

第二章：非线性方程数值解

内容概要

* 概述

* 二分法

* 迭代法

- 迭代法的基本思想
- 不动点迭代与收敛性
- 迭代过程的加速
- Newton-Raphson 法
- 割线法



概述

非线性方程的一般形式： $f(x) = 0$ (1)

这里 $f(x)$ 是变量 x 的函数，可以是代数多项式

$$f(x) = a_0 + a_1x + \cdots + a_nx^n$$

也可以是非多项式函数（例如对数函数、指数函数、三角函数等）。满足方程 (1) 的值 x^* 叫做方程的解或根，也叫函数 $f(x)$ 的零点。

4次以上的代数方程或者超越方程很难甚至无法求得精确解。本章将介绍常用的求解非线性方程的数值方法。

通常使用数值方法求解方程大致分为三个步骤：

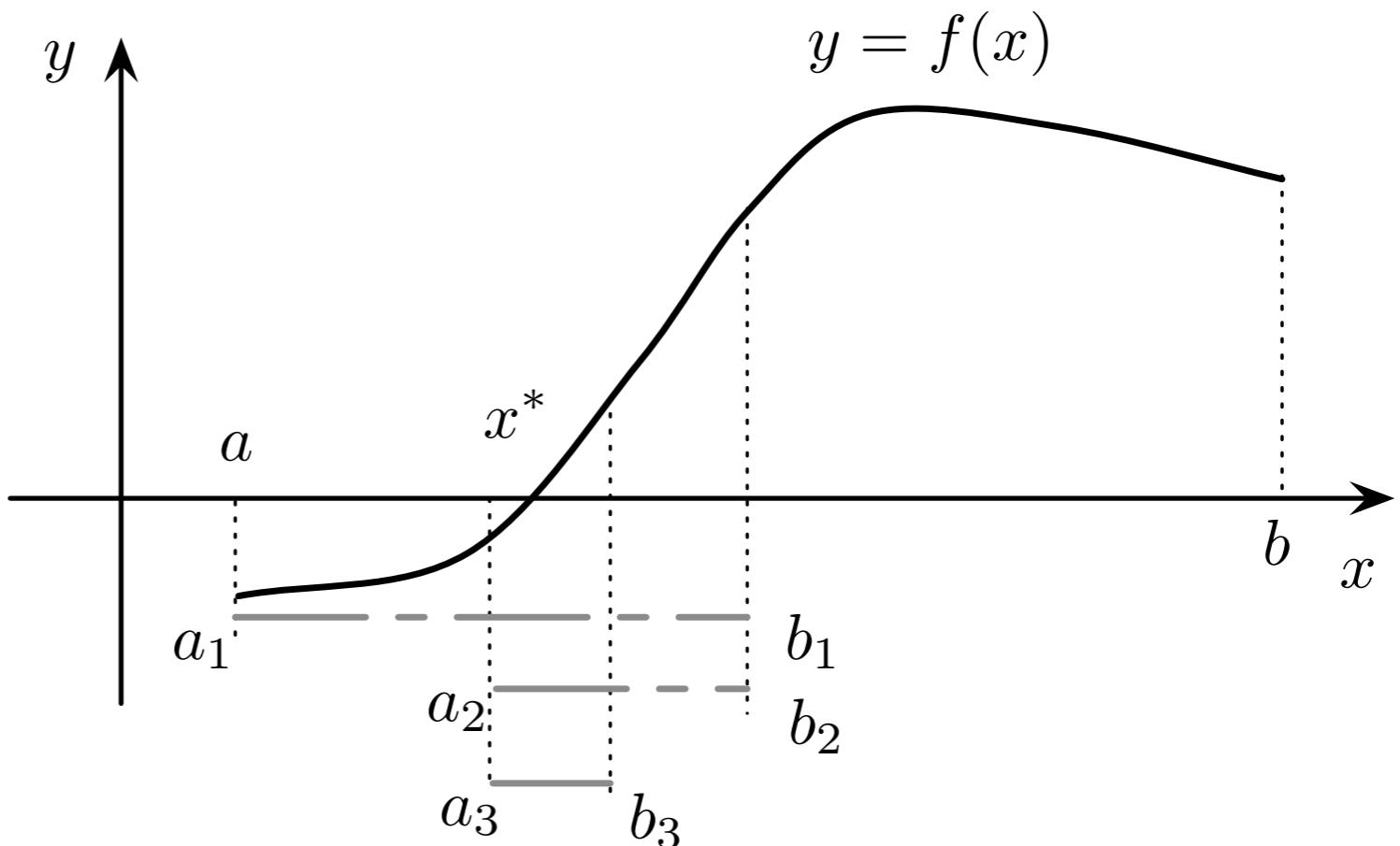
1. 判定根的存在性，方程有没有根，有几个根？
2. 确定根的分布范围，即确定有根区间，并将每一个根用不同的区间**隔开**。这样有根区间的任一点都可被视为根的一个近似值。
3. 根的精确化：对根的某个近似值**按某种方法**逐步精确化，使其满足给定的精度要求。

2.1 二分法(Bisection Method)

- * 二分法又称二分区间法，是求解方程根的最简单的方法。
- * 设函数 $f(x)$ 在闭区间 $[a, b]$ 上连续，且 $f(a)f(b) < 0$ ，根据连续函数的性质可知， $f(x) = 0$ 在 (a, b) 内必有实根，称区间 (a, b) 为方程的有根区间。为讨论方便，设方程在区间 (a, b) 只有一个实根 x^* 。
- * 二分法的基本思想是：将有根区间二等分，通过判断两端点函数值的符号，确定新的有根区间并再次将其二等分。依此逐步缩小有根区间，从而求出满足精度要求的近似根。
- * 二分法只能应用于有奇数次重根的问题。

2.1 二分法的计算过程

- 确定有唯一根区间
- 二等分为两个区间，判断新的有根区间
- 继续二等分，逐步缩小有根区间，直到满足精度要求
- 取最后一步的有根区间的中点作为近似值。



$$b_k - a_k = \frac{b - a}{2^k}$$

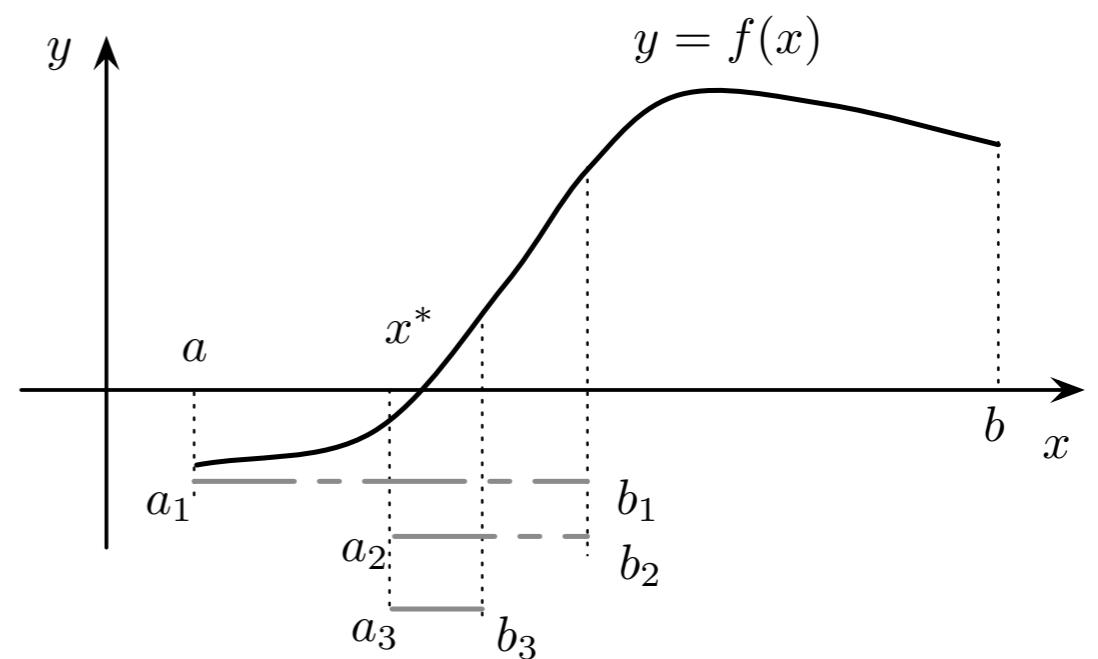
2.1 二分法的计算过程

- 对于给定的绝对精度要求 ε ，二分法终止的判断准则是

$$\frac{b_k - a_k}{2} \leq \varepsilon \quad \rightarrow \quad x_k \rightarrow x^*$$

$$\Rightarrow 2^{k+1} > \frac{b - a}{\varepsilon}$$

$$\Rightarrow k > \frac{\ln(b - a) - \ln 2\varepsilon}{\ln 2}$$



- 对于给定的精度要求，可以预先确定二分的次数。
- 优点：简单，收敛，只要求函数连续；
缺点：速度慢，不能求偶数重根。

Matlab 程序示范

bisect1.m, bisect2.m, bisect3.m

例2.1 求方程 $f(x) = x^3 - x - 1 = 0$ 在区间 $[1, 1.5]$ 内的实根，要求误差不超过0.005.

```
fun = inline('x^3-x-1');
[x_star, k] = bisect(fun, 1, 1.5, 0.005)
[x_star, k] = bisect2(fun, 1, 1.5, 0.005)
[x_star, index, k] = bisect3(fun, 1, 1.5, 0.005)
```

例2.2 求方程 $f(x) = \sin x - \frac{x^2}{4} = 0$ 在区间 $[1.5, 2]$ 内的实根。

```
fun = inline('sin(x) - x^2/4');
[x_star, k] = bisect(fun, 1.5, 2)
```

2.2 迭代法(iteration)

- * 二分法的主要思想是不断缩小有根区间，从而找出满足精度要求的有根区间，取最后的有根区间的中点作为方程的近似解。**循环体内得到的是新的有根区间。**
- * 迭代法用收敛于精确解的极限过程来逼近精确解，从而用有限步骤得到满足精度要求的近似解。迭代法是解方程最重要的数值方法。
- * 基本思想：已知方程的一个近似根，构造一个迭代格式来反复校正根的近似值，得到**根的近似值序列**，逐步精确化，直到满足给定的精度要求。对迭代格式的要求是得到的**序列收敛于方程的根**。

2.2.1 不动点迭代(Fixed point iteration)

- 把方程写成等价形式：

$$f(x) = 0 \Rightarrow x = \varphi(x)$$

迭代函数

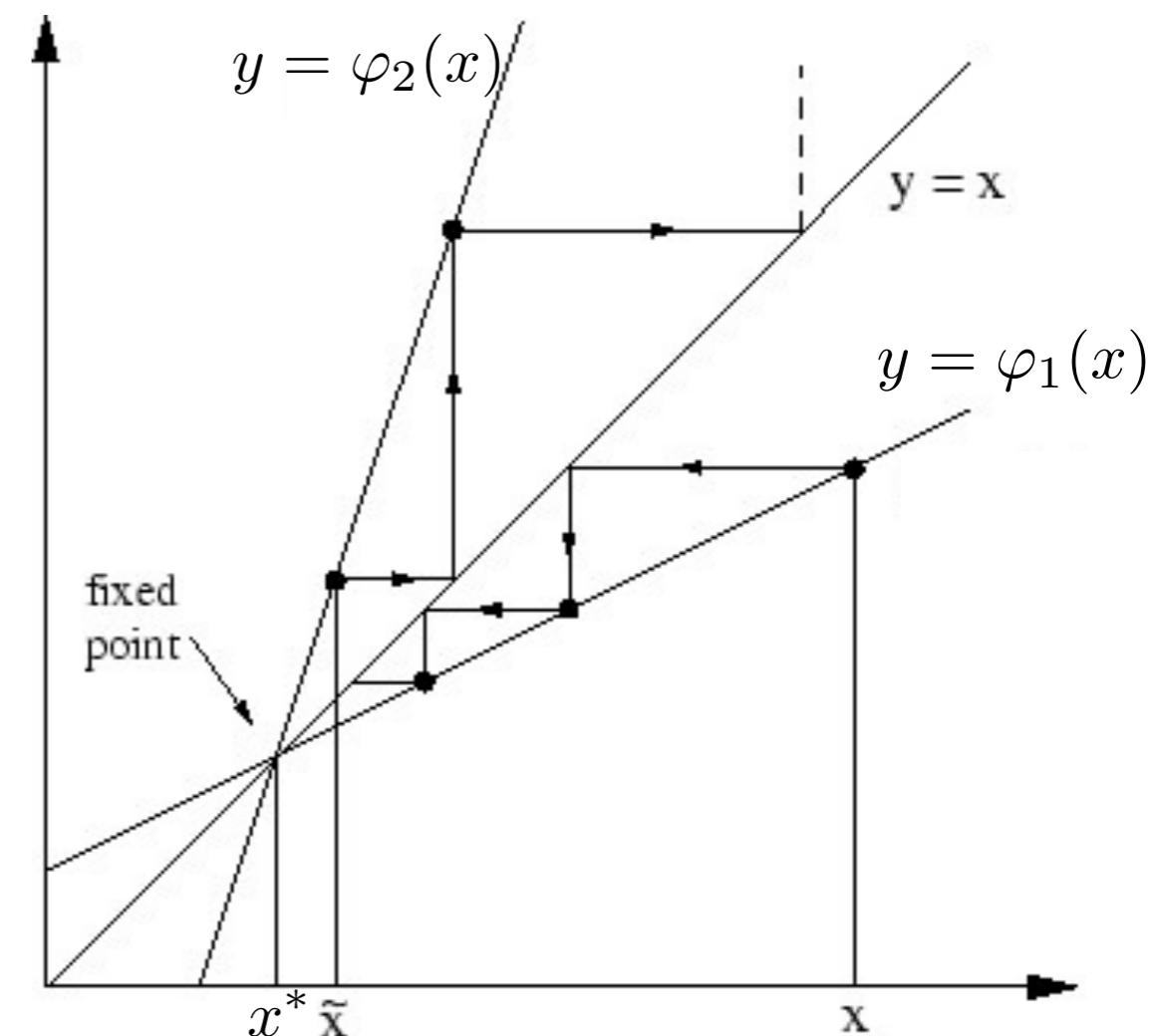
- 解方程问题转化为找函数 $y = \varphi(x)$ 与直线 $y = x$ 的交点的问题。这个交点就叫做**不动点**(fixed point)。

- 算法：

$$x_1 = \varphi(x_0), \quad x_2 = \varphi(x_1) \cdots$$

$$x_{k+1} = \varphi(x_k)$$

迭代公式



- 如果序列 $\{x_k\}$ 的极限存在，则迭代过程收敛：
$$x^* = \lim_{k \rightarrow \infty} x_k$$

Matlab 程序示范

fixedpoint.m

例2.3 求方程 $f(x) = x^3 - x - 1 = 0$ 在区间 $[1, 1.5]$ 内的实根。

构造迭代函数 $x = x^3 - 1$ ，取初值为 1.5

```
fun=inline('x^3-1');
[x, k] = fixedpoint(fun, 1.5);
```

发散

构造另一个等价的迭代函数 $x = \sqrt[3]{x + 1}$ ，取初值为 1.5

```
fun = inline('(x+1)^(1/3)');
[x, k] = fixedpoint(fun, 1.5)
```

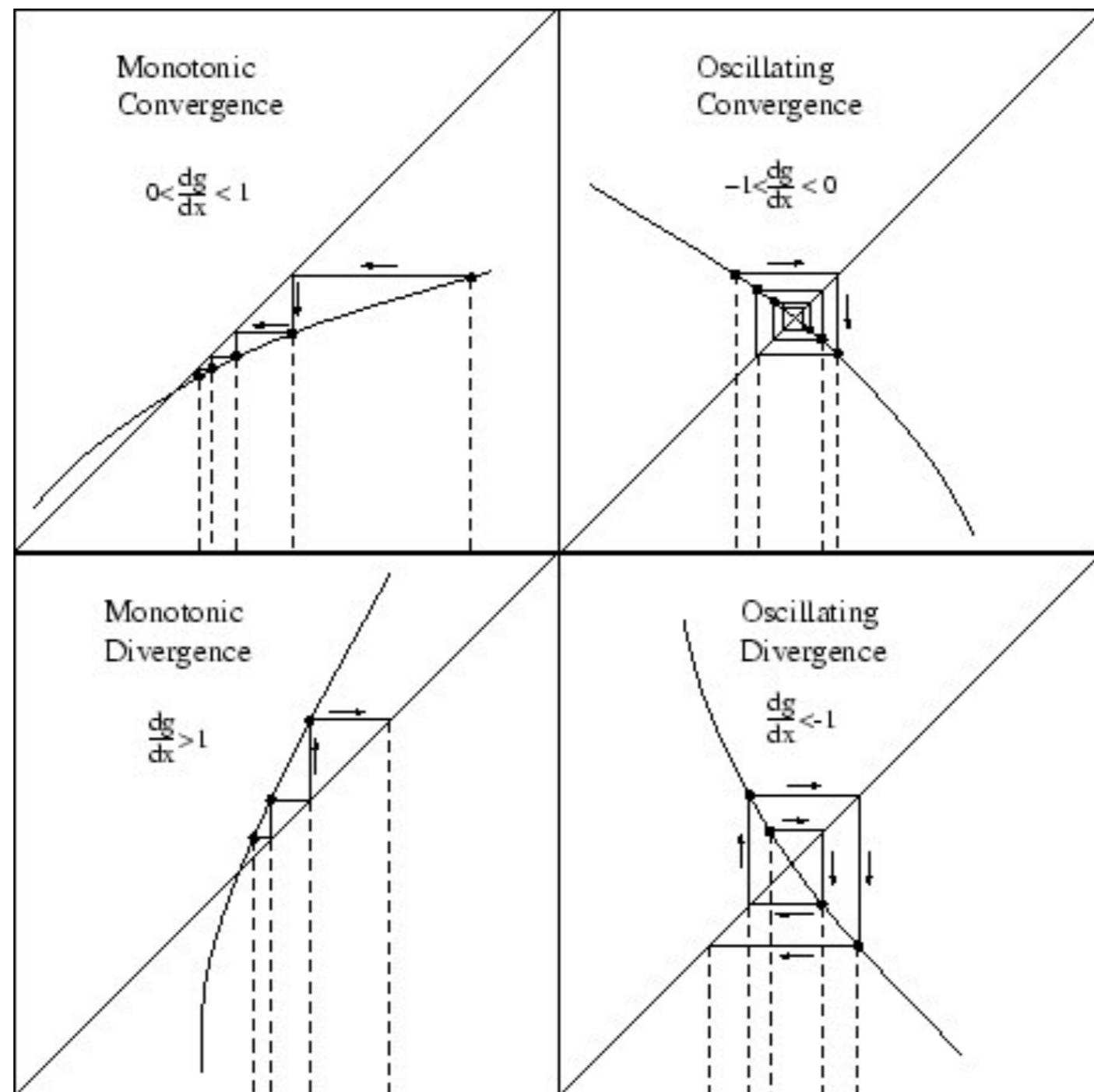
7次迭代得到结果

例2.4 求方程 $x = e^{-x}$ 在 $x=0.5$ 附近的实根。

```
fun = inline('exp(-x)');
[x, k] = fixedpoint(fun, 0.5)
```

18次迭代得到结果

不动点迭代的收敛性



不动点迭代的收敛性定理

定理2.1 设迭代函数 $\varphi(x)$ 满足条件：

- (1) 当 $x \in [a, b]$ 时， $a \leq \varphi(x) \leq b$ ；
- (2) 存在正数 $L < 1$ ，使任意 $x, y \in [a, b]$ ，都有
$$|\varphi(x) - \varphi(y)| \leq L|x - y|$$
 成立。

则方程 $x = \varphi(x)$ 在 $[a, b]$ 上有唯一解 x^* ，且对任意初始近似值 $x_0 \in [a, b]$ 迭代过程 $x_{k+1} = \varphi(x_k)$ 收敛，即

$$\lim_{k \rightarrow \infty} x_k = x^*$$

解读：

1. 函数 $\varphi(x)$ 在 y 轴上的变化小于在 x 轴上的变化。
2. 不要求连续可微。

不动点迭代的收敛性定理

定理2.2 设迭代函数 $\varphi(x)$ 满足条件:

$\varphi(x)$ 连续可微

- (1) 当 $x \in [a, b]$ 时, $a \leq \varphi(x) \leq b$;
- (2) 存在正数 $L < 1$, 使任意 $x \in [a, b]$, 都有
 $|\varphi(x)'| \leq L < 1$ 成立。

则方程 $x = \varphi(x)$ 在 $[a, b]$ 上有唯一解 x^* , 且对任意初始近似值 $x_0 \in [a, b]$ 迭代过程 $x_{k+1} = \varphi(x_k)$ 收敛, 即

$$\lim_{k \rightarrow \infty} x_k = x^*$$

定理2.3 在定理2.2的条件下, 有误差公式:

$$|x^* - x_k| \leq \frac{1}{1-L} |x_{k+1} - x_k| , \quad |x^* - x_k| \leq \frac{L^k}{1-L} |x_1 - x_0|$$

迭代法的收敛速度

定义2.2 设迭代过程 $x_{k+1} = \varphi(x_k)$ 收敛于方程 $x = \varphi(x)$ 的根 x^* ，设误差 $e_k = x^* - x_k$ ，若存在某实数 $p \geq 1$ 及常数 $C \neq 0$ ，使

$$\lim_{k \rightarrow \infty} \frac{|e_{k+1}|}{|e_k|^p} = C$$

则称迭代过程 $x_{k+1} = \varphi(x_k)$ 是 p 阶收敛的。

当 $p = 1$ 时，称为线性收敛，当 $p = 2$ 时，称为平方收敛。

定理2.5 对于迭代过程 $x_{k+1} = \varphi(x_k)$ ，如果 $\varphi^{(p)}(x)$ 在所求根 x^* 的邻近连续，并且

$$\varphi'(x^*) = \varphi''(x^*) = \dots = \varphi^{(p-1)}(x^*) = 0 \quad \varphi^{(p)}(x^*) \neq 0$$

则该迭代过程在点 x^* 附近是 p 阶收敛的。

迭代过程的加速

如果某迭代收敛，但是 $\varphi'(x^*) \neq 0$ ，则该迭代过程是线性收敛或者亚线性收敛的。线性收敛速度很慢，可以对迭代函数进行改进来对迭代过程进行加速。

迭代加速法一：

设原迭代过程为 $x_{k+1} = \varphi(x_k)$ ，令新的迭代函数为：

$$\phi(x) = \varphi(x) + \lambda(\varphi(x) - x)$$

其中 $\lambda \neq 1$ 是待定系数。思路是选择合适的系数 λ 使得 $\phi'(x^*) = 0$ 对新的迭代函数 $\phi(x)$ 求导，得：

$$\phi'(x^*) = \varphi'(x^*) + \lambda(\varphi'(x^*) - 1)$$

以此求得系数：

$$\lambda = \frac{\varphi'(x^*)}{1 - \varphi'(x^*)}$$

此系数使新的迭代函数在 x^* 的导数为零。

迭代过程的加速

但是由于精确解 x^* 无法得到，通常用 x_k 来代替 x^* ，令 $\omega_k = \varphi'(x_k)$ 系数由以下公式来近似：

$$\lambda \approx \frac{\omega_k}{1 - \omega_k}$$

将此表达式代入迭代函数，得到加速的迭代公式：

$$\begin{aligned} x_{k+1} &= \phi(x_k) = \varphi(x_k) + \frac{\omega_k}{1 - \omega_k} (\varphi(x_k) - x_k) \\ &= \frac{1}{1 - \omega_k} \varphi(x_k) - \frac{\omega_k}{1 - \omega_k} x_k \end{aligned}$$

若 $\varphi'(x)$ 在搜索区域内改变不大，其估计值为 L ，可用 L 来近似 ω_k ，得到新的加速公式：

$$x_{k+1} = \frac{1}{1 - L} \varphi(x_k) - \frac{L}{1 - L} x_k$$

迭代过程的加速

迭代加速法二（松弛法）：

在迭代函数 $x = \varphi(x)$ 的两端加上 $\lambda \cdot x$ ，得到方程

$$(1 + \lambda)x = \lambda x + \varphi(x)$$

$$x = \frac{\lambda}{1 + \lambda}x + \frac{1}{1 + \lambda}\varphi(x)$$

此方程与原方程 $x = \varphi(x)$ 等价，记等式右端为 $\phi(x)$ 。选取 λ 使得 $\phi'(x)$ 比 $\varphi'(x)$ 小。对等式右端求导得：

$$\phi'(x) = \frac{1}{1 + \lambda}(\lambda + \varphi'(x))$$

选取 $\lambda = -\varphi'(x^*)$ 使得 $\phi'(x^*) = 0$ ，但由于 x^* 未知，用 x_k 来代替 x^* ， $\lambda_k = -\varphi'(x_k)$ 。

迭代过程的加速

记 $\omega_k = \frac{1}{1 + \lambda_k}$ ，代入迭代公式，得到松弛法的迭代公式：

$$\begin{cases} \omega_k = \frac{1}{1 - \varphi'(x_k)}, \\ x_{k+1} = (1 - \omega_k)x_k + \omega_k\varphi(x_k) \end{cases}$$

其中 ω_k 称为松弛因子。

上述两种加速方法都需要计算导数 $\varphi'(x)$ 在 x_k 的值。

迭代过程的加速

迭代加速法三 (**Aitken** 加速法) :

此方法不需要计算导数 $\varphi'(x)$ 。设 \bar{x}_{k+1} 是 x_k 经过一次迭代得到的结果:

$$\bar{x}_{k+1} = \varphi(x_k)$$

由微分中值定理有: $x^* - \bar{x}_{k+1} = \varphi'(\xi)(x^* - x_k)$

其中 ξ 是 x^* 与 x_k 之间的某个点。假设 $\varphi'(x)$ 在求根区间变化不大, 用 L 来近似, 得到

$$x^* - \bar{x}_{k+1} \approx L(x^* - x_k) \quad (1)$$

对迭代值 \bar{x}_{k+1} 再用迭代公式校正一次得: $\bar{\bar{x}}_{k+1} = \varphi(\bar{x}_{k+1})$

同样由微分中值定理有: $x^* - \bar{\bar{x}}_{k+1} \approx L(x^* - \bar{x}_k)$ (2)

迭代过程的加速

(2) 式除以 (1) 式, 有:

$$\frac{x^* - \bar{\bar{x}}_{k+1}}{x^* - \bar{x}_{k+1}} \approx \frac{x^* - \bar{x}_{k+1}}{x^* - x_k}$$

求 x^* :

$$x^* \approx \bar{\bar{x}}_{k+1} - \frac{(\bar{\bar{x}}_{k+1} - \bar{x}_{k+1})^2}{\bar{\bar{x}}_{k+1} - 2\bar{x}_{k+1} + x_k}$$

得到新的迭代过程:

$$x_{k+1} = \bar{\bar{x}}_{k+1} - \frac{(\bar{\bar{x}}_{k+1} - \bar{x}_{k+1})^2}{\bar{\bar{x}}_{k+1} - 2\bar{x}_{k+1} + x_k}$$

Aitken迭代加速算法:

$$\begin{cases} \bar{x}_{k+1} = \varphi(x_k), \\ \bar{\bar{x}}_{k+1} = \varphi(\bar{x}_{k+1}), \\ x_{k+1} = \bar{\bar{x}}_{k+1} - \frac{(\bar{\bar{x}}_{k+1} - \bar{x}_{k+1})^2}{\bar{\bar{x}}_{k+1} - 2\bar{x}_{k+1} + x_k} \end{cases}$$

Matlab 程序示范

Aitken.m

例2.6 求方程 $f(x) = x^3 - x - 1 = 0$ 在 $x=1.5$ 附近的实根。

构造迭代函数: $x = x^3 - 1$

```
fun = inline('x^3-1');
[x, k] = Aitken(fun, 1.5)
```

5次迭代得到结果

加速算法不仅能对迭代加速，而且可以增加迭代的稳定性。

例2.7 求方程 $x = e^{-x}$ 在 $x=0.5$ 附近的实根。

```
fun = inline('exp(-x)');
[x, k] = Aitken(fun, 0.5)
```

2次迭代得到结果

迭代过程的加速

迭代加速法四（**Steffensen**加速法）：

与Aitken法基本一致，modified Aitken's acceleration。

$$\left\{ \begin{array}{l} \bar{x}_k = \varphi(x_k), \\ \bar{\bar{x}}_k = \varphi(\bar{x}_k), \\ x_{k+1} = \bar{x}_k - \frac{(\bar{x}_k - x_k)^2}{\bar{\bar{x}}_k - 2\bar{x}_k + x_k} \end{array} \right.$$

存在争议

```
fun = inline('x^3-l');
[x, k] = SteffensenI(fun, l.5)
fun = inline('exp(-x)');
[x, k] = SteffensenI(fun, 0.5)
```

Newton's Method

(one of the most important!)

Newton 法 (切线法)

对原方程进行线性化近似，得到迭代公式，对方程的解进行逼近。
将函数 $f(x)$ 在 x_0 处进行 Taylor 级数展开：

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots$$

用一阶展开式来近似函数 $f(x)$ ，有：

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0)$$

于是非线性方程 $f(x) = 0$ 可近似表达为 x 的线性方程：

$$f(x_0) + f'(x_0)(x - x_0) = 0$$

得到迭代方程：

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \quad x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$



Newton 法的几何意义

- 方程 $f(x) = 0$ 的根是曲线 $y = f(x)$ 与 x 轴交点的横坐标。

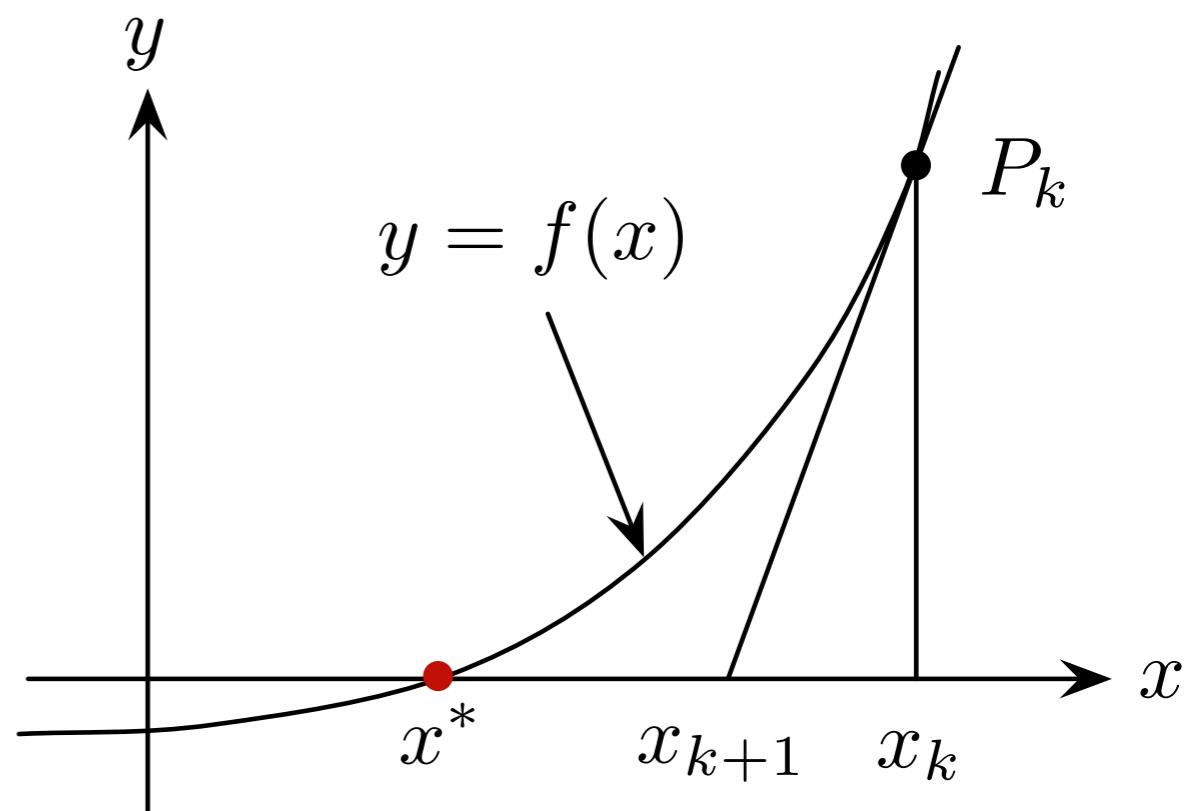
- 过 x_k 对应点 P_k 做 $y = f(x)$ 的切线，切线方程为：

$$y - f(x_k) = f'(x_k)(x - x_k)$$

- 切线与 x 轴的交点： $y = 0$

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

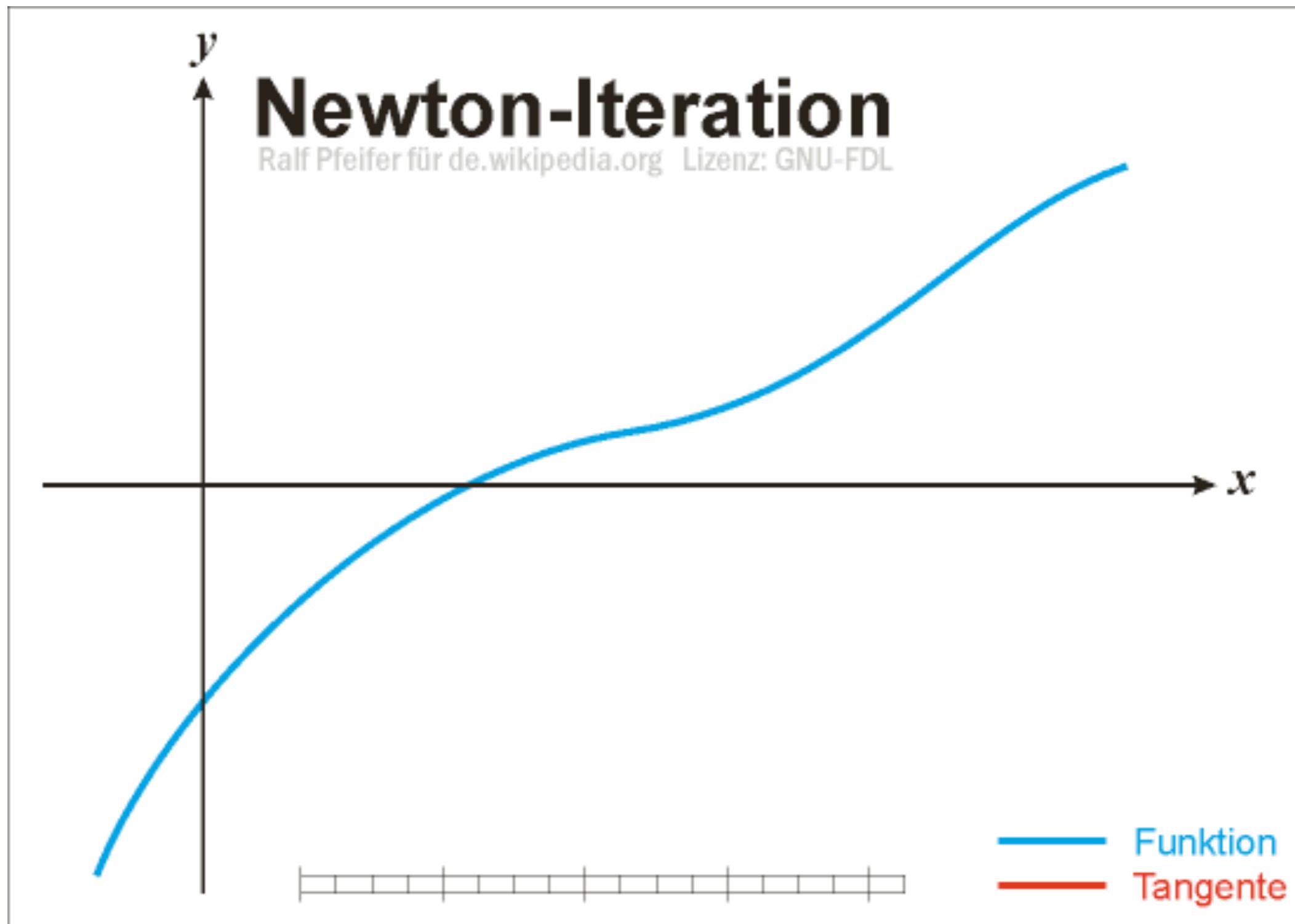
- 与 Newton 法的迭代公式吻合，所以 Newton 法也叫切线法。



Newton 法的计算步骤

- (1) 选定初值 x_0 , 计算 $f(x_0)$, $f'(x_0)$;
- (2) 计算新的近似值 $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$;
- (3) 如果满足精度要求 $|x_{k+1} - x_k| < \varepsilon$, 终止迭代, 并取 $x^* = x_{k+1}$ 为所求根的近似值。否则, $k = k + 1$, 再转步骤 (2) 计算。如果迭代次数超过规定的最大迭代次数 N_{\max} , 或者计算过程中出现 $f'(x_{k+1}) = 0$, 则认为 Newton 法解方程失败。

Newton 迭代过程演示



Matlab 程序示范

Newton1.m, Newton2.m

例2.10 求方程 $f(x) = x^3 + 2x^2 + 10x - 20 = 0$ 在 $x=1$ 附近的实根。

```
fun = inline('x^3+2*x^2+10*x-20');
dfun = inline('3*x^2 + 4*x+10')
[x_a ,k] = Newton1(fun,dfun,1)
[x_a ,index, k] = Newton2(fun,dfun,1)
```

Newton 法的收敛性

定理2.6 设函数 $f(x)$ 在区间 $[a, b]$ 上存在二阶连续导数，且满足条件：

- | | |
|---|-----------|
| (1) $f(a)f(b) < 0$; | 根的存在 |
| (2) 当 $x \in [a, b]$ 时， $f'(x) \neq 0$; | 函数单调，根的唯一 |
| (3) 当 $x \in [a, b]$ 时， $f''(x)$ 不变号； | 凹凸性 |
| (4) $\frac{f(a)}{f'(a)} \leq (a - b), \quad \frac{f(b)}{f'(b)} \leq (b - a).$ | 迭代在区间内 |

则对于任意初始值 $x_0 \in [a, b]$ ，Newton 迭代确定的序列收敛于方程 $f(x) = 0$ 在区间 $[a, b]$ 内的唯一根 x^* 。

Newton 法的收敛速度

如果 $f(x)$ 在根 x^* 附近有连续的二阶导数，且 x^* 是单根，则在根 x^* 附近，Newton 迭代法具有二阶的收敛速度。

证明：Newton 法的迭代函数 $\varphi(x) = x - \frac{f(x)}{f'(x)}$

$$\varphi'(x) = \frac{f(x)f''(x)}{(f'(x))^2}, \quad f(x^*) = 0, \quad f'(x^*) \neq 0,$$

$$\Rightarrow \varphi'(x^*) = 0$$

$$\varphi''(x^*) = \frac{f''(x^*)}{f'(x^*)}, \quad \text{只要 } f''(x^*) \neq 0 \text{ 就有 } \varphi''(x^*) \neq 0$$

Newton 法是平方收敛的。

Newton 法的收敛性例题

例2.11 用 Newton 法求方程 $xe^x - 1 = 0$ 在区间 $[0, 1]$ 上的根。

解：首先用定理2.6判定使用Newton法解此方程的收敛性。

$$f(x) = xe^x - 1 \quad f'(x) = e^x + xe^x \quad f''(x) = 2e^x + xe^x$$

- (1) $f(0) = -1, f(1) = e - 1, f(0)f(1) < 0$
- (2) $x_0 \in [0, 1], f'(x_0) > 0$
- (3) $x_0 \in [0, 1], f''(x_0) > 0$
- (4) $f(0) = -1, f'(0) = 1, f(1) = e - 1, f'(1) = 2e$

$$\left| \frac{f(0)}{f'(0)} \right| = 1 \quad \left| \frac{f(1)}{f'(1)} \right| = \frac{e-1}{2e}$$

Newton 法的收敛性例题

∴ 对于 $\forall x_0 \in [0, 1]$, Newton 迭代格式

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

收敛于方程在区间 $[0, 1]$ 上的根。

```
fun = inline('x*exp(x)-1');
dfun = inline('(1+x)*exp(x)');
[x_star, k] = newtonl(fun, dfun, 0.5);
```

4次迭代得到结果

Newton 法的收敛性与初值

定理2.7 设函数 $f(x)$ 在区间 $[a, b]$ 内有一阶导数和二阶导数，如

果 $x_0 \in [a, b]$ ，满足 (1) $f'(x_0) \neq 0, f''(x_0) \neq 0$ ；

(2) $|f'(x_0)|^2 > \left| \frac{f''(x_0)}{2} \right| |f(x_0)|$ ；则 x_0 作为初始近似值的 Newton 迭代格式 $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$ 所确定的序列收敛于 $f(x) = 0$ 的根 x^* 。

Newton 法解题全过程例题

例2.12 求 $\sqrt{115}$, 精度要求为 10^{-5} 。

构造方程 $f(x) = x^2 - 115 = 0$, $\sqrt{115}$ 是方程的正根, 问题转化为求此正根的近似值。

(1) 确定根 $x^* = \sqrt{115}$ 的范围。 $f(10) < 0$, $f(11) > 0$, 所以 $x^* \in [10, 11]$ 。

(2) 取初值为 $x_0 = 10$ 。此初值满足:

$$f'(x) = 2x, f''(x) = 2, f(10) = -15, f'(10) = 20, f''(10) = 2,$$

$$|f'(x_0)|^2 = 400 > \left| \frac{f''(x_0)}{2} \right| |f(x_0)| = 15$$

所以取 $x_0 = 10$ 为初值, Newton 迭代收敛。

Newton 法解题全过程例题

```
fun = inline('x*x-115');
dfun = inline('2*x');
[x_star, k] = newtonl(fun, dfun, 10)
```

4次迭代得到结果

Newton 平行弦法

迭代公式为： $x_{k+1} = x_k - C f(x_k)$, $C \neq 0, k = 1, 2, \dots$

$$C = \frac{1}{f'(x_0)} = \text{const.}$$

若 $|\varphi(x)| = |1 - Cf'(x)| < 1$ ，即 $0 < Cf'(x) < 2$ 在真实值附近成立，则迭代过程局部收敛。

用常数 C 来代替 $\frac{1}{f'(x_k)}$ ，不用在每一个估计值处都计算导数 $f'(x_k)$ ，从而减少了计算量。Newton 平行弦法又称为**简化Newton法**。

- 收敛较慢
- 计算简单



Newton 法初值问题示例

Newton 法具有局部收敛性，依赖初值 x_0 的选取。如果 x_0 偏离 x^* 较远，Newton 法有可能（不一定）会发散。见例2.13.

例2.13 用Newton法求方程 $f(x) = x^3 - x - 1 = 0$ 在1.5附近的一个根。

```
fun = inline('x^3-x-1');
dfun = inline('3*x^2-1');
[x_star, k] = newtonl(fun, dfun, 1.5);
```

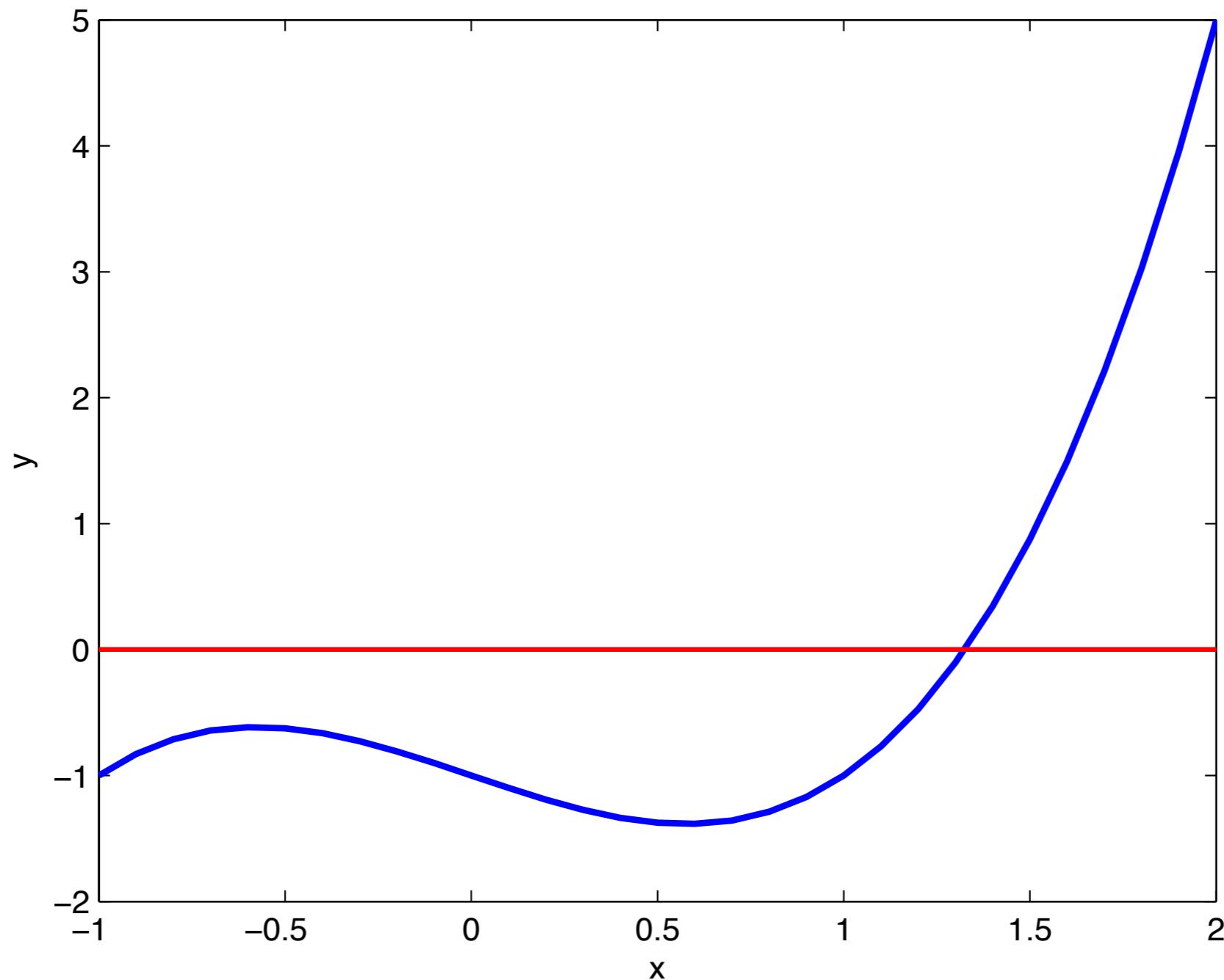
x_1	= +1.3478261e+00	$f(x)$	= +1.0068217e-01
x_2	= +1.3252004e+00	$f(x)$	= +2.0583619e-03
x_3	= +1.3247182e+00	$f(x)$	= +9.2437776e-07
x_4	= +1.3247180e+00	$f(x)$	= +1.8651747e-13

Newton 法初值问题示例

```
[x_star, k] = newtonl(fun, dfun, .6);
```

x_1 = +1.7900000e+01	f(x) = +5.71644e+03	df(x) = +8.00000e-02
x_2 = +1.1946802e+01	f(x) = +1.69217e+03	df(x) = +9.60230e+02
x_3 = +7.9855204e+00	f(x) = +5.00239e+02	df(x) = +4.27178e+02
x_4 = +5.3569093e+00	f(x) = +1.47368e+02	df(x) = +1.90306e+02
x_5 = +3.6249960e+00	f(x) = +4.30096e+01	df(x) = +8.50894e+01
x_6 = +2.5055892e+00	f(x) = +1.22244e+01	df(x) = +3.84218e+01
x_7 = +1.8201294e+00	f(x) = +3.20972e+00	df(x) = +1.78339e+01
x_8 = +1.4610441e+00	f(x) = +6.57774e-01	df(x) = +8.93861e+00
x_9 = +1.3393232e+00	f(x) = +6.31370e-02	df(x) = +5.40395e+00
x_10 = +1.3249129e+00	f(x) = +8.31373e-04	df(x) = +4.38136e+00
x_11 = +1.3247180e+00	f(x) = +1.50938e-07	df(x) = +4.26618e+00
x_12 = +1.3247180e+00	f(x) = +4.88498e-15	df(x) = +4.26463e+00

Newton 法初值问题示例



Newton 下山法

以上演示的Newton法不足的原因：在 $|f'(x_k)|$ 很小处Newton法过于敏感，迭代步长过大。

解决途径：**Newton** 下山法

引入下山因子 λ ，增加Newton法迭代的稳定性：

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

下山因子按 $\lambda = 1, \frac{1}{2}, \frac{1}{4}, \dots$ 选取，直到满足： $|f(x_{k+1})| < |f(x_k)|$

思想：减小迭代敏感处的迭代步长，使得迭代的每一步都在“**下山**”，从而增加迭代稳定性，并加速收敛。

Newton 下山法例题

例2.14 用Newton法求方程 $f(x) = x^3 - x - 1 = 0$ 在1.5附近的一个根。

与例2.13相同，取初值为 0.6，用Newton 下山法迭代。

```
fun = inline('x^3-x-1');
dfun = inline('3*x^2-1');
[x_star, k] = Newton3(fun, dfun, .6);
```

k	x	f(x)	lambda	abs(f)
1	+1.1406250e+00	-6.56643e-01	0.03125	+6.5664291e-01
2	+1.3668137e+00	+1.86640e-01	1.00000	+1.8663972e-01
3	+1.3262798e+00	+6.67040e-03	1.00000	+6.6704015e-03
4	+1.3247202e+00	+9.67388e-06	1.00000	+9.6738769e-06

Newton 下山法例题

例2.15 用Newton下山法求方程 $x = e^{-x}$ 在0.5附近的一个根。

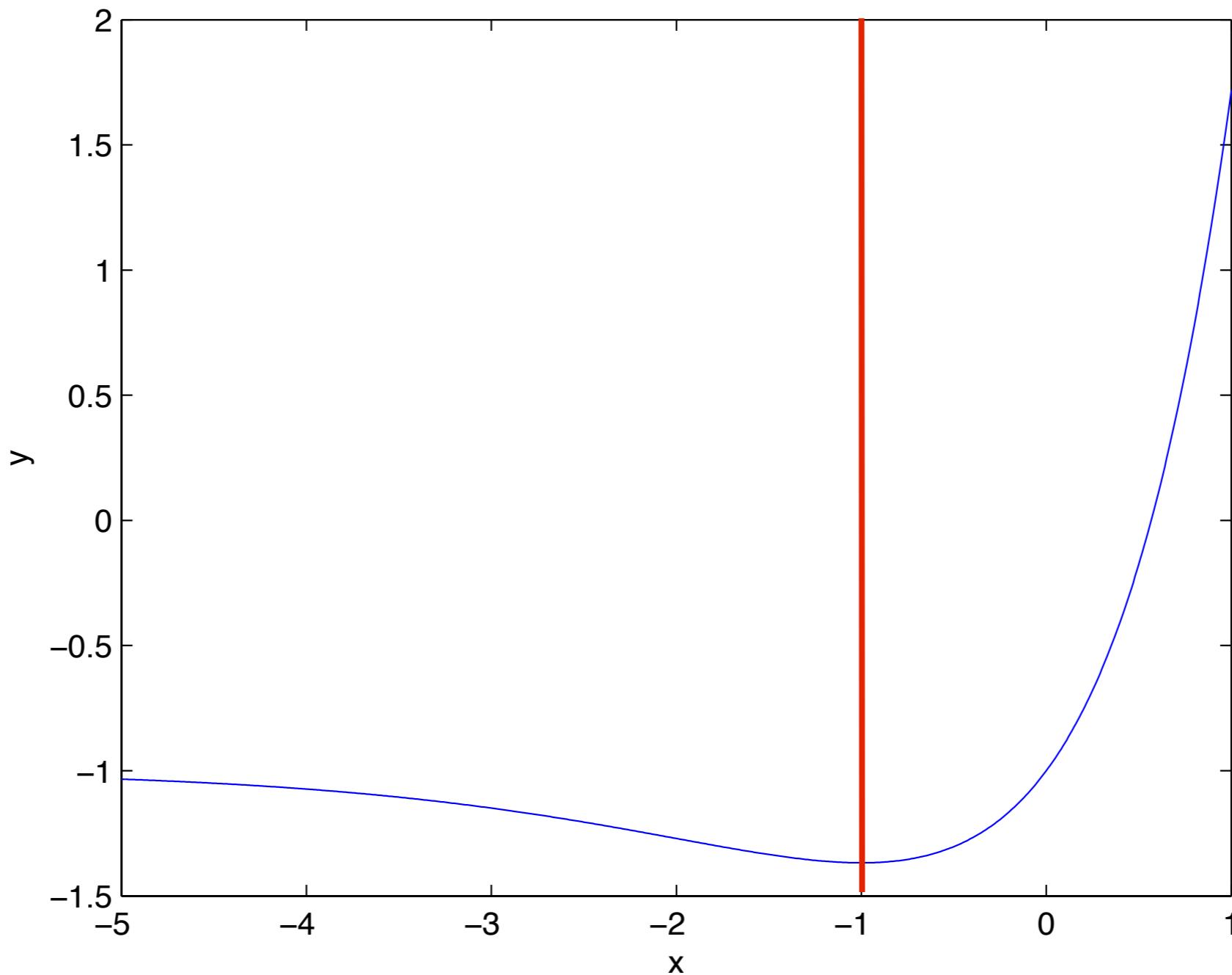
构造函数 $f(x) = xe^x - 1$, 取初值 -0.99, 用Newton法和Newton下山法解方程。

```
fun = inline('x*exp(x)-1');
dfun = inline('(1+x)*exp(x)');
[xstar, k] = newton1(fun, dfun, -0.99)      378次迭代得到结果
[xstar, k] = Newton3(fun, dfun, -0.99)      4次迭代得到结果
```

如果初值取为 -1 :

```
[xstar, k] = Newton3(fun, dfun, -1.0)      Newton法失效
fun = inline('exp(-x)');
[xstar, k] = Steffensen1(fun, -1)            4次迭代得到结果
```

Newton 下山法例题



割线法 (Secant Method)

- * Newton 法用切线来近似曲线得到方程解的估计值。
- * 割线法用过两点的割线来近似曲线得到方程解的估计值。
- * 对于非线性方程 $f(x) = 0$ ，假设 $f(x)$ 在 $[a, b]$ 上连续，且 $f(a)f(b) < 0$ 将过曲线上两点 $(x_{k-1}, f(x_{k-1}))$ 和 $(x_k, f(x_k))$ 的割线与 x 轴的交点的横坐标 x_{k+1} 作为方程 $f(x) = 0$ 的近似根。

割线的方程为： $y = f(x_k) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x - x_k)$

令 $y = 0$ ，得割线与 x 轴交点的横坐标

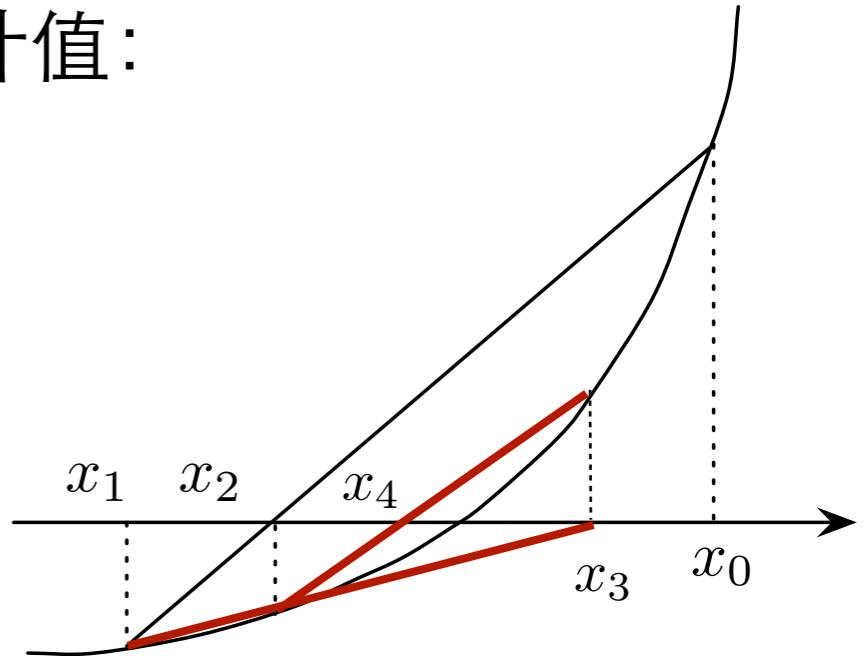
$$x = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k)$$

割线法 (Secant Method)

令割线与 x 轴交点的横坐标为方程新的估计值：

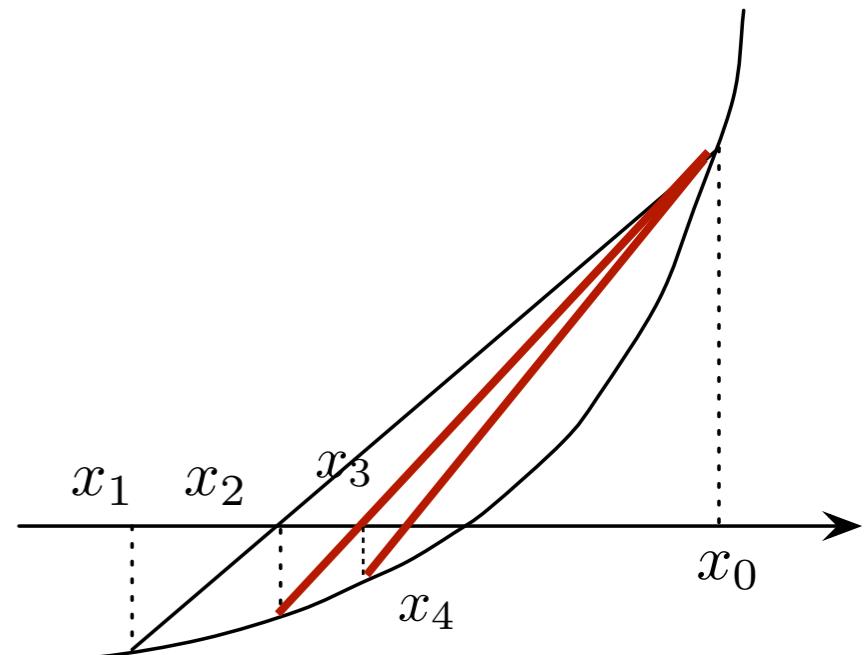
$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k),$$
$$k = 1, 2, \dots$$

双点割线法的迭代公式



$$x_{k+1} = x_k - \frac{x_k - x_0}{f(x_k) - f(x_0)} f(x_k),$$
$$k = 1, 2, \dots$$

单点割线法的迭代公式



割线法的收敛速度

定理2.8 设方程 $f(x) = 0$ 的根为 x^* ，若 $f(x)$ 在 x^* 附近有二阶连续导数， $f'(x) \neq 0$ ，而且初值 x_0, x_1 充分接近 x^* ，则双点割线法的迭代过程收敛，收敛速度为

$$|x_{k+1} - x^*| \approx \left| \frac{f''(x^*)}{2f'(x^*)} \right|^{0.618} |x_k - x^*|^{0.618}$$

即双点割线法是超线性收敛的，收敛阶数 $0 < p = 0.618 < 1$

单点割线法是线性收敛的。

sublinear convergence
亚线性收敛

割线法例题

例2.16 用双点割线法求方程 $2x^3 - 5x - 1 = 0$ 在区间 $[1, 2]$ 上的根。

```
fun = inline('2*x^3 - 5*x-1')
[x_star, k] = Secant(fun, 1, 2)
```

```
x_1 = +1.4444444e+00
x_2 = +1.61391802e+00
x_3 = +1.68710364e+00
x_4 = +1.67225044e+00
x_5 = +1.67297293e+00
x_6 = +1.67298165e+00
```

6次迭代得到结果

小结

- * 二分法最简单，但速度慢、无法计算偶次重根。
- * 不动点迭代是一类方法。四种加速算法。
- * Newton法速度快，要求函数连续可微，且导数在除了方程的根之外的搜索不能为零，局部收敛。可计算重根。
- * Newton下山法收敛更加稳定，速度比Newton法快。
- * 割线法简单，不要求连续可微，但需要更多存储空间（需要存储前两步的计算数据）。

作业：课后 1(1), 3(2), 4, 10

取3位有效数字

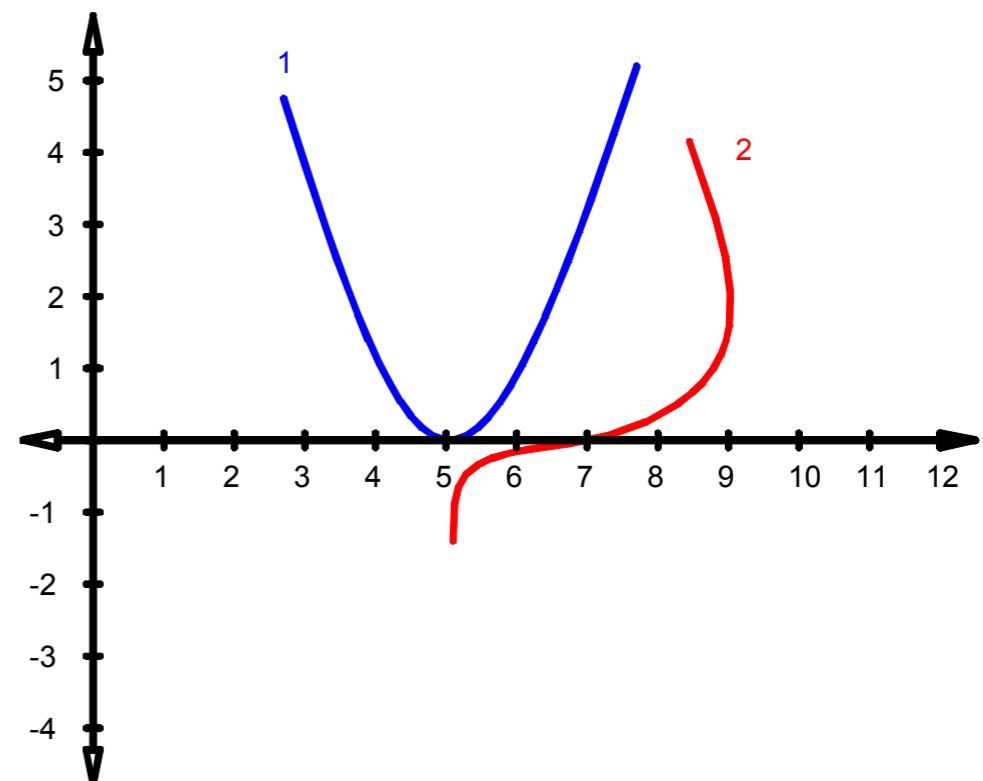
重根问题 (补充内容)

定义：对于方程 $f(x) = 0$ 的一个根 $x = p$ ，如果 p 是方程的 m 重根，则函数 $f(x)$ 可表示为

$$f(x) = (x - p)^m q(x)$$

其中 $q(x)$ 是连续可微函数且 $q(p) \neq 0$ 。

- 函数曲线在重根处与 x 轴平行 ($f'(p) = 0$)
- 例如 $f(x) = 2x^3 - 4x^2 + 2x = (x-1)^2(2x)$



重根问题 (补充内容)

对于方程 $f(x) = (x-p)^m q(x)$

$$f'(p) = f''(p) = \dots = f^{(m-1)}(p) = 0$$

Newton 法求解单根是平方收敛的；Newton 法求重根是线性收敛的。

为保持平方收敛速度，并避免小数作除数，定义新的函数

$$\mu(x) = \frac{f(x)}{f'(x)}$$

则 $x = p$ 是方程 $\mu(x) = 0$ 的一个单根。

$$\mu(x) = \frac{f(x)}{f'(x)} = \frac{(x-p)^m q(x)}{m(x-p)^{m-1} q(x) + (x-p)^m q'(x)} = (x-p) \frac{q(x)}{mq(x) + (x-p)q'(x)}$$



重根问题 (补充内容)

对方程 $\mu(x) = 0$ 用Newton法求解，获得平方收敛速度。

修正的**Newton**法：

$$x_{n+1} = x_n - \frac{\mu(x_n)}{\mu'(x_n)}$$

$$\mu' = \frac{(f'^2 - ff'')}{f'^2} \text{ where } \mu(x) = \frac{f(x)}{f'(x)}$$

$$\Rightarrow \frac{\mu}{\mu'} = \left(\frac{ff'}{f'^2 - ff''} \right)$$

$$x_{n+1} = x_n - \frac{f(x_n).f'(x_n)}{[f'(x_n)]^2 - f(x_n).f''(x_n)}$$

