数值分析

夏银华

中国科学技术大学

利用计算机求解方程 $f(x) = 0$.

两类问题:

- 计算代数/超越方程的实根，通常需要知道实根的大概位置。
- 计算代数方程的所有实/复根。

区间迭代方法(Bracketing methods)：

- 通过不断缩小区间，减少误差，找到收敛的数值解
- 方法有：二分法(Bisection) 和 试位法(False-Position, Regula Falsi)

开放迭代方法(Open methods)：

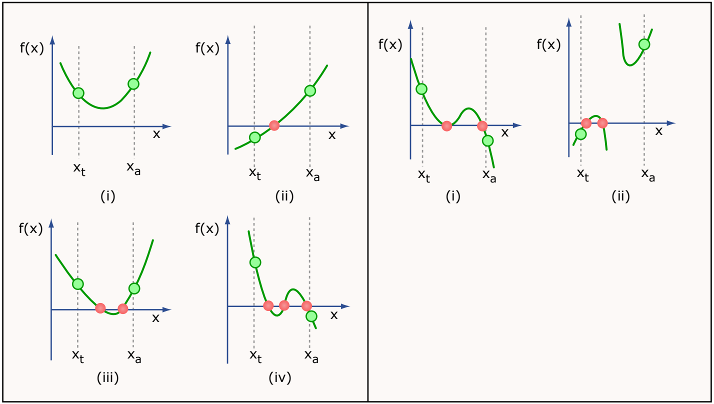
- 系统“试验及误差”方法，不需要给定区间（能从某个单值点出发求解）
- 计算更有效率，但不总是收敛
- 方法：单点迭代方法 (General method or Picard iteration), 牛顿迭代方法 (Newton-Raphson), (割线法)Secant method
- 非线性方程组的牛顿迭代方法

多项式求根

- 开放迭代方法
- 特殊迭代方法 (e.g. Muller's and Bairstow's methods)

区间迭代方法

图示：



区间迭代方法

例：平方根 (Heron's principle)

$$x^2 - a = 0 => x = \sqrt{a}$$

Heron's 方法: $x > 0$

$$x^2 - a = 0 => x = \frac{a}{x}$$

初始值

If $x_0 > \sqrt{a} \Leftrightarrow \frac{a}{x_0} < \sqrt{a}$

If $x_0 < \sqrt{a} \Leftrightarrow \frac{a}{x_0} > \sqrt{a}$

平均

$$x_1 = (x_0 + \frac{a}{x_0})/2$$

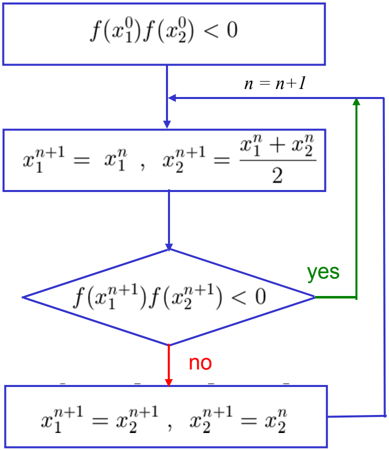
迭代公式:

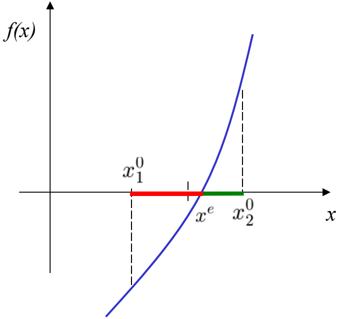
$$x_{n+1} = (x_n + \frac{a}{x_n})/2$$

区间迭代方法

二分法：

Algorithm





二分法 (Bisection)：

- 停止标准： 当误差可以接受 $e_x < \epsilon$ 或 $f(x_r) < \delta$
- 每次迭代均减少最大误差一半， 最大误差与迭代次数的关系：
$$n = \log_2(\frac{e_o}{e_d}),$$
其中 e_o 是初始误差， e_d 是期望误差。

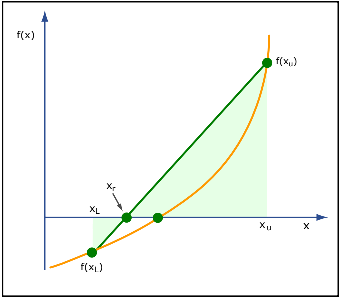
区间迭代方法

试位法 (False Position method)：

- 考虑 $f(x_L)$ 和 $f(x_U)$ 的大小

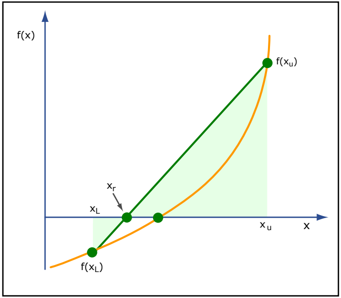
试位法 (False Position method)：

- 考虑 $f(x_L)$ 和 $f(x_U)$ 的大小



试位法 (False Position method)：

- 考虑 $f(x_L)$ 和 $f(x_U)$ 的大小



$$x_r = x_U - \frac{f(x_U)(x_L - x_U)}{f(x_L) - f(x_U)}$$

区间迭代方法

二分法与试位法：

- 均是收敛方法，但是需要减少函数求值次数。

二分法与试位法：

- 均是收敛方法，但是需要减少函数求值次数。
- 试位法的误差下降速度能够比二分法的快很多，但是当方程右端与直线相差很远时，收敛速度可能很慢。

二分法与试位法：

- 均是收敛方法，但是需要减少函数求值次数。
- 试位法的误差下降速度能够比二分法的快很多，但是当方程右端与直线相差很远时，收敛速度可能很慢。
- 改进的试位法：如果区间端点在数次迭代之后仍不变，改用二分法。

二分法与试位法：

- 均是收敛方法，但是需要减少函数求值次数。
- 试位法的误差下降速度能够比二分法的快很多，但是当方程右端与直线相差很远时，收敛速度可能很慢。
- 改进的试位法：如果区间端点在数次迭代之后仍不变，改用二分法。
- 最后一定要检测 $f(x_r)$ 离0 有多远。

开放迭代方法

不动点迭代：(General method or Picard method)

- 目标：收敛序列 $x_1, x_2, \cdots, x_n \to x^e, n \to \infty$

不动点迭代：(General method or Picard method)

- 目标：收敛序列 $x_1, x_2, \cdots, x_n \to x^e, n \to \infty$
- 重写问题 $f(x) = 0 <=> x = g(x)$, e.g.

$$g(x) = x + cf(x)$$

开放迭代方法

不动点迭代：(General method or Picard method)

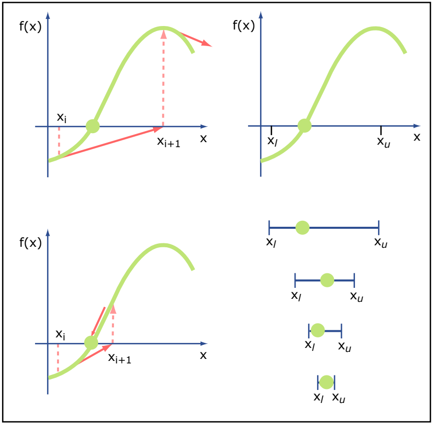
- 目标：收敛序列 $x_1, x_2, \cdots, x_n \to x^e, n \to \infty$
- 重写问题 $f(x) = 0 <=> x = g(x)$, e.g.

$$g(x) = x + cf(x)$$

- 迭代:

$$x_{n+1} = g(x_n)$$

开放迭代 VS. 区间迭代



开放迭代方法

不动点迭代：

停止迭代条件：

- 当 $x_{n+1} \neq x_n$一直继续迭代，

开放迭代方法

不动点迭代：

停止迭代条件：

- 当 $x_{n+1} \neq x_n$一直继续迭代，不现实!

不动点迭代：

停止迭代条件：

- 当 $x_{n+1} \neq x_n$一直继续迭代，不现实!
- 现实一点的停止条件：
  $|x_{n+1} - x_n| < \epsilon$ or
  $|f(x_{n+1}) - f(x_n)| < \delta$

收敛定理：

假设： $g(x)$ 满足 **Lipschitz** 条件: 即存在常数 k 使得

$$|g(x) - g(x^e)| = |g(x) - x^e| \leq k|x - x^e|, \forall x \in I$$

收敛定理：

假设： $g(x)$ 满足 **Lipschitz 条件**: 即存在常数 k 使得

$$|g(x) - g(x^e)| = |g(x) - x^e| \leq k|x - x^e|, \forall x \in I$$

则，我们得到下面的**收敛准则 (convergence criteria)**: $x_n \in I$

$$|x_n - x^e| = |g(x_{n-1}) - x^e| \leq k|x_{n-1} - x^e|$$

收敛定理：

假设： $g(x)$ 满足 **Lipschitz** 条件: 即存在常数 k 使得

$$|g(x) - g(x^e)| = |g(x) - x^e| \leq k|x - x^e|, \forall x \in I$$

则， 我们得到下面的**收敛准则 (convergence criteria)**: $x_n \in I$

$$|x_n - x^e| = |g(x_{n-1}) - x^e| \leq k|x_{n-1} - x^e|$$

因此， 我们有

$$|x_n - x^e| \leq k^n|x_0 - x^e|,$$

收敛定理：

假设： $g(x)$ 满足 **Lipschitz** 条件: 即存在常数 k 使得

$$|g(x) - g(x^e)| = |g(x) - x^e| \leq k|x - x^e|, \forall x \in I$$

则，我们得到下面的**收敛准则 (convergence criteria)**: $x_n \in I$

$$|x_n - x^e| = |g(x_{n-1}) - x^e| \leq k|x_{n-1} - x^e|$$

因此， 我们有

$$|x_n - x^e| \leq k^n|x_0 - x^e|,$$

收敛， 对 $x_0 \in I$ 和 $k < 1$。

收敛性：

假设： 如果 $g(x)$ 的导数存在，那么由平均值定理

$$g(x)-g(x^e)=g'(\xi)(x-x^e),\xi\in[x,x^e]$$

开放迭代方法

收敛性：

假设： 如果 $g(x)$ 的导数存在，那么由平均值定理

$$g(x) - g(x^e) = g'(\xi)(x - x^e), \xi \in [x, x^e]$$

则， 收敛的充分条件：

$$|g'(x)|_{x \in I} \leq k < 1$$

误差估计：

绝对误差 $|x_n - x^e|$:

$$
\begin{aligned}
|x_n - x^e| &\le |x_n - x_{n+1}| + |x_{n+1} - x^e| \\
&= |x_n - x_{n+1}| + |g(x_n) - x^e| \\
&\le |x_n - x_{n+1}| + k|x_n - x^e|
\end{aligned}
$$

误差估计：

绝对误差 $|x_n - x^e|$:

$$\begin{aligned} |x_n - x^e| &\leq |x_n - x_{n+1}| + |x_{n+1} - x^e| \\ &= |x_n - x_{n+1}| + |g(x_n) - x^e| \\ &\leq |x_n - x_{n+1}| + k|x_n - x^e| \end{aligned}$$

那么

$$|x_n - x^e| \leq \frac{1}{1-k}|x_n - x_{n+1}|$$

误差估计:

绝对误差 $|x_n - x^e|$:

$$\begin{aligned} |x_n - x^e| &\le |x_n - x_{n+1}| + |x_{n+1} - x^e| \\ &= |x_n - x_{n+1}| + |g(x_n) - x^e| \\ &\le |x_n - x_{n+1}| + k|x_n - x^e| \end{aligned}$$

那么

$$|x_n - x^e| \le \frac{1}{1-k}|x_n - x_{n+1}|$$

即

$$|x_{n+1} - x^e| \le \frac{k}{1-k}|x_n - x_{n+1}|$$

收敛速度：

- 迭代方法的收敛速度通常用 **收敛阶 (Order of Convergence)** 来衡量。

收敛速度：

- 迭代方法的收敛速度通常用 **收敛阶 (Order of Convergence)** 来衡量。
- 考虑数列 $x_0, x_1, \cdots$ 和误差 $e_n = x_n - x^e$。如果存在数p 和常数 $C \neq 0$ 使得

$$\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^p} = C$$

那么 p 就被定义为 **收敛阶 (Order of Convergence)** 或 **收敛指数 (Convergence exponent)**，称C为渐进常数。

收敛速度：

- 迭代方法的收敛速度通常用 **收敛阶 (Order of Convergence)** 来衡量。
- 考虑数列 $x_0, x_1, \cdots$ 和误差 $e_n = x_n - x^e$。如果存在数p 和常数 $C \neq 0$ 使得
$$\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^p} = C$$
那么 p 就被定义为 **收敛阶 (Order of Convergence)** 或 **收敛指数 (Convergence exponent)**，称C为渐进常数。
- 固定点迭代：通常只是线性收敛$(p = 1)$。

牛顿 (Newton-Raphson) 迭代方法

- 求解$f(x) = 0$的迭代格式也可以写成

$$x_{n+1} = g(x_n) = x_n + h(x_n)f(x_n)$$

牛顿 (Newton-Raphson) 迭代方法

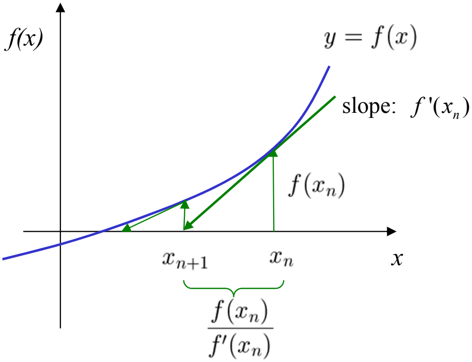
- 求解$f(x)=0$的迭代格式也可以写成

$$x_{n+1}=g(x_n)=x_n+h(x_n)f(x_n)$$

- Newton-Raphson

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

牛顿 (Newton-Raphson) 迭代方法：图示



牛顿 (Newton-Raphson) 迭代方法：收敛性

收敛准则：

$$|x_n - x^e| = |g(x_{n-1}) - x^e| \leq |g'(\xi)||x_{n-1} - x^e|$$

牛顿 (Newton-Raphson) 迭代方法：收敛性

收敛准则：

$$|x_n - x^e| = |g(x_{n-1}) - x^e| \leq |g'(\xi)||x_{n-1} - x^e|$$

当 $g'(\xi) \approx 0$，收敛迅速

$$\begin{aligned} g(x) &= x + h(x)f(x) \\ g'(x) &= 1 + h'(x)f(x) + h(x)f'(x) \\ &\approx 1 + h(x)f'(x) \end{aligned}$$

牛顿 (Newton-Raphson) 迭代方法：收敛性

收敛准则：

$$|x_n - x^e| = |g(x_{n-1}) - x^e| \leq |g'(\xi)||x_{n-1} - x^e|$$

当 $g'(\xi) \approx 0$，收敛迅速

$$\begin{aligned} g(x) &= x + h(x)f(x) \\ g'(x) &= 1 + h'(x)f(x) + h(x)f'(x) \\ &\approx 1 + h(x)f'(x) \end{aligned}$$

因此，选择

$$h(x) = -\frac{1}{f'(x)}$$

牛顿 (Newton-Raphson) 迭代方法：例

平方根：

$$f(x) = x^2 - a$$

$$f'(x) = 2x$$

Newton-Raphson

$$x_{n+1} = x_n - \frac{x_n^2 - a}{2x_n} = \frac{1}{2}(x_n + \frac{a}{x_n})$$

牛顿 (Newton-Raphson) 迭代方法：例

平方根：

$$f(x) = x^2 - a$$

$$f'(x) = 2x$$

Newton-Raphson

$$x_{n+1} = x_n - \frac{x_n^2 - a}{2x_n} = \frac{1}{2}(x_n + \frac{a}{x_n})$$

即 **Heron's formula**

牛顿 (Newton-Raphson) 迭代方法：例

除法：

$$x = \frac{1}{a}$$

$$f(x) = ax - 1$$

$$f'(x) = a$$

$$g(x) = x - \frac{ax-1}{a} = x - x^e(ax-1) \approx x - x(ax-1)$$

牛顿 (Newton-Raphson) 迭代方法：例

除法：

$$x = \frac{1}{a}$$

$$f(x) = ax - 1$$

$$f'(x) = a$$

$$g(x) = x - \frac{ax-1}{a} = x - x^e(ax-1) \approx x - x(ax-1)$$

Newton-Raphson

$$x_{n+1} = x_n - x_n(ax_n - 1)$$

牛顿 (Newton-Raphson) 迭代方法：收敛速度

定义：

$$e_n = x_n - x^e$$

牛顿 (Newton-Raphson) 迭代方法：收敛速度

定义：

$$e_n = x_n - x^e$$

Taylor展开：

$$g(x_n) = g(x^e) + e_n g'(x^e) + \frac{1}{2} e_n^2 g''(x^e) + \cdots$$

牛顿 (Newton-Raphson) 迭代方法：收敛速度

定义：

$$e_n = x_n - x^e$$

Taylor展开：

$$g(x_n) = g(x^e) + e_n g'(x^e) + \frac{1}{2} e_n^2 g''(x^e) + \cdots$$

由于 $g'(x^e) = 0$

$$g(x_n) - g(x^e) \approx \frac{1}{2} e_n^2 g''(x^e)$$

牛顿 (Newton-Raphson) 迭代方法：收敛速度

定义：

$$e_n = x_n - x^e$$

Taylor展开：

$$g(x_n) = g(x^e) + e_n g'(x^e) + \frac{1}{2} e_n^2 g''(x^e) + \cdots$$

由于 $g'(x^e) = 0$

$$g(x_n) - g(x^e) \approx \frac{1}{2} e_n^2 g''(x^e)$$

$$e_{n+1} = x_{n+1} - x^e \approx \frac{1}{2} e_n^2 g''(x^e)$$

牛顿 (Newton-Raphson) 迭代方法：收敛速度

定义：

$$e_n = x_n - x^e$$

Taylor展开：

$$g(x_n) = g(x^e) + e_n g'(x^e) + \frac{1}{2} e_n^2 g''(x^e) + \cdots$$

由于 $g'(x^e) = 0$

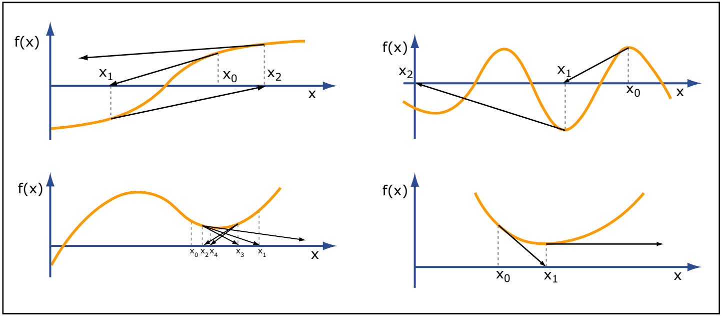
$$g(x_n) - g(x^e) \approx \frac{1}{2} e_n^2 g''(x^e)$$

$$e_{n+1} = x_{n+1} - x^e \approx \frac{1}{2} e_n^2 g''(x^e)$$

二阶 (quadratic) 收敛。

牛顿 (Newton-Raphson) 迭代方法：注意事项

- 零点为弯曲点 (inflection point)， i.e. $f''(x^e) = 0$
- 迭代可能会在局部极值点附近震荡
- 遇到斜率接近0的情形
- 零点处斜率也是零



二分法虽然没有牛顿迭代法效率高，但会被经常用来用来缩小区间作初值。

牛顿 (Newton-Raphson) 迭代方法：多重根

p重根

$$f(x) = (x - x^e)^p f_1(x), f_1(x^e) \neq 0$$

牛顿 (Newton-Raphson) 迭代方法：多重根

p重根

$$f(x) = (x - x^e)^p f_1(x), f_1(x^e) \neq 0$$

Newton-Raphson

$$x_{n+1} = g(x_n) = x_n - \frac{(x_n - x^e)^p f_1(x_n)}{p(x_n - x^e)^{p-1} f_1(x_n) + (x_n - x^e)^p f_1'(x_n)}$$

即

$$x_{n+1} = g(x_n) = x_n - \frac{(x_n - x^e) f_1(x_n)}{p f_1(x_n) + (x_n - x^e) f_1'(x_n)}$$

牛顿 (Newton-Raphson) 迭代方法：多重根

p重根

$$f(x) = (x - x^e)^p f_1(x), f_1(x^e) \neq 0$$

Newton-Raphson

$$x_{n+1} = g(x_n) = x_n - \frac{(x_n - x^e)^p f_1(x_n)}{p(x_n - x^e)^{p-1} f_1(x_n) + (x_n - x^e)^p f_1'(x_n)}$$

即

$$x_{n+1} = g(x_n) = x_n - \frac{(x_n - x^e) f_1(x_n)}{p f_1(x_n) + (x_n - x^e) f_1'(x_n)}$$

$$g'(x^e) = 1 - \frac{1}{p}$$

牛顿 (Newton-Raphson) 迭代方法：多重根

p重根

$$f(x) = (x - x^e)^p f_1(x), f_1(x^e) \neq 0$$

Newton-Raphson

$$x_{n+1} = g(x_n) = x_n - \frac{(x_n - x^e)^p f_1(x_n)}{p(x_n - x^e)^{p-1} f_1(x_n) + (x_n - x^e)^p f_1'(x_n)}$$

即

$$x_{n+1} = g(x_n) = x_n - \frac{(x_n - x^e) f_1(x_n)}{p f_1(x_n) + (x_n - x^e) f_1'(x_n)}$$

$$g'(x^e) = 1 - \frac{1}{p}$$

二阶 (quadratic) 收敛。

割线法(Secant method)：

- 在 Newton-Raphson法中，每一步需要做两次函数求值 $f(x_n)$, $f'(x_n)$
- $f(x)$ 有时没有解析表达式

割线法(Secant method)：

- 在 Newton-Raphson法中，每一步需要做两次函数求值 $f(x_n)$, $f'(x_n)$
- $f(x)$ 有时没有解析表达式

近似导数：

$$f'(x) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

割线法：

- 迭代：

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

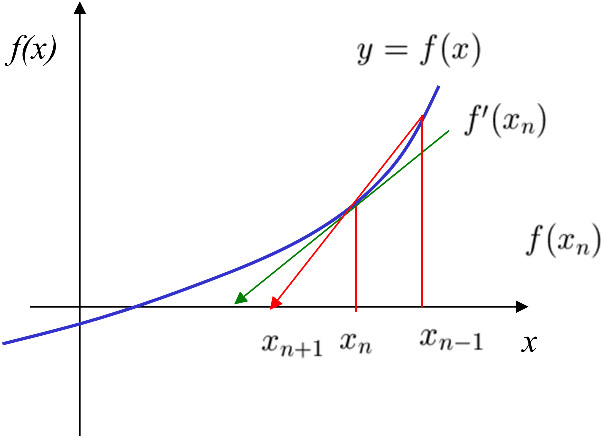
割线法：

- 迭代：

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

- 每一步迭代只有一次函数求值 $f(x_n)$

割线法：图示



割线法：收敛速度

- 绝对误差： $e_n = x_n - x^e$

$$e_{n+1} = x_{n+1} - x^e = \frac{f(e_n + x^e)(e_{n-1} + x^e) - f(e_{n-1} + x^e)(e_n + x^e)}{f(e_n + x^e) - f(e_{n-1} + x^e)}$$

割线法：收敛速度

- 绝对误差： $e_n = x_n - x^e$

$$e_{n+1} = x_{n+1} - x^e = \frac{f(e_n + x^e)(e_{n-1} + x^e) - f(e_{n-1} + x^e)(e_n + x^e)}{f(e_n + x^e) - f(e_{n-1} + x^e)}$$

- 利用Taylor展开：
  绝对误差 $e_{n+1} \approx \frac{1}{2} e_n e_{n-1} \frac{f''(x^e)}{f'(x^e)}$
  相对误差 $\frac{e_{n+1}}{|x^e|} \approx \frac{1}{2} \frac{e_n}{|x^e|} \frac{e_{n-1}}{|x^e|} \frac{f''(x^e)}{f'(x^e)} |x^e|$

割线法：收敛速度

由定义

$$e_n = A(x^e)e_{n-1}^m$$

割线法：收敛速度

由定义

$$e_n = A(x^e)e_{n-1}^m$$

得到

$$e_{n+1} = B(x^e)e_n^{1+1/m}$$

割线法：收敛速度

由定义

$$e_n = A(x^e)e_{n-1}^m$$

得到

$$e_{n+1} = B(x^e)e_n^{1+1/m}$$

即

$$1 + \frac{1}{m} = m => m = \frac{1}{2}(1 + \sqrt{5}) \approx 1.62$$

压缩映射：

- 若$g(s) = s$，称s为函数g 的**不动点 (fixed point)**

压缩映射：

- 若$g(s) = s$，称s为函数g 的**不动点 (fixed point)**
- 若存在 $k < 1$ 使得 $|g(x) - g(y)| \leq k|x, y|, \quad \forall x, y \in I$，其中$I \in \mathcal{R}$ 为闭区间，则称函数 $g : I \to I$ 是**压缩 (contractive)**的。

压缩映射定理：

如果函数g是闭区间 I 上的**压缩映射**，则g在I上有**唯一**不动点，并且从任意点x_0出发迭代 $x_{n+1} = g(x_n)$均可得到该不动点。

高阶收敛迭代方法：

- 假设压缩映射 $g(x)$ 满足
  $g^{(k)}(x^e) = 0, \quad 0 \le k < q, \quad g^{(q)}(x^e) \neq 0$

高阶收敛迭代方法：

- 假设压缩映射 $g(x)$ 满足
  $g^{(k)}(x^e) = 0, \quad 0 \leq k < q, \quad g^{(q)}(x^e) \neq 0$
- 定义 $e_n = x_n - x^e$，由Taylor展开,

$$
\begin{aligned}
e_{n+1} &= x_{n+1} - x^e = g(x_n) - g(x^e) \\
&= g(x^e + e_n) - g(x^e) \\
&= \left(g(x^e) + g(x^e)e_n + \frac{1}{2}e_n^2 g(x^e) + \cdots \right) - g(x^e) \\
&= \frac{1}{q!} e_n^q g^{(q)}(\xi_n)
\end{aligned}
$$

高阶收敛迭代方法：

- 假设压缩映射 $g(x)$ 满足
  $g^{(k)}(x^e)=0, \quad 0 \le k < q, \quad g^{(q)}(x^e) \ne 0$
- 定义 $e_n = x_n - x^e$，由Taylor展开，

$$
\begin{aligned}
e_{n+1} &= x_{n+1} - x^e = g(x_n) - g(x^e) \\
&= g(x^e + e_n) - g(x^e) \\
&= \left(g(x^e) + g(x^e)e_n + \frac{1}{2}e_n^2 g(x^e) + \cdots \right) - g(x^e) \\
&= \frac{1}{q!} e_n^q g^{(q)}(\xi_n)
\end{aligned}
$$

- $\lim\limits_{n\to\infty} \frac{e_{n+1}}{e_n^q} = \frac{1}{q!} g^{(q)}(\xi_n)$

多项式零点（根）

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

多项式零点（根）

$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$

多项式零点存在定理：

- 非常数多项式在复平面上至少存在一个零点

多项式零点（根）

$$p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$

多项式零点存在定理：

- 非常数多项式在复平面上至少存在一个零点
- n次多项式在复平面上存在n个零点(包括重数)

多项式零点

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$

多项式零点范围：

- （上界）$\rho_p = 1 + |a_n|^{-1} \max_{0 \le k < n} |a_k|$

多项式零点

$p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$

多项式零点范围：

- （上界）$\rho_p = 1 + |a_n|^{-1} \max\limits_{0\le k<n} |a_k|$
- （下界）ρ_s^{-1}，其中ρ_s 是多项式$s(x) = x^n p(\frac{1}{x})$ 的零点上界。

多项式零点

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

多项式零点

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

Horner's 算法：

-

$$\begin{aligned}
b_{n-1} &= a_n \\
b_{n-2} &= a_{n-1} + x_0 b_{n-1} \\
&\cdots \\
b_0 &= a_1 + x_0 b_1 \\
p(x_0) &= a_0 + x_0 b_0
\end{aligned}$$

多项式零点

$p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$

Horner's 算法：

- $$\begin{aligned} b_{n-1} &= a_n \\ b_{n-2} &= a_{n-1} + x_0 b_{n-1} \\ &\cdots \\ b_0 &= a_1 + x_0 b_1 \\ p(x_0) &= a_0 + x_0 b_0 \end{aligned}$$

- 定义$q(x) = b_{n-1}x^{n-1} + \cdots + b_1 x + b_0$，有$q(x) = \frac{p(x)-p(x_0)}{x-x_0}$

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$

牛顿迭代求根算法：

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

多项式零点

$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$

牛顿迭代求根算法：

- $x_{k+1} = x_k - \frac{p(x_k)}{p'(x_k)}$。
- 每一步需要利用 Horner's 算法 对$p(x_k)$ 和 $p'(x_k)$求值。

多项式零点

$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$

牛顿迭代求根算法：

- $x_{k+1} = x_k - \frac{p(x_k)}{p'(x_k)}$。
- 每一步需要利用 Horner's 算法 对$p(x_k)$ 和 $p'(x_k)$求值。
- $p'(x_k) = q_{x_k}(x_k)$

多项式零点

$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$

牛顿迭代求根算法：

- $x_{k+1} = x_k - \frac{p(x_k)}{p'(x_k)}$。
- 每一步需要利用 Horner's 算法 对$p(x_k)$ 和 $p'(x_k)$求值。
- $p'(x_k) = q_{x_k}(x_k)$
- 在x_k为圆心，$n|x_k - x_{k+1}|$为半径的圆盘上一定有$p(x)$的一个零点。

多项式零点

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

多项式零点

$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$

Bairstow's 方法：

- 假设 a_k， $0 \le k \le n$均为实数，可能存在复根。

$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$

Bairstow's 方法：

- 假设 a_k， $0 \le k \le n$均为实数，可能存在复根。
- 则如果复数w是多项式$p(x)$的根，那么$\overline{w}$也是根。

$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$

Bairstow's 方法：

- 假设 a_k， $0 \le k \le n$均为实数，可能存在复根。
- 则如果复数w是多项式$p(x)$的根，那么$\overline{w}$也是根。
- $p(x) = (x - w)(x - \overline{w})q(x)$

多项式零点

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$

Bairstow's 方法：

- 假设 a_k， $0 \leq k \leq n$均为实数，可能存在复根。

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

Bairstow's 方法：

- 假设 $a_k, \quad 0 \le k \le n$均为实数，可能存在复根。
- $p(x) = q(x)(x^2 - ux - v) + r(x)$，其中

$$\begin{aligned} q(x) &= b_n x^{n-2} + \cdots + b_3 x + b_2, \\ r(x) &= b_1(z-u) + b_0, \\ b_k &= a_k + u b_{k+1} + v b_{k+2}, \quad k = n, \cdots 0, \text{ and } b_{n+1} = b_{n+2} = 0. \end{aligned}$$

多项式零点

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

$p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$

Bairstow's 方法：

- 寻找 (u,v) 使得 $b_0(u,v) = b_1(u,v) = 0$

$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$

Bairstow's 方法：

- 寻找 (u, v) 使得 $b_0(u, v) = b_1(u, v) = 0$
- 利用牛顿迭代法求解**非线性方程组**

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

Bairstow's 方法：

- 寻找 (u, v) 使得 $b_0(u, v) = b_1(u, v) = 0$
- 利用牛顿迭代法求解**非线性方程组**
- $\vec{u}_{n+1} = \vec{u}_n - J^{-1}\vec{b}(\vec{u})$，其中

$$\vec{u} = \begin{pmatrix} u \\ v \end{pmatrix}, \quad \vec{b}(\vec{u}) = \begin{pmatrix} b_0(u, v) \\ b_1(u, v) \end{pmatrix}, \quad J = \begin{pmatrix} \partial_u b_0(u, v) & \partial_v b_0(u, v) \\ \partial_u b_1(u, v) & \partial_v b_1(u, v) \end{pmatrix}$$

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

Bairstow's 方法：

- 寻找 (u, v) 使得 $b_0(u,v) = b_1(u,v) = 0$
- 利用牛顿迭代法求解**非线性方程组**
- $\vec{u}_{n+1} = \vec{u}_n - J^{-1}\vec{b}(\vec{u})$，其中

$$\vec{u} = \begin{pmatrix} u \\ v \end{pmatrix}, \quad \vec{b}(\vec{u}) = \begin{pmatrix} b_0(u,v) \\ b_1(u,v) \end{pmatrix}, \quad J = \begin{pmatrix} \partial_u b_0(u,v) & \partial_v b_0(u,v) \\ \partial_u b_1(u,v) & \partial_v b_1(u,v) \end{pmatrix}$$

- $|J| \neq 0$

高阶方法：

- Laguerre 方法
- Halley 方法
- Muller 方法
- $\cdots$

初值选取：同伦法 (Homotopy)

- 选择 $\boldsymbol{h}(t, \boldsymbol{x}) = 0$ 使得 $\boldsymbol{h}(0, \boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}_0)$ $\boldsymbol{h}(1, \boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x})$，例如：

$$
\begin{aligned}
\boldsymbol{h}(t, \boldsymbol{x}) &= t\boldsymbol{f}(\boldsymbol{x}) + (1-t)(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}_0)) \\
&= \boldsymbol{f}(\boldsymbol{x}) + (t-1)\boldsymbol{f}(\boldsymbol{x}_0)
\end{aligned}
$$

初值选取：同伦法 (Homotopy)

- 选择 $\boldsymbol{h}(t, \boldsymbol{x}) = 0$ 使得 $\boldsymbol{h}(0, \boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}_0)$ $\boldsymbol{h}(1, \boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x})$，例如：

$$\begin{aligned}\boldsymbol{h}(t, \boldsymbol{x}) &= t\boldsymbol{f}(\boldsymbol{x}) + (1-t)(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}_0)) \\ &= \boldsymbol{f}(\boldsymbol{x}) + (t-1)\boldsymbol{f}(\boldsymbol{x}_0)\end{aligned}$$

- $\boldsymbol{h}(t, \boldsymbol{x}(t)) = 0$

初值选取：同伦法 (Homotopy)

- 选择 $\boldsymbol{h}(t,\boldsymbol{x})=0$ 使得 $\boldsymbol{h}(0,\boldsymbol{x})=\boldsymbol{f}(\boldsymbol{x}_0)$ $\boldsymbol{h}(1,\boldsymbol{x})=\boldsymbol{f}(\boldsymbol{x})$，例如：

$$\begin{aligned}\boldsymbol{h}(t,\boldsymbol{x}) &= t\boldsymbol{f}(\boldsymbol{x})+(1-t)(\boldsymbol{f}(\boldsymbol{x})-\boldsymbol{f}(\boldsymbol{x}_0)) \\ &= \boldsymbol{f}(\boldsymbol{x})+(t-1)\boldsymbol{f}(\boldsymbol{x}_0)\end{aligned}$$

- $\boldsymbol{h}(t,\boldsymbol{x}(t))=0$

$$\begin{cases}\boldsymbol{x}'(t)=-\boldsymbol{h}_{\boldsymbol{x}}(t,\boldsymbol{x}(t))^{-1}\boldsymbol{h}_t(t,\boldsymbol{x}(t)), \\ \boldsymbol{x}(0)=\boldsymbol{x}_0.\end{cases}$$

初值选取：同伦法 (Homotopy)

- $\boldsymbol{h}(t(s), \boldsymbol{x}(s)) = 0$, 令 $\boldsymbol{y}(s) = (t, \boldsymbol{x}(s))^T$，则有 $\boldsymbol{h}(\boldsymbol{y}(s)) = 0$

牛顿迭代方法

初值选取：同伦法 (Homotopy)

- $\boldsymbol{h}(t(s), \boldsymbol{x}(s)) = 0$, 令 $\boldsymbol{y}(s) = (t, \boldsymbol{x}(s))^T$，则有 $\boldsymbol{h}(\boldsymbol{y}(s)) = 0$

$$\boldsymbol{h}'(\boldsymbol{y})\boldsymbol{y}'(s) = 0,$$

初值选取：同伦法 (Homotopy)

- $\boldsymbol{h}(t(s), \boldsymbol{x}(s)) = 0$, 令 $\boldsymbol{y}(s) = (t, \boldsymbol{x}(s))^T$，则有 $\boldsymbol{h}(\boldsymbol{y}(s)) = 0$

$$\boldsymbol{h}'(\boldsymbol{y})\boldsymbol{y}'(s) = 0,$$

$$\begin{cases} y_j'(t) = (-1)^j |A_j|, \\ \boldsymbol{y}(0) = \begin{pmatrix} 0 \\ \boldsymbol{x}_0 \end{pmatrix}, \end{cases}$$

其中 A_j 为 $\boldsymbol{h}'(\boldsymbol{y})$ 除去第 j 列。