5\,3\,0\,0)^T$$ ***optimal solution***

where $x_B = (x_1\ x_2\ x_3)^T,\ x_N = (x_4\ x_5)^T$

The linear program has the new form:

$$\min \quad z = -13 + x_4 + 2x_5$$

$$s.t. \quad \cdots$$

Now, we cannot improve the objective!

Linear Programming

The Simplex Method: Definition

Given the set of *m* simultaneous linear equations in *n* unknowns ($m < n$)

$$Ax = b,$$

let B be any nonsingular $m \times m$ submatrix made up of columns of A.

Then, if all $n - m$ components of x not associated with columns of B are set equal to zero, the solution to the resulting set of equations is said to be a basic solution to Ax = b with respect to the basis B. The components of x associated with columns of B are called basic variables.

Linear Programming

The Simplex Method: Definition

A vector x satisfying

$$\begin{aligned} Ax &= b \\ x &\geqslant 0, \end{aligned} \qquad (**)$$

is said to be **feasible** for these constraints. A feasible solution to the constraints (**) that is also basic is said to be a **basic feasible solution**; if this solution is also a degenerate basic solution, it is called a degenerate basic feasible solution.

Linear Programming

- **The Simplex Method: Procedure**

Suppose we have a basic feasible solution

$$(\mathbf{x_B}, \mathbf{0}) = (y_{10}, y_{20}, \ldots, y_{m0}, 0, 0, \ldots, 0)$$

with a tableau having an identity matrix appearing in the first *m* columns

$$\begin{array}{cccccccc}
\mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_m & \mathbf{a}_{m+1} & \cdots & \mathbf{a}_n & \mathbf{b} \\
1 & 0 & & 0 & y_{1,m+1} & \cdots & y_{1n} & y_{10} \\
0 & 1 & & 0 & y_{2,m+1} & \cdots & y_{2n} & y_{20} \\
\cdot & \cdot & & \cdot & \cdot & & \cdot & \cdot \\
\cdot & \cdot & & \cdot & \cdot & & \cdot & \cdot \\
\cdot & \cdot & & \cdot & \cdot & & \cdot & \cdot \\
0 & 0 & & 1 & y_{m,m+1} & \cdots & y_{mn} & y_{m0}
\end{array}$$

Linear Programming

The Simplex Method: Procedure

The value of the objective function corresponding to any solution x is

$$z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

$$\Downarrow (\mathbf{x_B}, \mathbf{0})$$

$$z_0 = \mathbf{c}_\mathbf{B}^T \mathbf{x_B}$$

where $\mathbf{c}_\mathbf{B}^T = [c_1, c_2, \ldots, c_m]$.

Linear Programming

The Simplex Method: Procedure

From the tableau

$$x_1 = y_{10} - \sum_{j=m+1}^{n} y_{1j} x_j$$

$$x_2 = y_{20} - \sum_{j=m+1}^{n} y_{2j} x_j$$

$$\cdot$$

$$\cdot$$

$$\cdot$$

$$x_m = y_{m0} - \sum_{j=m+1}^{n} y_{mj} x_j .$$

Eliminate them from the objective function.

Linear Programming

The Simplex Method: Procedure

$$z = \mathbf{c}^T \mathbf{x} = z_0 + (c_{m+1} - z_{m+1})\, x_{m+1} + (c_{m+2} - z_{m+2})\, x_{m+2} + \cdots + (c_n - z_n)\, x_n$$

where

$$z_j = y_{1j}c_1 + y_{2j}c_2 + \cdots + y_{mj}c_m, \quad m+1 \leqslant j \leqslant n$$

If $r_j = c_j - z_j$ is negative for some j, $m+1 \le j \le n$, then increasing x_j from zero to some positive value would decrease the total cost, and therefore would yield a better solution.

Linear Programming

The Simplex Method: Procedure

Determination of Vector to Leave Basis

Suppose we have the basic feasible solution

$$x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \cdots + x_m\mathbf{a}_m = \mathbf{b} \quad x_i > 0, \; i = 1, 2, \ldots, m.$$

The representation the vector $\mathbf{a}_q, q > m$

$$\mathbf{a}_q = y_{1q}\mathbf{a}_1 + y_{2q}\mathbf{a}_2 + \cdots + y_{mq}\mathbf{a}_m$$

$$(x_1 - \varepsilon y_{1q})\mathbf{a}_1 + (x_2 - \varepsilon y_{2q})\mathbf{a}_2 + \cdots + (x_m - \varepsilon y_{mq})\mathbf{a}_m + \varepsilon\mathbf{a}_q = \mathbf{b}$$

Set ε equal to the value corresponding to the first place where one (or more) of the coefficients vanishes. $\varepsilon = \min_i \{x_i / y_{iq} : y_{iq} > 0\}$

Linear Programming

The Simplex Method: Algorithm

Step 0. Form the tableau corresponding to a basic feasible solution. The relative cost coefficients can be found by row reduction.

Step 1. If each $r_j \geq 0$, stop; the current basic feasible solution is optimal.

Step 2. Select q such that $r_q < 0$ to determine which nonbasic variable is to become basic.

Step 3. Calculate the ratios y_{i0}/y_{iq} for $y_{iq} > 0$, $i = 1, 2, \ldots, m$. If no $y_{iq} > 0$, stop; the problem is unbounded. Otherwise, select p as the index i corresponding to the minimum ratio.

Step 4. Pivot on the *pq*th element, updating all rows including the last. Return to *Step 1*.

Linear Programming

Linear Programming: Example 10

$$
\begin{aligned}
\max \quad & 7x_1 + 5x_2 \\
\text{s.t.} \quad & 3x_1 + 2x_2 \le 90 \\
& 4x_1 + 6x_2 \ge 200 \\
& 7x_2 \le 210 \\
& x_1, x_2 \ge 0
\end{aligned}
$$

```
f = [-7;-5]; A = [3,2;4,6;0,7];
b = [90;200;210]; lb = zeros(2,1);
[xStar,fval,exitflag] = linprog(f,A,b,[],[],lb)
```

xStar = $(14, 24)^T$

MATLAB Built-in Functions

- **MATLAB Built-in Functions for Optimization**
  - ✓ **Find minimum of single-variable function on fixed interval: *fminbnd***
  - ✓ **Find minimum of unconstrained multivariable function using derivative-free method: *fminsearch***
  - ✓ **Find minimum of unconstrained multivariable function: *fminunc***
  - ✓ **Find minimum of constrained nonlinear multivariable function: *fmincon***

Optimization

- **One-Dimensional Minimization**
  - **Golden Section Search**
  - **Newton's Method**
  - **Inaccurate Line Search**
- **Unconstrained Optimization**
  - **Steepest Descent Method**
  - **Newton's Method**
  - **Quasi-Newton Method**
- **Linear Programming**
  - ✓ **Geometry of Linear Programming**
  - ✓ **The Simplex Method**

Thank You !