# 数值计算方法

## -求解非线性方程

September 14, 2016

# 本节概要

# 求解非线性方程概述

非线性科学是当今科学发展的一个重要方向，而求解非线性方程是现代科学中不可或缺的内容。我们通常利用数值方法求满足指定精度要求的近似解。

**Definition 1 (函数的根)**

我们通常称函数 $f(x)$ 在 $x = r$ 处有根如果 $f(r) = 0$。

**Remark 1.** *习惯上说，多项式的零点称为根，在求解非线性方程时，我们也把一般函数的零点称为根。*

# 确定有根区间

解方程的第一步是确定有根的区间，这里我们可以利用连续函数的中值定理来寻找这个区间，具体来说我们有

**Theorem 2**

*设 $f$ 是 $[a,b]$ 上的连续函数且满足 $f(a)f(b)<0$。则 $f$ 在 $[a,b]$ 上有根，即存在 $r\in[a,b]$ 使得 $f(r)=0$。*

定理可以直接利用连续函数中值定理证明，并可以由下图直观的表现。

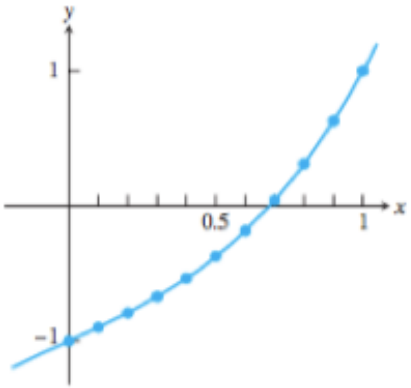


Figure 1: $f(x)=x^3+x-1$ 的图像

# 二分法

虽然画图可以让我们大致观察到根所在的区间，但是一般我们并不能够用这样的方法来确定根所在的具体位置，一般来说我们利用二分法这种算法来寻找根。

**Algorithm 2 (二分法伪代码).**

*利用条件 $f(a)f(b)$ 确定有根的区间 $[a,b]$，确定精度 $TOL$。*

while $\frac{(b-a)}{2} > TOL$
  $c = \frac{a+b}{2}$
  if $f(c) = 0$
    停机，结束程序输出 $c$
  if $f(a)f(c) < 0$
    $b = c$
  else
    $a = c$
  end
end

最终的区间 $[a,b]$ 必定含有根，并输出 $\frac{a+b}{2}$ 作为根的近似。

# 二分法应用举例

**Example 3**

在区间 $[0,1]$ 上利用二分法寻找函数 $f(x)=x^3+x-1$ 的根。

我们注意到 $f(0)f(1)=(-1)(1)<0$，因此在 $[0,1]$ 上必定有根。之后的计算可以用下表来表示并可以在 Matlab 中通过调节 $TOL$ 的值将根所在的区间长度缩小。

| $i$ | $a_i$ | $f(a_i)$ | $c_i$ | $f(c_i)$ | $b_i$ | $f(b_i)$ |
|---|---|---|---|---|---|---|
| 0 | 0.0000 | − | 0.5000 | − | 1.0000 | + |
| 1 | 0.5000 | − | 0.7500 | + | 1.0000 | + |
| 2 | 0.5000 | − | 0.6250 | − | 0.7500 | + |
| 3 | 0.6250 | − | 0.6875 | + | 0.7500 | + |
| 4 | 0.6250 | − | 0.6562 | − | 0.6875 | + |
| 5 | 0.6562 | − | 0.6719 | − | 0.6875 | + |
| 6 | 0.6719 | − | 0.6797 | − | 0.6875 | + |
| 7 | 0.6797 | − | 0.6836 | + | 0.6875 | + |
| 8 | 0.6797 | − | 0.6816 | − | 0.6836 | + |
| 9 | 0.6816 | − | 0.6826 | + | 0.6836 | + |

Figure 2: 利用二分法在 $[0,1]$ 上寻找 $f(x)=x^3+x-1$ 的计算过程

# 二分法的误差

如果 $[a,b]$ 是初始的搜索区间，则 $n$ 次二分之后，搜索区间缩小为了 $\frac{b-a}{2^n}$，设 $n$ 次二分后的区间为 $[a_n,b_n]$，则 $x_c=\frac{a_n+b_n}{2}$ 作为根 $r$ 的近似。我们发现，由于搜索区间缩小为了 $\frac{b-a}{2^n}$，所以误差有以下的上界

$$\text{解的误差}=|x_c-r|<\frac{b-a}{2^{n+1}}. \tag{1}$$

同时，此时函数求值的次数为 $n+2$ 次。

**Definition 4**

如果解的误差小于 $0.5\times10^{-p}$，则我们称方程的解精确到小数点后 $p$ 位。

# 二分法的误差

**Example 5**

利用二分法寻找 $f(x) = \cos x - x$ 在 $[0, 1]$ 上的根，并要求精确到小数点后六位。

由于在 $n$ 次二分后，近似解的误差上界满足 $\frac{b-a}{2^{n+1}} = \frac{1}{2^{n+1}}$，利用精确到小数点后 $n$ 位的定义，我们有

$$\frac{1}{2^{n+1}} < 0.5 \times 10^{-6}$$

$$n > \frac{6}{\log_{10} 2} \approx \frac{6}{0.301} = 19.9.$$

也就是说 20 步之后我们将会达到所要求的精度。

# 二分法的误差

计算过程可见下表

| $k$ | $a_k$ | $f(a_k)$ | $c_k$ | $f(c_k)$ | $b_k$ | $f(b_k)$ |
|---|---|---|---|---|---|---|
| 0 | 0.000000 | + | 0.500000 | + | 1.000000 | − |
| 1 | 0.500000 | + | 0.750000 | − | 1.000000 | − |
| 2 | 0.500000 | + | 0.625000 | + | 0.750000 | − |
| 3 | 0.625000 | + | 0.687500 | + | 0.750000 | − |
| 4 | 0.687500 | + | 0.718750 | + | 0.750000 | − |
| 5 | 0.718750 | + | 0.734375 | + | 0.750000 | − |
| 6 | 0.734375 | + | 0.742188 | − | 0.750000 | − |
| 7 | 0.734375 | + | 0.738281 | + | 0.742188 | − |
| 8 | 0.738281 | + | 0.740234 | − | 0.742188 | − |
| 9 | 0.738281 | + | 0.739258 | − | 0.740234 | − |
| 10 | 0.738281 | + | 0.738770 | + | 0.739258 | − |
| 11 | 0.738769 | + | 0.739014 | + | 0.739258 | − |
| 12 | 0.739013 | + | 0.739136 | − | 0.739258 | − |
| 13 | 0.739013 | + | 0.739075 | + | 0.739136 | − |
| 14 | 0.739074 | + | 0.739105 | − | 0.739136 | − |
| 15 | 0.739074 | + | 0.739090 | − | 0.739105 | − |
| 16 | 0.739074 | + | 0.739082 | + | 0.739090 | − |
| 17 | 0.739082 | + | 0.739086 | − | 0.739090 | − |
| 18 | 0.739082 | + | 0.739084 | + | 0.739086 | − |
| 19 | 0.739084 | + | 0.739085 | − | 0.739086 | − |
| 20 | 0.739084 | + | 0.739085 | − | 0.739085 | − |

Figure 3: 二分法求 $f(x) = \cos x - x$ 的根的计算过程

精确到六位小数的近似解为 0.739085.

# 等距离扫描法

在确定有根区间时，如果函数比较简单，可以用画图的办法进行确定，但是如果函数比较复杂，那么进行一次求值需要的计算量比较大，而画图又需要大量的求值，那么画图法就不是一个好的方法。这里介绍一种等步长扫描法寻找有根的区间，具体算法如下。

**Algorithm 3 (等距离扫描法寻找有根区间).**
$h > 0$ 为给定步长，$b$ 为搜索上限，$h_m$ 为最小步长。
while $h \geq h_m$
取 $x_0 = a, x_1 = a + h$。
  while $f(x_0)f(x_1) > 0$ 且 $x_1 \leq b$
    令 $x_0 = x_1$，$x_1 = x_1 + h$。
  end
  if $x_1 < b$　　搜索成功输出 $[x_0, x_1]$ 作为有根区间。
  else
    令 $h = \frac{h}{2}$。
  end
end

# 不动点

**Definition 6 (不动点)**

如果实数 $r$ 满足 $g(r) = r$，则 $r$ 被称为函数 $g(x)$ 不动点。

例如 $f(x) = \cos x - x$ 的根就是 $g(x) = \cos x$ 的不动点，因为 $g(x) = \cos x = x$ 等价于 $f(x) = \cos x - x = 0$。

**Definition 7 (不动点迭代)**

当我们将 $f(x) = 0$ 等价的写为 $g(x) = x$ 后，不动点迭代就是将一个初始猜测 $x_0$ 代入 $g(x)$，并且对 $g$ 进行迭代

$$x_{i+1} = g(x_i), i = 0, 1, 2, \ldots \tag{2}$$

进而得到数列 $\{x_i\}_{i=0}^{\infty}$。

# 不动点迭代序列与不动点

通过不动点迭代的得到的数列并不一定能保证收敛，但是根据上一章的连续极限定理，如果不动点迭代得到的序列是收敛的，则这个收敛的序列必定收敛于 $g(x)$ 的不动点，这是因为

$$r = \lim_{i\to\infty} x_{i+1} = \lim_{i\to\infty} g(x_i) = g(\lim_{i\to\infty} x_i) = g(r), \tag{3}$$

其中第三个等号我们用到了连续收敛定理。

以上式子能够成立的前提是迭代序列收敛。对于寻找同一个函数 $f(x)$ 的根，我们可以构造出很多不同的迭代函数 $g(x)$，但并不是每一个迭代函数 $g(x)$ 都会得到收敛序列。我们通过以下的例子来说明这一点。

# 不动点迭代的收敛性

**Example 8**

将求解 $f(x) = x^3 + x - 1 = 0$ 的问题换为不同的不动点迭代问题，并进行迭代。

我们选择以下几种方法将 $f(x) = 0$ 转化为 $g(x) = x$。

1. 第一种方法
$$g_1(x) = 1 - x^3. \tag{4}$$
2. 第二种方法
$$g_2(x) = (1 - x)^{\frac{1}{3}}. \tag{5}$$
3. 第三种方法
$$g_3(x) = \frac{1 + 2x^3}{1 + 3x^2}. \tag{6}$$

# 不动点迭代的收敛性

对于三个不同的迭代函数，如果选择初始猜测为 $x_0 = 0.5$，则迭代的结果如下面几个表所示

| $i$ | $x_i$ |
|---|---|
| 0 | 0.50000000 |
| 1 | 0.87500000 |
| 2 | 0.33007813 |
| 3 | 0.96403747 |
| 4 | 0.10405419 |
| 5 | 0.99887338 |
| 6 | 0.00337606 |
| 7 | 0.99999996 |
| 8 | 0.00000012 |
| 9 | 1.00000000 |
| 10 | 0.00000000 |
| 11 | 1.00000000 |
| 12 | 0.00000000 |

Figure 4: 不动点迭代 $g_1(x) = 1 - x^3$ 的计算结果

# 不动点迭代的收敛性

| $i$ | $x_i$ |
|---|---|
| 0 | 0.50000000 |
| 1 | 0.79370053 |
| 2 | 0.59088011 |
| 3 | 0.74236393 |
| 4 | 0.63631020 |
| 5 | 0.71380081 |
| 6 | 0.65900615 |
| 7 | 0.69863261 |
| 8 | 0.67044850 |
| 9 | 0.69072912 |
| 10 | 0.67625892 |
| 11 | 0.68664554 |
| 12 | 0.67922234 |

| $i$ | $x_i$ |
|---|---|
| 13 | 0.68454401 |
| 14 | 0.68073737 |
| 15 | 0.68346460 |
| 16 | 0.68151292 |
| 17 | 0.68291073 |
| 18 | 0.68191019 |
| 19 | 0.68262667 |
| 20 | 0.68211376 |
| 21 | 0.68248102 |
| 22 | 0.68221809 |
| 23 | 0.68240635 |
| 24 | 0.68227157 |
| 25 | 0.68236807 |

Figure 5: 不动点迭代 $g_2(x) = (1-x)^{\frac{1}{3}}$ 的计算结果

# 不动点迭代的收敛性

| $i$ | $x_i$ |
|---|---|
| 0 | 0.50000000 |
| 1 | 0.71428571 |
| 2 | 0.68317972 |
| 3 | 0.68232842 |
| 4 | 0.68232780 |
| 5 | 0.68232780 |
| 6 | 0.68232780 |
| 7 | 0.68232780 |

Figure 6: 不动点迭代 $g_3(x) = \frac{1+2x^3}{1+3x^2}$ 的计算结果

可以看到，迭代函数 $g_1$ 无法生成收敛的序列，迭代函数 $g_2$ 通过 20 步左右的迭代值稳定在 0.682 附近，而迭代函数 $g_3$ 则通过 4 次迭代就将迭代序列的值稳定在了 0.68232780，即小数点后 8 位。可以看到迭代函数选取不同迭代效果差别非常大。

# 不动点迭代的几何解释

以上几个不动点迭代的结果可以用以下图片更直观的解释。

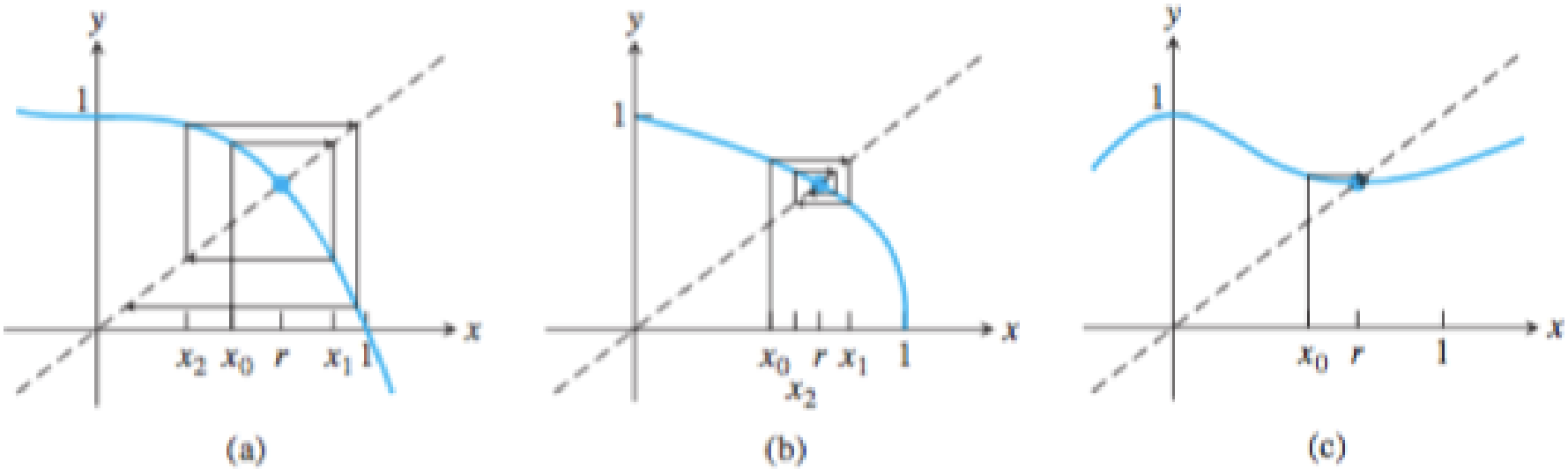


Figure 7: 不动点迭代的几何意义，(a) 对应 $g_1$，(b) 对应 $g_2$，(c) 对应 $g_3$

# 不动点迭代的线性收敛性

不动点迭代的收敛性可以由两个线性函数的不动点迭代更清楚的表现出来。我们设

$$g_1(x) = -\frac{3}{2}x + \frac{5}{2}, \quad g_2(x) = -\frac{1}{2}x + \frac{3}{2}. \tag{7}$$

容易得到，$x = 1$ 是这两个函数的不动点。但是下图可以看到两个不动点迭代的区别。

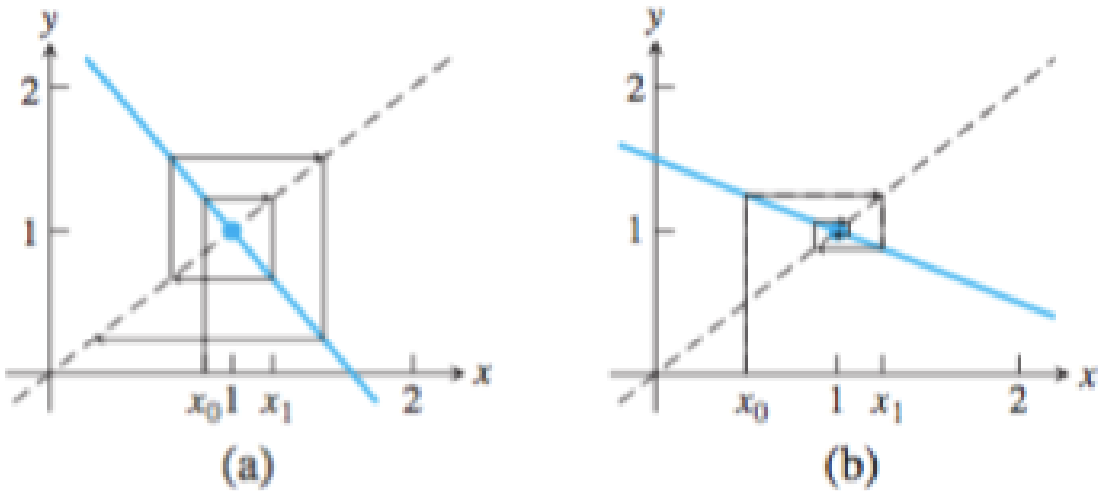


Figure 8: 不动点迭代的几何解释，(a) 对应 $g_1$，(b) 对应 $g_2$

# 不动点迭代的线性收敛性

事实上，两个迭代函数最大的区别是函数的斜率，即 $g'(x)$ 的值。从方程的角度看，$r = 1$ 是两个函数的不动点，对 $g_1$，我们有

$$
\begin{aligned}
g_1(x) &= -\frac{3}{2}(x-1)+1, \\
g_1(x)-1 &= -\frac{3}{2}(x-1), \\
x_{i+1}^1 - 1 &= -\frac{3}{2}(x_i^1 - 1).
\end{aligned} \tag{8}
$$

对 $g_2$，我们有

$$
\begin{aligned}
g_2(x) &= -\frac{1}{2}(x-1)+1, \\
g_2(x)-1 &= -\frac{1}{2}(x-1), \\
x_{i+1}^2 - 1 &= -\frac{1}{2}(x_i^2 - 1).
\end{aligned} \tag{9}
$$

# 不动点迭代的线性收敛性

这里我们可以看出，如果令 $e_i^1 = |x_i^1 - 1|$ 为 $g_1$ 迭代到第 $i$ 步后的误差绝对值，$e_i^2 = |x_i^2 - 1|$ 为 $g_2$ 迭代到第 $i$ 步后的误差的绝对值，我们可以发现 $e_{i+1}^1 = \frac{3}{2}e_i^1$，而 $e_{i+1}^2 = \frac{1}{2}e_i^2$，即迭代函数 $g_1$ 产生的误差不断扩大，而迭代函数 $g_2$ 产生的误差不断缩小。为了更一般的讨论不动点迭代的收敛性，我们有如下定义。

**Definition 9 (线性收敛)**

令 $e_i$ 表示一个迭代方法在第 $i$ 步迭代后的误差的绝对值。如果

$$\lim_{i\to\infty} \frac{e_{i+1}}{e_i} = S < 1, \tag{10}$$

则称这个方法具有线性收敛性，且收敛率为 $S$。

# 不动点迭代的线性收敛性

**Definition 10 (局部收敛)**

对于一个迭代方法，如果初始猜测 $x_0$ 足够靠近 $r$ 就能够产生收敛于 $r$ 的迭代序列，则称这个迭代法是局部收敛的。

根据这些定义，我们有如下定理

**Theorem 11 (不动点迭代的线性收敛性)**

*假设 $g$ 是一个连续可导函数满足 $g(r) = r$ 且 $S = |g'(r)| < 1$。则当初始猜测 $x_0$ 足够靠近不动点 $r$ 时，由 $g$ 这个函数定义的不动点迭代所产生的迭代序列收敛于不动点 $r$，且这个收敛是线性的，收敛率为 $S$。*

# 不动点迭代应用举例

**Example 12**

利用不动点迭代寻找 $\cos x = \sin x$ 的根。

我们将这个问题转化为求 $g(x) = x + \cos x - \sin x$ 的不动点的问题。求解的结果可以由下表表示

# 不动点迭代应用举例

| $i$ | $x_i$ | $g(x_i)$ | $e_i = \lvert x_i - r \rvert$ | $e_i / e_{i-1}$ |
|---|---|---|---|---|
| 0 | 0.0000000 | 1.0000000 | 0.7853982 | |
| 1 | 1.0000000 | 0.6988313 | 0.2146018 | 0.273 |
| 2 | 0.6988313 | 0.8211025 | 0.0865669 | 0.403 |
| 3 | 0.8211025 | 0.7706197 | 0.0357043 | 0.412 |
| 4 | 0.7706197 | 0.7915189 | 0.0147785 | 0.414 |
| 5 | 0.7915189 | 0.7828629 | 0.0061207 | 0.414 |
| 6 | 0.7828629 | 0.7864483 | 0.0025353 | 0.414 |
| 7 | 0.7864483 | 0.7849632 | 0.0010501 | 0.414 |
| 8 | 0.7849632 | 0.7855783 | 0.0004350 | 0.414 |
| 9 | 0.7855783 | 0.7853235 | 0.0001801 | 0.414 |
| 10 | 0.7853235 | 0.7854291 | 0.0000747 | 0.415 |
| 11 | 0.7854291 | 0.7853854 | 0.0000309 | 0.414 |
| 12 | 0.7853854 | 0.7854035 | 0.0000128 | 0.414 |
| 13 | 0.7854035 | 0.7853960 | 0.0000053 | 0.414 |
| 14 | 0.7853960 | 0.7853991 | 0.0000022 | 0.415 |
| 15 | 0.7853991 | 0.7853978 | 0.0000009 | 0.409 |
| 16 | 0.7853978 | 0.7853983 | 0.0000004 | 0.444 |
| 17 | 0.7853983 | 0.7853981 | 0.0000001 | 0.250 |
| 18 | 0.7853981 | 0.7853982 | 0.0000001 | 1.000 |
| 19 | 0.7853982 | 0.7853982 | 0.0000000 | |

Figure 9: $g(x) = x + \cos x - \sin x$ 的迭代结果

# 不动点迭代应用举例

在这个例子中我们可以观察到以下几点。

- 首先，迭代序列收敛到 $0.7853982 \approx \frac{\pi}{4}$，因为 $\cos\frac{\pi}{4} = \sin\frac{\pi}{4}$。
- 第二，表的第四列列出了迭代序列与真解 $\frac{\pi}{4}$ 之间的误差，这个误差随着迭代的进行逐步减小。
- 第三，表的第五列展示出了两步之间的误差的比值，可以看到，在大部分的时候这个比值停留在 0.414。这与我们的理论推导是高度一致的，因为

$$S = |g'(r)| = |1 - \sin r - \cos r| = \left|1 - \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}\right| = |1 - \sqrt{2}| \approx 0.414. \tag{11}$$

# 停机准则

与二分法不同，不动点迭代一般不能在计算开始之前确定迭代的步数，因为不动点迭代误差上界与迭代步数之间不存在如二分法一样的确定的公式。因此，我们需要用计算结果来确定是否停止迭代。一般的，迭代停止的准则（又称为停机准则）为

$$|x_{i+1} - x_i| < \text{TOL}, \tag{12}$$

或者当解不接近于 0 时，停机准则为

$$\frac{|x_{i+1} - x_i|}{|x_{i+1}|} < \text{TOL}, \tag{13}$$

其中 TOL 是预先设定的一个误差容许值。当然，设 $\theta > 0$ 是一个接近于 0 的数，当解也接近于 0 的时候，我们可以用如下的混合停机准则

$$\frac{|x_{i+1} - x_i|}{\max\{|x_{i+1}|, \theta\}} < \text{TOL}. \tag{14}$$

# 牛顿迭代的几何意义

牛顿迭代是一种特殊的不动点迭代，一般具有比二分法和大多数不动点迭代都快的收敛速度。牛顿迭代可以由以下的图像给出几何解释

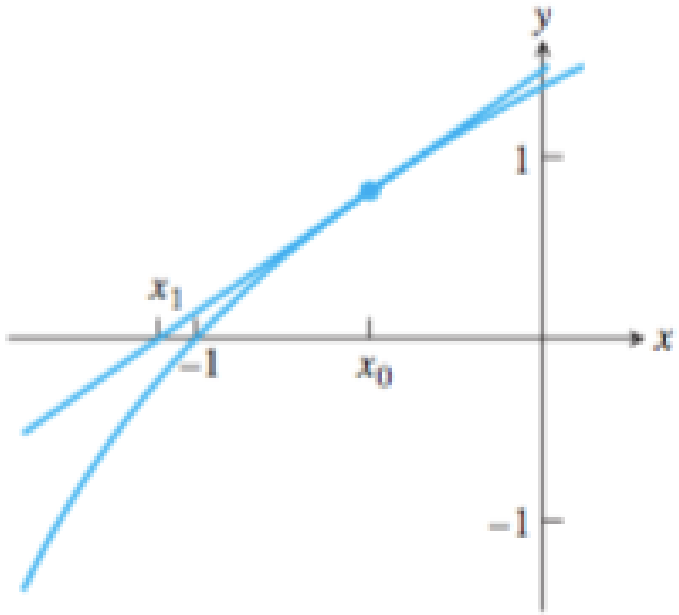


Figure 10: 牛顿迭代的几何解释

# 牛顿迭代的代数导出

牛顿迭代还可以从代数的角度导出。我们将 $f(r)$（$f$ 在根 $r$ 处的值）在根 $x_0$ 处作 Taylor 展开

$$f(r) = f(x_0) + f'(x_0)(r - x_0) + \frac{f''(x_0)}{2!}(r - x_0)^2 + \dots. \tag{15}$$

取线性部分作为 $f(r)$ 的近似

$$f(x_0) + f'(x_0)(r - x_0) \approx f(r) = 0. \tag{16}$$

若 $f'(x_0) \neq 0$，则有

$$r \approx x_0 - \frac{f(x_0)}{f'(x_0)}. \tag{17}$$

# 牛顿迭代法

根据以上的推导，我们可以构造牛顿迭代算法

**Algorithm 4 (牛顿迭代法).**
$x_0 = \text{初始猜测}$
$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, i = 0, 1, 2, \ldots$

**Example 13**

利用牛顿法解方程 $x^3 + x - 1 = 0$。

求导可得 $f'(x) = 3x^2 + 1$，则由牛顿迭代得

$$\begin{aligned} x_{i+1} &= x_i - \frac{x_i^3 + x_i - 1}{3x_i^2 + 1} \\ &= \frac{2x_i^3 + 1}{3x_i^2 + 1}. \end{aligned} \tag{18}$$

# 牛顿迭代法

计算结果见下表

| $i$ | $x_i$ | $e_i = \lvert x_i - r \rvert$ | $e_i / e_{i-1}^2$ |
|---|---|---|---|
| 0 | −0.70000000 | 1.38232780 | |
| 1 | 0.12712551 | 0.55520230 | 0.2906 |
| 2 | 0.95767812 | 0.27535032 | 0.8933 |
| 3 | 0.73482779 | 0.05249999 | 0.6924 |
| 4 | 0.68459177 | 0.00226397 | 0.8214 |
| 5 | 0.68233217 | 0.00000437 | 0.8527 |
| 6 | 0.68232780 | 0.00000000 | 0.8541 |
| 7 | 0.68232780 | 0.00000000 | |

# 牛顿迭代法

迭代的图示如下

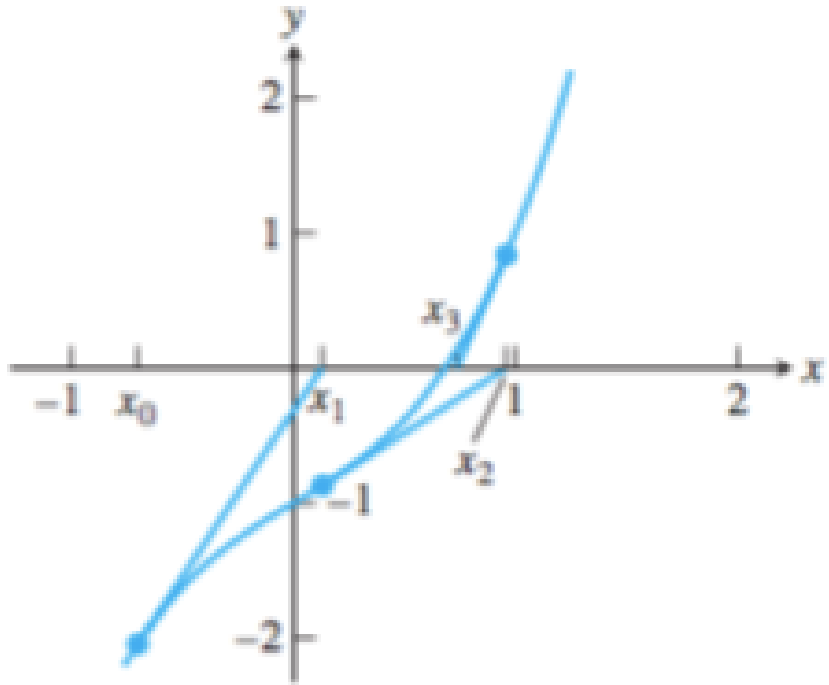


Figure 11: $f(x) = x^3 + x - 1 = 0$ 牛顿迭代示意图

# 牛顿迭代法的二阶收敛性

## Definition 14

令 $e_i$ 表示第 $i$ 步的迭代结果，如果某种迭代法误差的绝对值满足

$$M = \lim_{i \to \infty} \frac{e_{i+1}}{e_i^2} < \infty, \tag{19}$$

则称这种迭代法是二阶收敛的。

## Theorem 15 (牛顿迭代的二阶收敛性)

*假设 $f$ 是二阶连续可导函数且 $f(r) = 0$。如果 $f'(r) \neq 0$，则牛顿迭代局部且二阶收敛到 $r$。第 $i$ 步迭代结果的误差的绝对值 $e_i$ 满足*

$$\lim_{i \to \infty} \frac{e_{i+1}}{e_i^2} = M = \frac{f''(r)}{2f'(r)}. \tag{20}$$

# 线性收敛和二阶收敛性的比较

设 $S = |g'(r)|$，其中 $g$ 为不动点迭代函数，$r$ 为不动点，我们有

$$e_{i+1} \approx Se_i, \tag{21}$$

而牛顿迭代的有

$$e_{i+1} \approx Me_i^2. \tag{22}$$

我们可以发现

1. 当 $e_i < 1$ 开始，二阶收敛的速度明显较线性收敛快。
2. $S$ 的值对于线性收敛非常重要，因为只有 $S < 1$ 时，迭代才可能收敛。
3. $M$ 的值对于二阶收敛并不重要，因为只要 $M$ 不是很大，$e_i^2$ 会很快占主导。

# 牛顿迭代的线性收敛性

定理15并不意味着对于任意的 $f(x)=0$ 的求根问题都是二阶收敛。

**Example 16**

利用牛顿迭代寻找 $f(x)=x^2$ 的根。

利用牛顿迭代

$$\begin{aligned} x_{i+1} &= x_i - \frac{f(x_i)}{f'(x_i)} \\ &= x_i - \frac{x_i^2}{2x_i} \\ &= \frac{x_i}{2}. \end{aligned} \tag{23}$$

# 牛顿迭代的线性收敛性

以下是这样的计算结果

| $i$ | $x_i$ | $e_i = \lvert x_i - r \rvert$ | $e_i / e_{i-1}$ |
|---|---|---|---|
| 0 | 1.000 | 1.000 | |
| 1 | 0.500 | 0.500 | 0.500 |
| 2 | 0.250 | 0.250 | 0.500 |
| 3 | 0.125 | 0.125 | 0.500 |
| ⋮ | ⋮ | ⋮ | ⋮ |

Figure 12: $f(x) = x^2$ 牛顿迭代示意图

# 牛顿迭代的线性收敛性

**Example 17**

利用牛顿迭代寻找 $f(x) = x^m$ 的根。

利用牛顿迭代

$$\begin{aligned} x_{i+1} &= x_i - \frac{x_i^m}{m(x_i^{m-1})} \\ &= \frac{m-1}{m} x_i. \end{aligned} \tag{24}$$

因为 $r = 0$，令 $e_i = |x_i - r| = x_i$，有

$$e_{i+1} = Se_i = \frac{m-1}{m} e_i. \tag{25}$$

# 牛顿迭代的线性收敛性

**Theorem 18**

*设 $f$ 在 $[a,b]$ 上是 $(m+1)-$ 阶连续可导函数，且在 $r \in [a,b]$ 处有 $m$ 重根。则牛顿法局部收敛到 $r$，而第 $i$ 步的误差的绝对值 $e_i$ 满足*

$$\lim_{i \to \infty} \frac{e_{i+1}}{e_i} = S, \tag{26}$$

*其中*

$$S = \frac{m-1}{m}. \tag{27}$$

造成牛顿迭代线性收敛的原因是，在重根 $r$ 处，$f'(r) = 0$，与牛顿迭代二阶收敛的条件 $f'(r) \neq 0$ 相矛盾。

# 改进的牛顿迭代

如果某个根的重数 $m$ 在事先是知道的，那么可以通过改进得到二阶收敛的牛顿迭代。

**Theorem 19 (改进的牛顿迭代的二阶收敛性)**

*设 $f$ 在 $[a,b]$ 上是 $(m+1)-$ 阶连续可导函数，且在 $r \in [a,b]$ 处有 $m$ 重根，则改进的牛顿迭代*

$$x_{i+1} = x_i - \frac{mf(x_i)}{f'(x_i)} \tag{28}$$

*局部二阶收敛到根 $r$。*

# 改进的牛顿迭代

**Example 20**

通过数值方法寻找 $f(x) = \sin x + x^2 \cos x - x^2 - x$ 的根 $r = 0$。

对 $f(x)$ 求导，我们发现

$$\begin{aligned} f'(x) &= \cos x + 2x \cos x - x^2 \sin x - 2x - 1, \\ f''(x) &= -\sin x + 2\cos x - 4x \sin x - x^2 \cos x - 2, \\ f'''(x) &= -\cos x - 6 \sin x - 6x \cos x + x^2 \sin x. \end{aligned} \tag{29}$$

由此得 $f(0) = f'(0) = f''(0) = 0$ 而 $f'''(0) = -1$。. 构造牛顿迭代

$$\begin{aligned} x_{i+1} &= x_i - \frac{f(x_i)}{f'(x_i)} \\ &= x_i - \frac{\sin x_i + x_i^2 \cos x_i - x_i^2 - x_i}{-\sin x_i + 2\cos x_i - 4x_i \sin x_i - x_i^2 \cos x_i - 2}. \end{aligned} \tag{30}$$

# 改进的牛顿迭代

计算结果如下表，可以观察到 $\frac{e_{i+1}}{e_i} \approx \frac{2}{3}$。

| $i$ | $x_i$ | $e_i = \lvert x_i - r \rvert$ | $e_i / e_{i-1}$ |
|---|---|---|---|
| 1 | 1.00000000000000 | 1.00000000000000 | |
| 2 | 0.72159023986075 | 0.72159023986075 | 0.72159023986075 |
| 3 | 0.52137095182040 | 0.52137095182040 | 0.72253049309677 |
| 4 | 0.37530830859076 | 0.37530830859076 | 0.71984890466250 |
| 5 | 0.26836349052713 | 0.26836349052713 | 0.71504809348561 |
| 6 | 0.19026161369924 | 0.19026161369924 | 0.70896981301561 |
| 7 | 0.13361250532619 | 0.13361250532619 | 0.70225676492686 |
| 8 | 0.09292528672517 | 0.09292528672517 | 0.69548345417455 |
| 9 | 0.06403926677734 | 0.06403926677734 | 0.68914790617474 |
| 10 | 0.04377806216009 | 0.04377806216009 | 0.68361279513559 |
| 11 | 0.02972805552423 | 0.02972805552423 | 0.67906284694649 |
| 12 | 0.02008168373777 | 0.02008168373777 | 0.67551285759009 |
| 13 | 0.01351212730417 | 0.01351212730417 | 0.67285828621786 |
| 14 | 0.00906579564330 | 0.00906579564330 | 0.67093770205249 |
| 15 | 0.00607029292263 | 0.00607029292263 | 0.66958192766231 |
| 16 | 0.00405885109627 | 0.00405885109627 | 0.66864171927113 |
| 17 | 0.00271130367793 | 0.00271130367793 | 0.66799781850081 |
| 18 | 0.00180995966250 | 0.00180995966250 | 0.66756065624029 |
| 19 | 0.00120772384467 | 0.00120772384467 | 0.66726561353325 |
| 20 | 0.00080563307149 | 0.00080563307149 | 0.66706728946460 |

Figure 13: 利用牛顿迭代计算 $f(x) = \sin x + x^2 \cos x - x^2 - x$ 根的结果

# 改进的牛顿迭代

如果构造改进的牛顿迭代

$$
\begin{aligned}
x_{i+1} &= x_i - \frac{3f(x_i)}{f'(x_i)} \\
&= x_i - \frac{3(\sin x_i + x_i^2 \cos x_i - x_i^2 - x_i)}{-\sin x_i + 2\cos x_i - 4x_i \sin x_i - x_i^2 \cos x_i - 2},
\end{aligned} \tag{31}
$$

则计算结果如下

| $i$ | $x_i$ |
|---|---|
| 0 | 1.00000000000000 |
| 1 | 0.16477071958224 |
| 2 | 0.01620733771144 |
| 3 | 0.00024654143774 |
| 4 | 0.00000006072272 |
| 5 | −0.00000000633250 |

Figure 14: 利用改进的牛顿迭代计算 $f(x) = \sin x + x^2 \cos x - x^2 - x$ 根的结果

# 牛顿迭代小结

牛顿迭代是非常重要的求解非线性方程根的方法，最大的优点是收敛速度快，尤其是当迭代值落入局部收敛区域的时候，往往几步迭代就可以得到比较精确的解。

但牛顿迭代不是万能的，如果初值选区不当很可能会出现牛顿迭代发散的情况产生，并且在迭代结果为重根的时候只有线性的收敛速度。除此之外，牛顿迭代需要计算一阶导数，在导数比较复杂的时候，会产生较大的计算量。

# 无导数求解非线性方程

牛顿迭代的其中一个缺点是需要求函数的一阶导数，这在导数比较复杂的时候会产生较大的计算量，甚至是不可能求出来的，因此，我们需要用一些其它方法代替牛顿迭代。这里介绍几个牛顿迭代常用的变形。

# 割线法

割线法与牛顿迭代类似，主要区别是利用一个差分来代替导数，从几何上说，就是将切线用一条割线代替。具体来说，在已知前两次迭代结果 $x_i$ 和 $x_{i-1}$ 的时候，我们利用

$$f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}, \tag{32}$$

得到割线法：

**Algorithm 5 (割线法).**
$x_0, x_1$ *为初始猜测*
$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}, i = 1, 2, 3, \ldots$

# 割线法的超线性收敛

与不动点迭代和牛顿迭代不同，割线法需要两个初始值的猜测。
可以证明当割线法的迭代结果收敛到 $r$ 且 $f'(r) \neq 0$ 的时候，误差满足

$$e_{i+1} \approx \left|\frac{f''(r)}{2f'(r)}\right| e_i e_{i-1}, \tag{33}$$

进而可以证明

$$e_{i+1} \approx \left|\frac{f''(r)}{2f'(r)}\right|^{\alpha-1} e_i^{\alpha}, \tag{34}$$

其中 $\alpha = \frac{1+\sqrt{5}}{2} \approx 1.62$。很明显，割线法的收敛比线性收敛好但比二阶收敛差一些，这样的收敛性一般称为超线性收敛。

# 错位法

错位法是割线法的另一个变形。假设我们知道函数的根在 $[a,b]$ 中且 $f(a)f(b)<0$，那么令下一次的迭代结果为

$$c=a-\frac{f(a)(a-b)}{f(a)-f(b)}=\frac{bf(a)-af(b)}{f(a)-f(b)}. \tag{35}$$

可以证明，因为 $f(a)$ 与 $f(b)$ 反号，所以 $a\leq c\leq b$。接下来根据 $f(a)f(c)<0$ 或者 $f(c)f(b)<0$ 来确定新的搜索区间。错位法的算法如下。

# 错位法的算法

**Algorithm 6 (错位法).**
给出有根区间 $[a,b]$ 且 $f(a)f(b)<0$。
for $i=1,2,3,\ldots$
  $c=\frac{bf(a)-af(b)}{f(a)-f(b)}$
  if $f(c)=0$，停机，输出 $c$
  if $f(a)f(c)<0$
    $b=c$
  else
    $a=c$
  end
end

# 课后阅读

[NA] 第 1 章 1.1，1.2，1.4，1.5.1