非线性求解方程

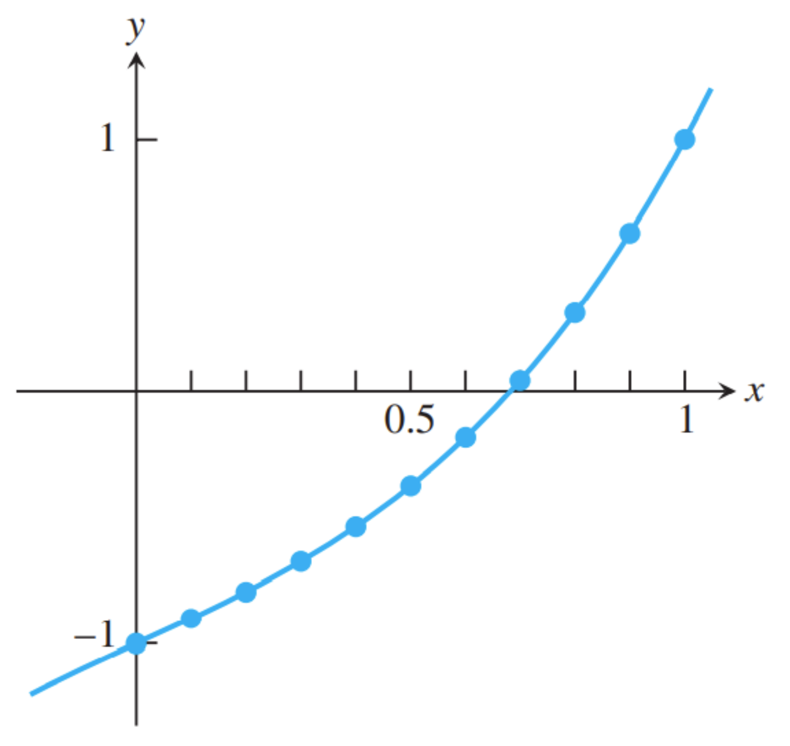
- ◆二分法
- ◆不动点迭代
- ◆精度极限\根搜索敏感性
- ◆牛顿迭代法
- ◆割线法
- ◆抛物法\混合方法
- ◆方程组的牛顿迭代法

数值求解方程的必要性

- 方程求解是工程计算中最重要的问题之一
- 很多方程无法得到解析解，依赖于数值技术
- 光的衍射理论：$x - \tan x = 0$
- 行星轨道计算的开普勒方程：$x - a\sin x = b$

二分法

- **定理：** 如果$f(x) \in C[a,b]$，并满足$f(a)f(b) < 0$，则$\exists r \in (a,b)$使得$f(r) = 0$.



$f(x) = x^3 + x - 1$的函数图像

二分法

- 二分法伪代码(pseudo-code)

Given initial interval $[a, b]$ such that $f(a)f(b) < 0$

while $(b - a)/2 > \text{TOL}$

    $c = (a + b)/2$

    if $f(c) = 0$, **stop**, **end**

    if $f(a)f(c) < 0$

        $b = c$

    else

        $a = c$

    end

end

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

The approximate root is $(a + b)/2$.

- 误差？计算量？

二分法

- 例：用二分法求$f(x) = \cos(x) - x$在[0,1]之间的根，并使得根具有6位有效数字。

估算二分次数：

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

$$\rightarrow n > \frac{6}{\log_{10} 2} \approx 19.9$$

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	−

不动点迭代

- 引例：对任意初值重复应用余弦函数

$x_0 = 0$

1	1.00000e+000
2	5.40302e-001
3	8.57553e-001
4	6.54290e-001
5	7.93480e-001
6	7.01369e-001
7	7.63960e-001
8	7.22102e-001
9	7.50418e-001
10	7.31404e-001
11	7.44237e-001
12	7.35605e-001
13	7.41425e-001
14	7.37507e-001
15	7.40147e-001
16	7.38369e-001
17	7.39567e-001
18	7.38760e-001
19	7.39304e-001
20	7.38938e-001

20	7.38938e-001
21	7.39184e-001
22	7.39018e-001
23	7.39130e-001
24	7.39055e-001
25	7.39106e-001
26	7.39071e-001
27	7.39094e-001
28	7.39079e-001
29	7.39089e-001
30	7.39082e-001
31	7.39087e-001
32	7.39084e-001
33	7.39086e-001
34	7.39085e-001
35	7.39086e-001
36	7.39085e-001
37	7.39085e-001
38	7.39085e-001
39	7.39085e-001
40	7.39085e-001

$x_0 = 1234$

1	-7.98551e-001
2	6.97746e-001
3	7.66293e-001
4	7.20487e-001
5	7.51485e-001
6	7.30676e-001
7	7.44723e-001
8	7.35275e-001
9	7.41646e-001
10	7.37358e-001
11	7.40248e-001
12	7.38302e-001
13	7.39613e-001
14	7.38730e-001
15	7.39325e-001
16	7.38924e-001
17	7.39194e-001
18	7.39012e-001
19	7.39134e-001
20	7.39052e-001

20	7.39052e-001
21	7.39108e-001
22	7.39070e-001
23	7.39095e-001
24	7.39078e-001
25	7.39090e-001
26	7.39082e-001
27	7.39087e-001
28	7.39084e-001
29	7.39086e-001
30	7.39084e-001
31	7.39086e-001
32	7.39085e-001
33	7.39085e-001
34	7.39085e-001
35	7.39085e-001
36	7.39085e-001
37	7.39085e-001
38	7.39085e-001
39	7.39085e-001
40	7.39085e-001

不动点迭代

- 定义：如果$g(r)=r$，则r为$g(x)$的不动点。

- 例：$g(x)=\cos(x)$的不动点为$0.739085\ldots$

- 例：地图

不动点迭代

- 例：$g(x) = x^3$的不动点？
- 不动点与方程根的关系：$g(r) = r \Leftrightarrow f(r) = g(r) - r = 0$
- 例：$r = 0.739085\ldots$为$f(x) = \cos x - x$的根。

不动点迭代

- 不动点迭代

$$
\begin{aligned}
x_0 &= \text{initial guess} \\
x_{i+1} &= g(x_i) \ \textbf{for} \ i = 0, 1, 2, \ldots
\end{aligned}
$$

$$
\begin{aligned}
x_1 &= g(x_0) \\
x_2 &= g(x_1) \\
x_3 &= g(x_2) \\
&\vdots
\end{aligned}
$$

- 如果$g(x)$连续且$\lim_{i\to\infty} x_i = r$（收敛）：

$$g(r) = g\left(\lim_{i\to\infty} x_i\right) = \lim_{i\to\infty} g(x_i) = \lim_{i\to\infty} x_{i+1} = r$$

不动点迭代

- 需要回答的问题：

1. 方程的不动点函数是否唯一？
2. 是否总是收敛？
3. 什么条件下收敛？
4. 收敛速度？

不动点迭代

- 例：用不动点迭代求解$x^3 + x - 1 = 0$

$$g(x) = 1 - x^3$$

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

$$g(x) = \sqrt[3]{1 - x}$$

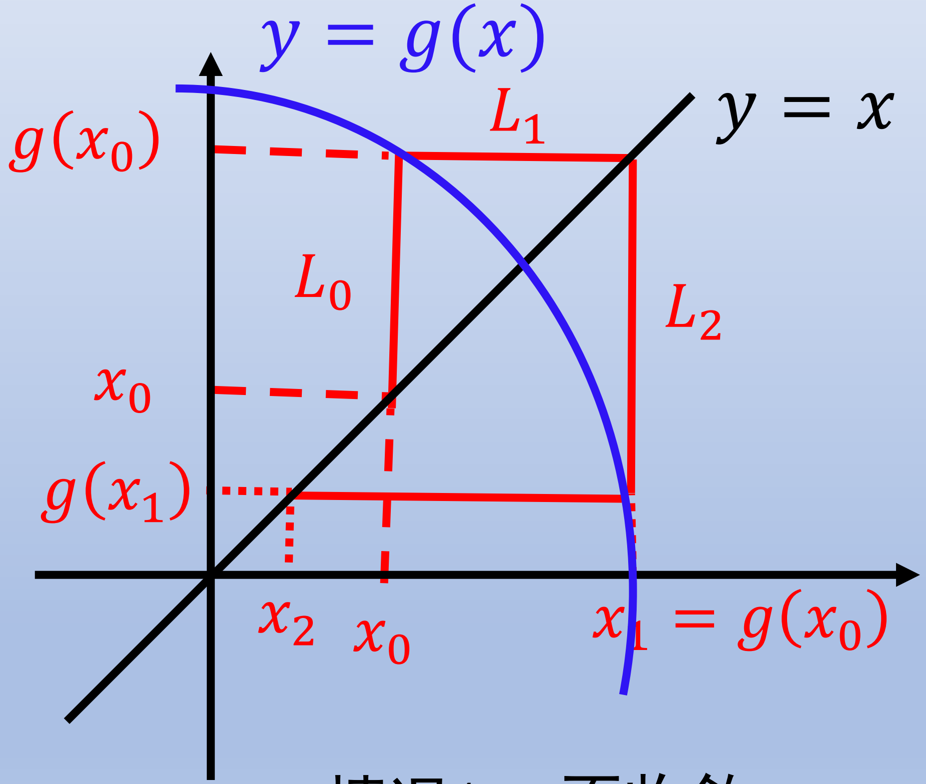
i	x_i	i	x_i
0	0.50000000	13	0.68454401
1	0.79370053	14	0.68073737
2	0.59088011	15	0.68346460
3	0.74236393	16	0.68151292
4	0.63631020	17	0.68291073
5	0.71380081	18	0.68191019
6	0.65900615	19	0.68262667
7	0.69863261	20	0.68211376
8	0.67044850	21	0.68248102
9	0.69072912	22	0.68221809
10	0.67625892	23	0.68240635
11	0.68664554	24	0.68227157
12	0.67922234	25	0.68236807

$$g(x) = \frac{1 + 2x^3}{1 + 3x^2}$$

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

不动点迭代

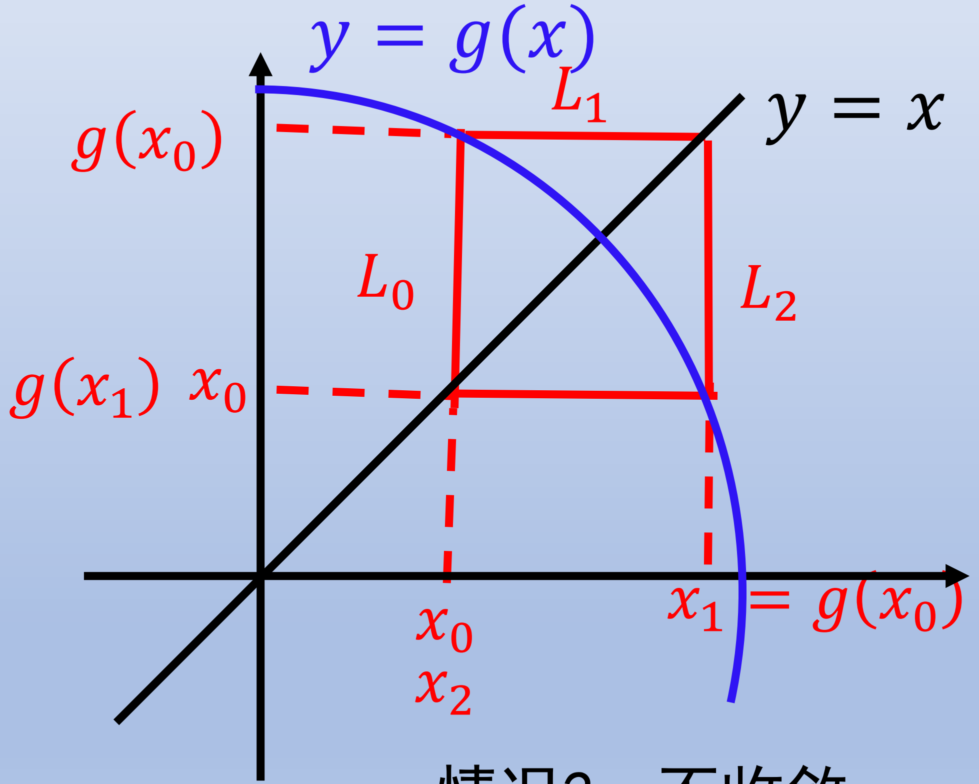
- 收敛的几何解释



情况1　不收敛

$$L_2 = |g(x_1) - g(x_0)| > L_0 = L_1 = |x_1 - x_0|$$

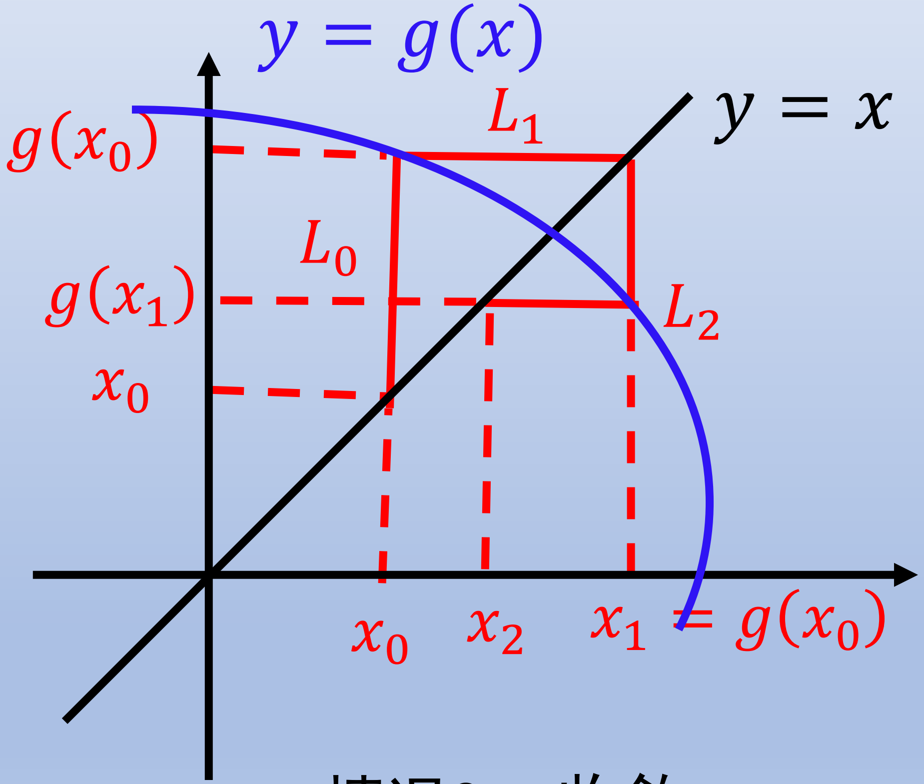
不动点迭代



情况2　不收敛

$$L_2 = |g(x_1) - g(x_0)| = L_0 = L_1 = |x_1 - x_0|$$

不动点迭代

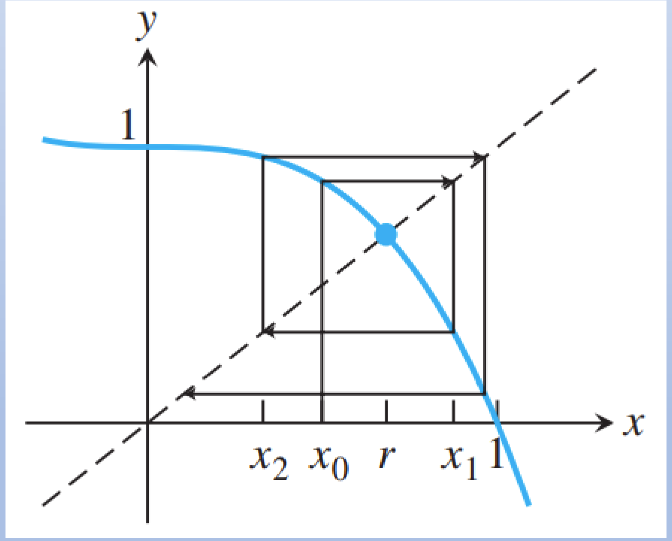


情况3　收敛

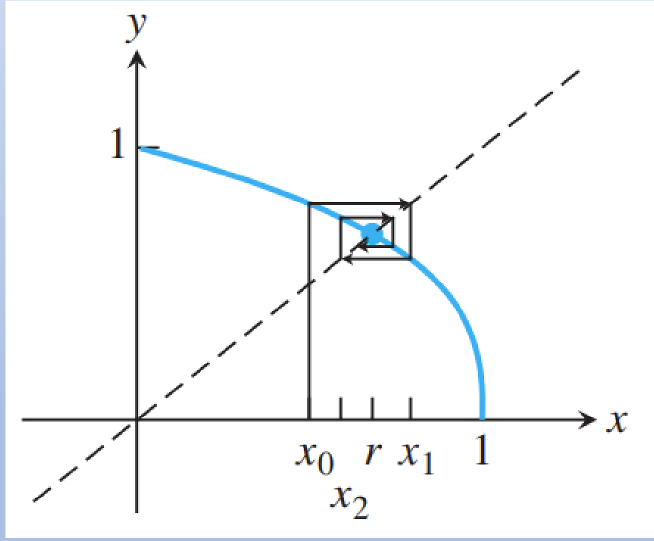
$$L_2 = |g(x_1) - g(x_0)| < L_0 = L_1 = |x_1 - x_0|$$

不动点迭代

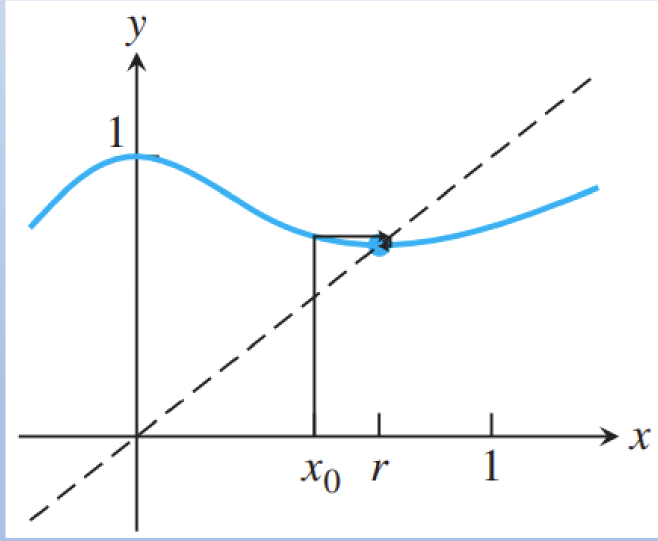
- 不同不动点迭代的蛛网图（cobweb）



$$g(x) = 1 - x^3$$



$$g(x) = \sqrt[3]{1 - x}$$



$$g(x) = \frac{1 + 2x^3}{1 + 3x^2}$$

不动点迭代

- 定义：假设$g(x)$在区间I存在不动点x^*，若对$\forall x_0 \in I$，不动点迭代产生的序列$\{x_k\}$收敛到x^*，则称不动点迭代全局收敛。

- 定义：假设$g(x)$在区间I存在不动点x^*，若存在x^*的邻域$N \subset I$，对$\forall x_0 \in N$，不动点迭代产生的序列$\{x_k\}$收敛到x^*，则称不动点迭代局部收敛。

- 定义：如果存在常数$L > 0$, 使得$\forall x_1, x_2 \in I$满足$|f(x_1) - f(x_2)| \le L|x_1 - x_2|$，则称函数$f(x)$在区间$I$满足Lipschitz条件，其中$L$称为Lipschitz常数。

不动点迭代

- 定理：$g(x) \in C[a,b]$连续

1、如果$\forall x \in [a,b]$有$a \le g(x) \le b$,则$g(x)$在$[a,b]$上存在不动点。

2、在满足条件1的基础上，如果存在常数$L \in (0,1)$，使得

$$|g(x_1) - g(x_2)| \le L|x_1 - x_2|, \forall x_1, x_2 \in [a,b],$$

则:

① $g(x)$在$[a,b]$上存在唯一不动点x^*

② $\forall x_0 \in [a,b]$, 不动点迭代收敛到x^*(全局收敛)，误差满足：

$$|x_k - x^*| \le \frac{L}{1-L}|x_k - x_{k-1}| \le \cdots \le \frac{L^k}{1-L}|x_1 - x_0|$$

不动点迭代

- 推论： $g(x)$在$[a,b]$连续，在(a,b)可微，对 $\forall x \in [a,b]$有$a \le g(x) \le b$，且存在常数$L \in (0,1)$，使得

$$|g'(x)| \le L, \forall x \in (a,b)$$

则$g(x)$在$[a,b]$上存在唯一不动点。

- 局部收敛定理：设x^*为$g(x)$的不动点，如果$g'(x)$在区间I连续且$|g'(x^*)| < 1$，则不动点迭代$x_{i+1} = g(x_i)$局部收敛。

不动点迭代

- 例：$g(x) = \cos x$是否局部\全局收敛？

- 例：求解$x^3 + x - 1 = 0$的不动点迭代对于$r \approx 0.6823$是否收敛？

1. $g(x) = 1 - x^3$; $|g'(r)| = 3r^2 \approx 1.395$

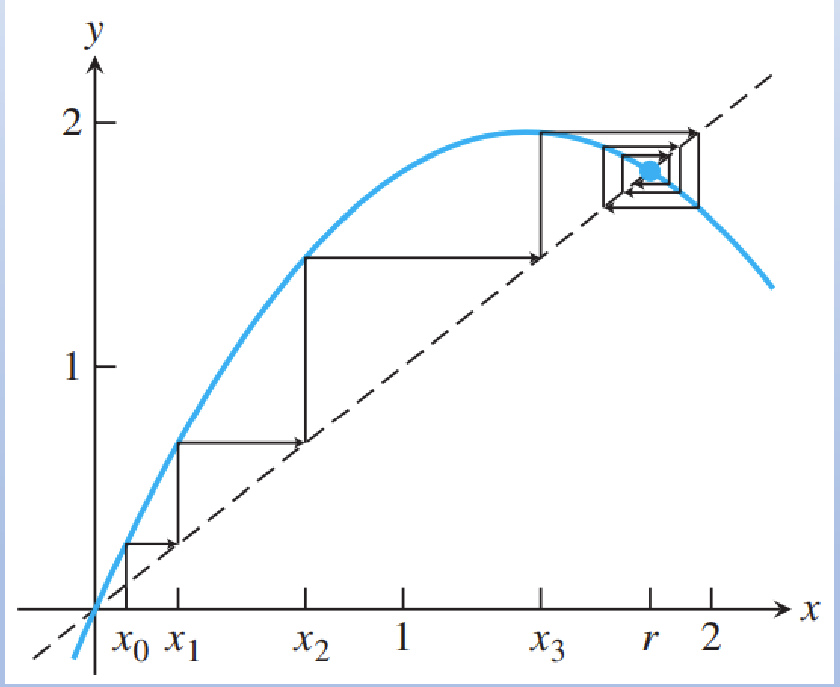
2. $g(x) = \sqrt[3]{1 - x}$; $|g'(r)| = \frac{1}{3}(1 - r)^{-\frac{2}{3}} \approx 0.716$

3. $g(x) = \frac{1+2x^3}{1+3x^2}$; $g'(x) = \frac{6x^2(1+3x^2) - 6x(1+2x^3)}{(1+3x^2)^2} = \frac{6x(x^3+x-1)}{(1+3x^2)^2}$

不动点迭代

- 例：迭代寻找$g(x) = 2.8x - x^2$的不动点

$$x_0 = 0.1000$$
$$x_1 = 0.2700$$
$$x_2 = 0.6831$$
$$x_3 = 1.4461$$
$$x_4 = 1.9579,$$



迭代过程的蛛网图

不动点迭代

- 例：古巴比伦人计算$\sqrt{2} = 1.41421356\ldots$

$$x_0 = 1 \qquad x_1 = \frac{x_0 + \frac{2}{x_0}}{2} = \frac{1 + \frac{2}{1}}{2} = \frac{3}{2} \qquad x_2 = \frac{x_1 + \frac{2}{x_1}}{2} = \frac{\frac{3}{2} + \frac{4}{3}}{2} = \frac{17}{12} \approx 1.41666\ldots$$

$$x_3 = \frac{x_2 + \frac{2}{x_2}}{2} = \frac{\frac{17}{12} + \frac{24}{17}}{2} = \frac{577}{408} \approx 1.41421568\ldots \qquad x_4 = \frac{x_3 + \frac{2}{x_3}}{2} \approx 1.41421356\ldots$$

- 该过程相当于不动点迭代:$x_{i+1} = \frac{x_i + \frac{2}{x_i}}{2}$

不动点迭代

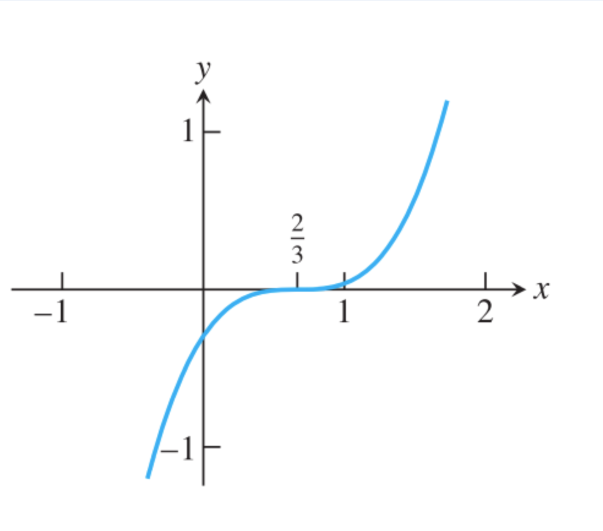
- 迭代终止条件
  1. 绝对误差终止条件
$$|x_{i+1} - x_i| < TOL$$
  2. 相对误差终止条件(x_i不在零附近)
$$\frac{|x_{i+1} - x_i|}{|x_i|} < TOL$$
  3. 混合终止条件(x_i在零附近)
$$\frac{|x_{i+1} - x_i|}{\max(\theta, |x_i|)} < TOL, \theta > 0$$
  4. 为了防止迭代失败出现锁死的情况，需要加上迭代步数的限制

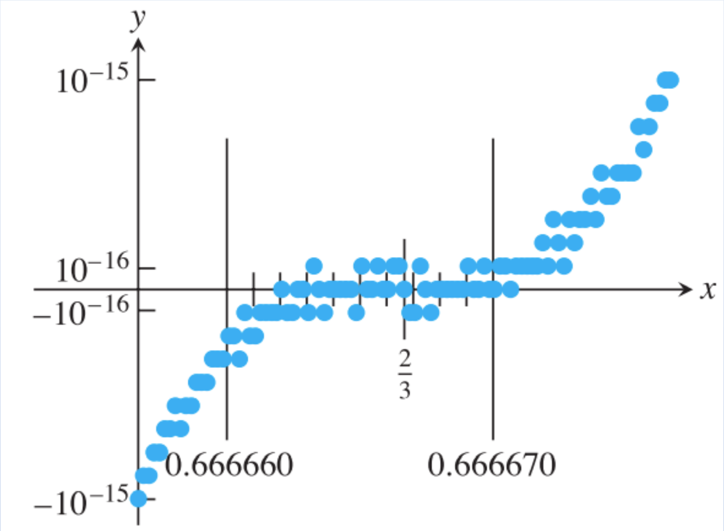
精度的极限

- 例：二分法求根$f(x) = x^3 - 2x^2 + \frac{4}{3}x - \frac{8}{27} = \left(x - \frac{2}{3}\right)^3$

i	a_i	$f(a_i)$	c_i	$f(c_i)$	b_i	$f(b_i)$
0	0.0000000	−	0.5000000	−	1.0000000	+
1	0.5000000	−	0.7500000	+	1.0000000	+
2	0.5000000	−	0.6250000	−	0.7500000	+
3	0.6250000	−	0.6875000	+	0.7500000	+
4	0.6250000	−	0.6562500	−	0.6875000	+
5	0.6562500	−	0.6718750	+	0.6875000	+
6	0.6562500	−	0.6640625	−	0.6718750	+
7	0.6640625	−	0.6679688	+	0.6718750	+
8	0.6640625	−	0.6660156	−	0.6679688	+
9	0.6660156	−	0.6669922	+	0.6679688	+
10	0.6660156	−	0.6665039	−	0.6669922	+
11	0.6665039	−	0.6667480	+	0.6669922	+
12	0.6665039	−	0.6666260	−	0.6667480	+
13	0.6666260	−	0.6666870	+	0.6667480	+
14	0.6666260	−	0.6666565	−	0.666687	+
15	0.6666565	−	0.6666718	+	0.6666870	+
16	0.6666565	−	0.6666641	0	0.6666718	+

精度的极限





$f(x)=\left(x-\frac{2}{3}\right)^3$的函数图像　　放大的双精度函数图像

- 前向误差：$|x_n - r|$
- 后向误差：$|f(x_n)|$
- 误差放大因子=相对前向误差/相对后向误差
- 条件数：问题本身所决定的误差放大

精度的极限

- 定义：如果$f(r) = f'(r) = \cdots = f^{(m-1)}(r) = 0, f^{(m)}(r) \neq 0$，则称$r$为$f(x) = 0$的$m$重根。

- 例：$f(x) = \sin x - x$的近似根$x_c = 0.001$

前向误差：$|x_c - r| = 0.001$

后向误差：$|\sin(0.001) - 0.001| \approx 1.6667 \times 10^{-10}$

根搜索的敏感度

- 例：Wilkinson多项式$W(x) = (x-1)(x-2)\dots(x-20)$

$$
\begin{aligned}
W(x) = x^{20} &- 210x^{19} + 20615x^{18} - 1256850x^{17} + 53327946x^{16} - 1672280820x^{15} \\
&+ 40171771630x^{14} - 756111184500x^{13} + 11310276995381x^{12} \\
&- 135585182899530x^{11} + 1307535010540395x^{10} - 10142299865511450x^{9} \\
&+ 63030812099294896x^{8} - 311333643161390640x^{7} \\
&+ 1206647803780373360x^{6} - 3599979517947607200x^{5} \\
&+ 8037811822645051776x^{4} - 12870931245150988800x^{3} \\
&+ 13803759753640704000x^{2} - 8752948036761600000x \\
&+ 2432902008176640000.
\end{aligned}
\tag{1.20}
$$

- 用Matlab求根，初始估计为$r = 16$

```
>> fzero(@wilkpoly,16)

ans =

  16.01468030580458
```

根搜索的敏感度

- 敏感性问题：输入的小误差⟹输出的大误差。
- 例：求$f(x)$的根

原问题：$f(x) = 0 \Longrightarrow x = r$

输入有误差的问题：$f(x) + \epsilon g(x) = 0 \Longrightarrow x = r + \Delta r$

$$\Delta r \approx \frac{-\epsilon g(r)}{f'(r) + \epsilon g'(r)} \approx -\epsilon \frac{g(r)}{f'(r)}$$

误差放大因子$= \left|\frac{\Delta r / r}{\epsilon g(r) / g(r)}\right| = \left|\frac{g(r)}{r f'(r)}\right|$

根搜索的敏感度

- 例：估计$P(x) = (x-1)(x-2)(x-3)(x-4)(x-5)(x-6) - 10^{-6}x^7$的最大根

$$\Delta r \approx -\epsilon \frac{g(r)}{f'(r)} = -\epsilon \frac{6^7}{5!} = -2332.8\epsilon, \epsilon = -10^{-6}$$

$$r + \Delta r \approx 6.0023328$$

精确的根是6. 0023268

$$误差放大因子 = \left|\frac{g(r)}{rf'(r)}\right| = \frac{6^7}{6 \times 5!} \approx 388.8$$

根搜索的敏感度

- 例：Wilkinson多项式中，x^{15}项中的误差对于$r = 16$的影响。

定义:$W_\epsilon(x) = W(x) + \epsilon g(x)$，$g(x) = -1672280820x^{15}$

$$\Delta r = -\frac{\epsilon g(r)}{W'(r)} = \frac{16^{15} \times 1672280820\epsilon}{15!\,4!} \approx 6.1432 \times 10^{13}\epsilon$$

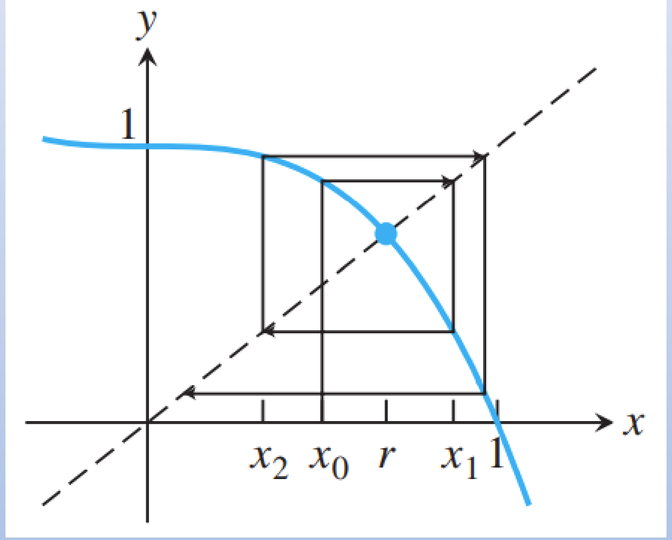
$$\epsilon_{mach} \approx \pm 2^{-52} \approx \pm 2.22 \times 10^{-16},\ \Delta r \approx \pm 0.0136$$

Matlab解≈ 16.01468

$$\text{误差放大因子} = \left|\frac{g(r)}{rW'(r)}\right| \approx 3.8 \times 10^{12}$$

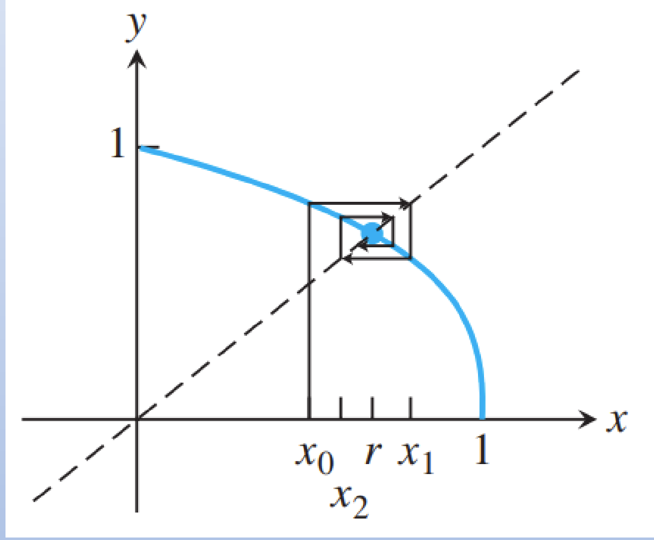
牛顿迭代法

- 不动点迭代遗留的问题：收敛速度？



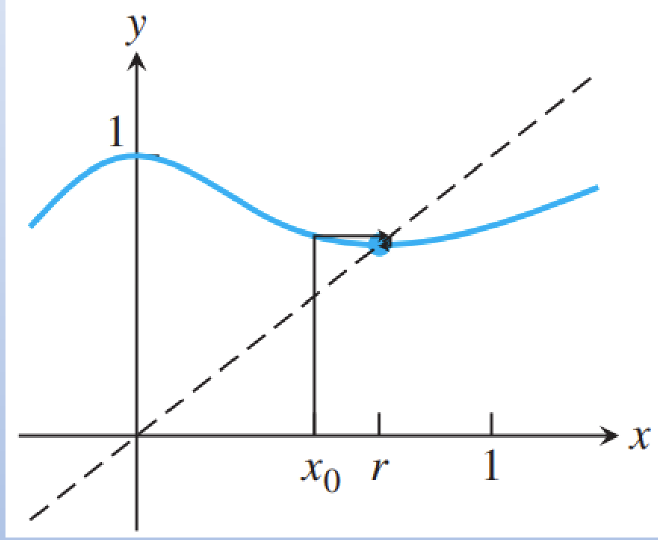
$$g(x) = 1 - x^3$$

$$|g'(r)| \approx 1.395$$



$$g(x) = \sqrt[3]{1 - x}$$

$$|g'(r)| \approx 0.716$$



$$g(x) = \frac{1 + 2x^3}{1 + 3x^2}$$

$$|g'(r)| = 0$$

- 定义：设序列$\{x_k\}$收敛到x^*，定义误差$e_i := |x_i - x^*|$，如果存在实数$p \geq 1$,使得

$$\lim_{i\to\infty} \frac{e_{i+1}}{e_i^p} = C$$

则称序列p阶收敛（$p = 1$称为线性收敛，需满足$C < 1$；$p = 2$称为平方收敛）。

- 定理：假设函数$g \in C^p(a,b), g(x^*) = x^* \in (a,b)$, 并且满足

$$g'(x^*) = g''(x^*) \dots = g^{(p-1)}(x^*) = 0, g^{(p)}(x^*) \neq 0, (p \geq 1)$$

则不动点迭代$x_{i+1} = g(x_i)$, p阶局部收敛到不动点x^*，且有

$$\lim_{i\to\infty} \frac{e_{i+1}}{e_i^p} = \frac{\left|g^{(p)}(x^*)\right|}{p!},$$

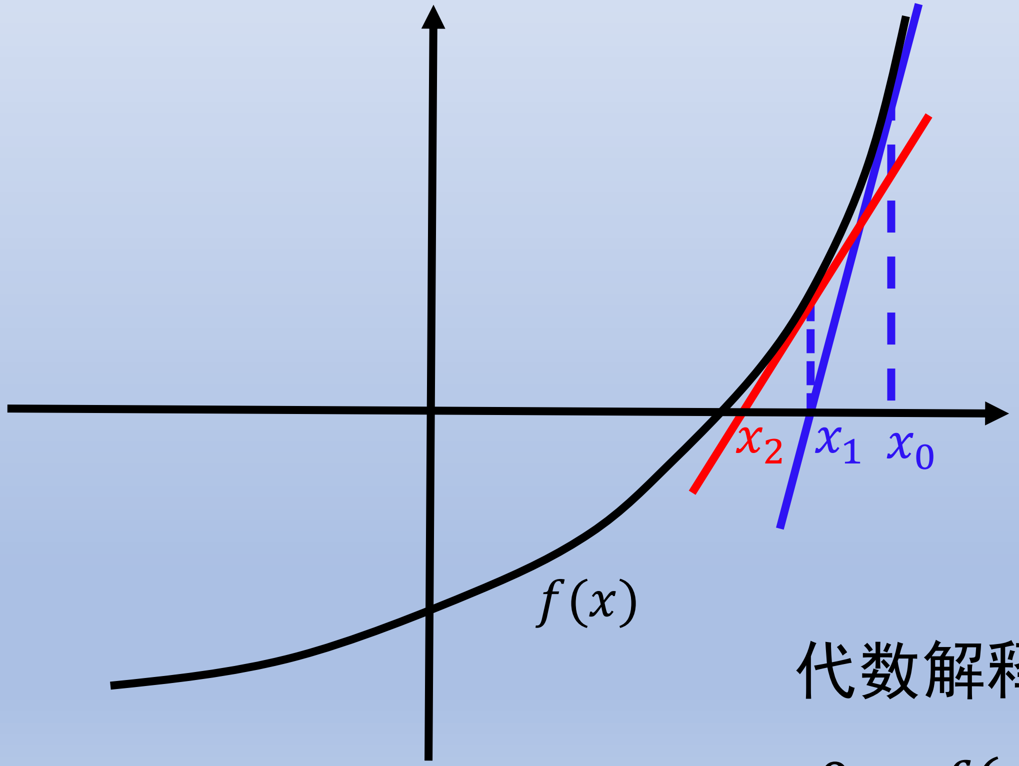
$p = 1$时需满足$|g'(x^*)| < 1$。

牛顿迭代法

- 例：求$\sqrt{a}$的不动点迭代$x_{i+1} = \frac{x_i+\frac{a}{x_i}}{2}$几阶收敛？
- 思考：能否专门设计不动点迭代使得$g'(x^*) = 0$?

牛顿迭代法

- Newton－Raphson 迭代法



几何解释

经过$(x_i, f(x_i))$的切线L_i：

$$y = f(x_i) + f'(x_i)(x - x_i)$$

迭代公式（L_i与x轴的交点）：

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

代数解释：

$$0 = f(x) = f(x_i) + f'(x_i)(x - x_i) \cancel{+ \frac{1}{2} f''(\xi)(x - x_i)^2}$$

牛顿迭代

- 例：求解$x^3 + x - 1 = 0$

不动点迭代：$x_{i+1} = \sqrt[3]{1 - x_i}$

i	x_i	i	x_i
0	0.50000000	13	0.68454401
1	0.79370053	14	0.68073737
2	0.59088011	15	0.68346460
3	0.74236393	16	0.68151292
4	0.63631020	17	0.68291073
5	0.71380081	18	0.68191019
6	0.65900615	19	0.68262667
7	0.69863261	20	0.68211376
8	0.67044850	21	0.68248102
9	0.69072912	22	0.68221809
10	0.67625892	23	0.68240635
11	0.68664554	24	0.68227157
12	0.67922234	25	0.68236807

牛顿迭代？

$$x_{i+1} = \frac{1 + 2x_i^3}{1 + 3x_i^2}$$

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

牛顿迭代法

- 定理：如果$f \in C^2[a,b], \exists r \in (a,b)$满足$f(r)=0$, 且$f'(r) \neq 0, f''(r) \neq 0$, 则牛顿迭代法局部二次收敛到$r$. 且迭代误差$e_i = |x_i - r|$满足

$$\lim_{i\to\infty} \frac{e_{i+1}}{e_i^2} = \left|\frac{f''(r)}{2f'(r)}\right|$$

- 定理：如果$f \in C^2(\mathbb{R})$是单调递增的凸函数，$f(r)=0$, 则r是唯一零点，并且对$\forall x_0 \in \mathbb{R}$牛顿迭代都将收敛到$r$.

牛顿迭代法

- 例：计算$\sqrt{a}$

  问题转换为求$f(x)=x^2-a$的根，用牛顿迭代法。

- 例：牛顿法求$f(x)=ax+b$的根，$a\neq 0$

- 例：牛顿法求$f(x)=x^m$的根。

牛顿迭代法

- 定理：假设$f \in C^{m+1}[a,b]$在$[a,b]$上有m重根$r(m>1)$，则牛顿迭代法局部线性收敛到r，误差满足

$$\lim_{i\to\infty}\frac{e_{i+1}}{e_i}=\frac{m-1}{m}$$

- 定理：假设$f \in C^{m+1}[a,b]$在$[a,b]$上有m重根r 且$f^{(m+1)}(r)\neq 0$，则改进的牛顿迭代法

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

局部二次收敛到r，误差满足

$$\lim_{i\to\infty}\frac{e_{i+1}}{e_i^2}=\frac{1}{m(m+1)}\left|\frac{f^{(m+1)}(r)}{f^{(m)}(r)}\right|$$

- 例：从$x_0 = 1$开始, 用牛顿法寻找$f(x) = \sin x + x^2 \cos x - x^2 - x$的重根$r = 0$，并评估需要多少步才能精确到小数点后第6位。

$$f'(x) = \cos x + 2x \cos x - x^2 \sin x - 2x - 1$$

$$f''(x) = -\sin x + 2\cos x - 4x \sin x - x^2 \cos x - 2$$

$$f'''(x) = -\cos x - 6\sin x - 6x \cos x + x^2 \sin x$$

$r = 0$是三重根，第n步的误差

$$Error = \left(\frac{2}{3}\right)^n < 0.5 \times 10^{-6}$$

$$n > \frac{\log_{10} 0.5 - 6}{\log_{10}\left(\frac{2}{3}\right)} \approx 35.78$$

牛顿迭代法

- 标准的牛顿迭代
- 改进的牛顿迭代

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

i	x_i	$e_i = \|x_i - r\|$	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

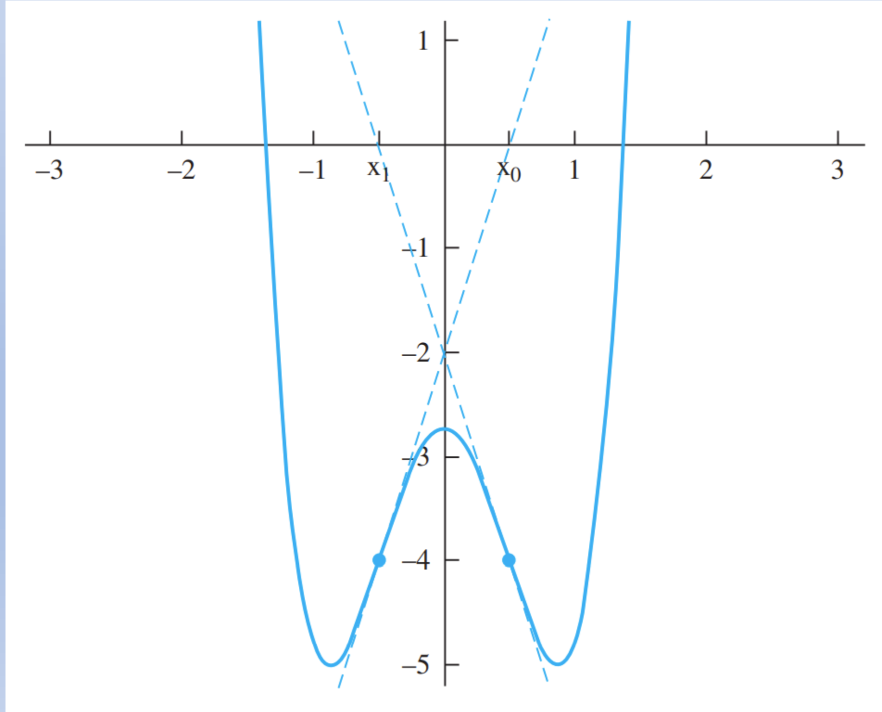
牛顿迭代法

- 牛顿迭代失败的方式：
1. 初始估计不在局部收敛域内
2. 导数为零
3. 根式下出现负值等

- 例：求$f(x) = 4x^4 - 6x^2 - \frac{11}{4}$的根，$x_0 = \frac{1}{2}$

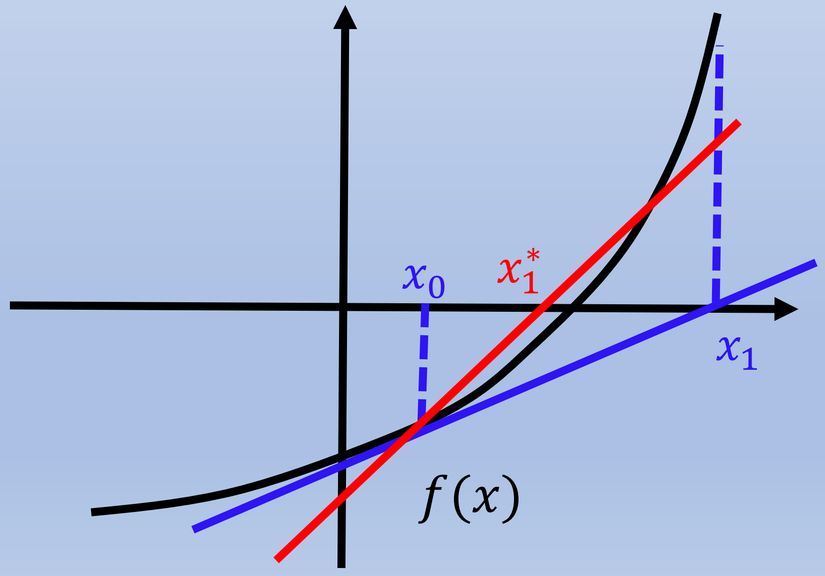
$$x_{i+1} = \frac{12x_i^4 - 6x_i^2 + \frac{11}{4}}{16x_i^3 - 12x_i}$$

$$x_1 = -\frac{1}{2}, x_2 = \frac{1}{2}, \dots$$



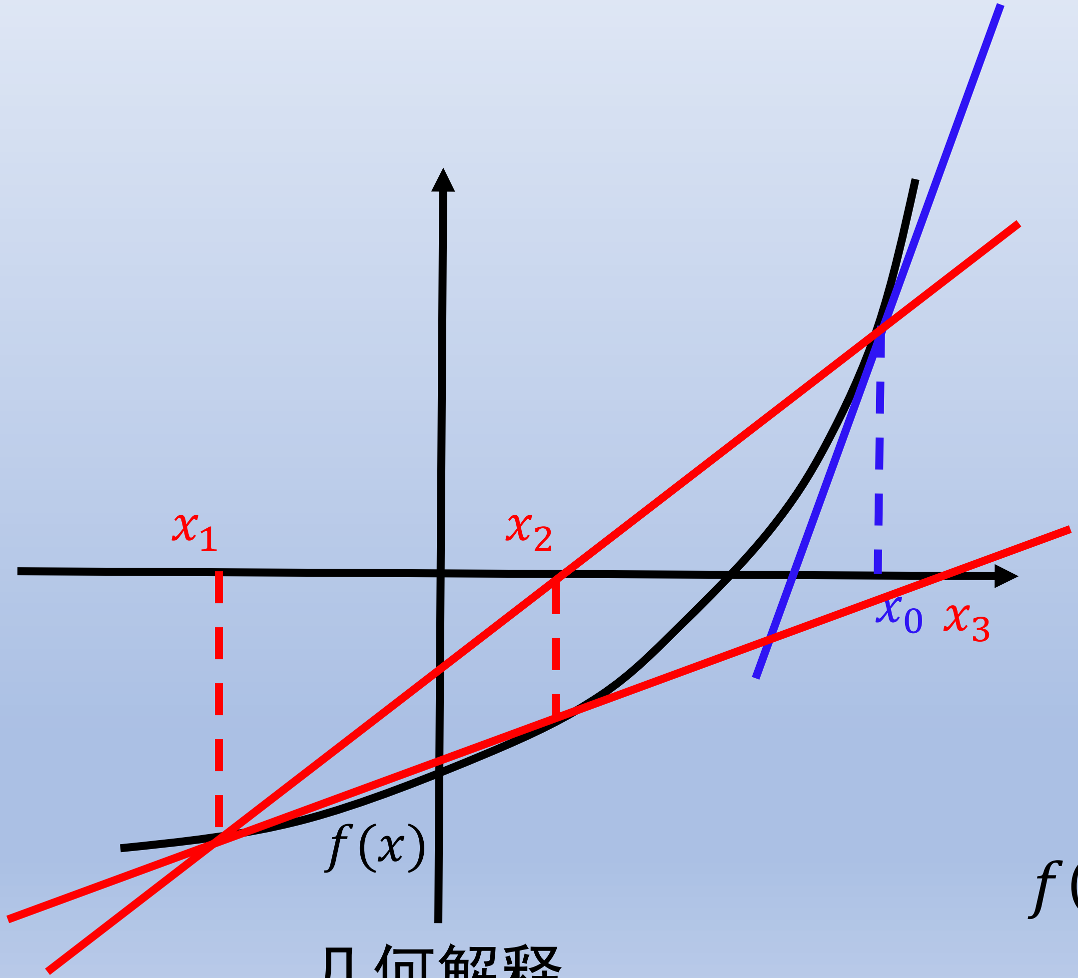
牛顿迭代法

- 牛顿下山法：$x_{i+1} = x_i - \lambda \dfrac{f(x_i)}{f'(x_i)}$
  - ➢初始取$\lambda = 1$，如果$|f(x_{i+1})| < |f(x_i)|$，则进入下一次迭代
  - ➢否则，让λ减半重新进行迭代，直到$|f(x_{i+1})| < |f(x_i)|$。



几何解释

割线法



几何解释

牛顿迭代公式（切线与x轴的交点）：

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

割线法迭代公式（割线与x轴的交点）：

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

谁会收敛更快？

$$f(x) = f(x_i) + f'(x_i)(x - x_i) + \frac{1}{2} f''(\xi)(x - x_i)^2$$

割线法

- 例：求解$x^3+x-1=0$

不动点迭代：$x_{i+1}=\sqrt[3]{1-x_i}$

i	x_i	i	x_i
0	0.50000000	13	0.68454401
1	0.79370053	14	0.68073737
2	0.59088011	15	0.68346460
3	0.74236393	16	0.68151292
4	0.63631020	17	0.68291073
5	0.71380081	18	0.68191019
6	0.65900615	19	0.68262667
7	0.69863261	20	0.68211376
8	0.67044850	21	0.68248102
9	0.69072912	22	0.68221809
10	0.67625892	23	0.68240635
11	0.68664554	24	0.68227157
12	0.67922234	25	0.68236807

牛顿迭代

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

割线法

i	x_i
0	0.00000000000000
1	1.00000000000000
2	0.50000000000000
3	0.63636363636364
4	0.69005235602094
5	0.68202041964819
6	0.68232578140989
7	0.68232780435903
8	0.68232780382802
9	0.68232780382802

割线法

- 误差分析：如果割线法收敛到函数$f \in C^2(\mathbb{R})$的根r，且$f'(r) \neq 0, f''(r) \neq 0$，我们有近似误差关系

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

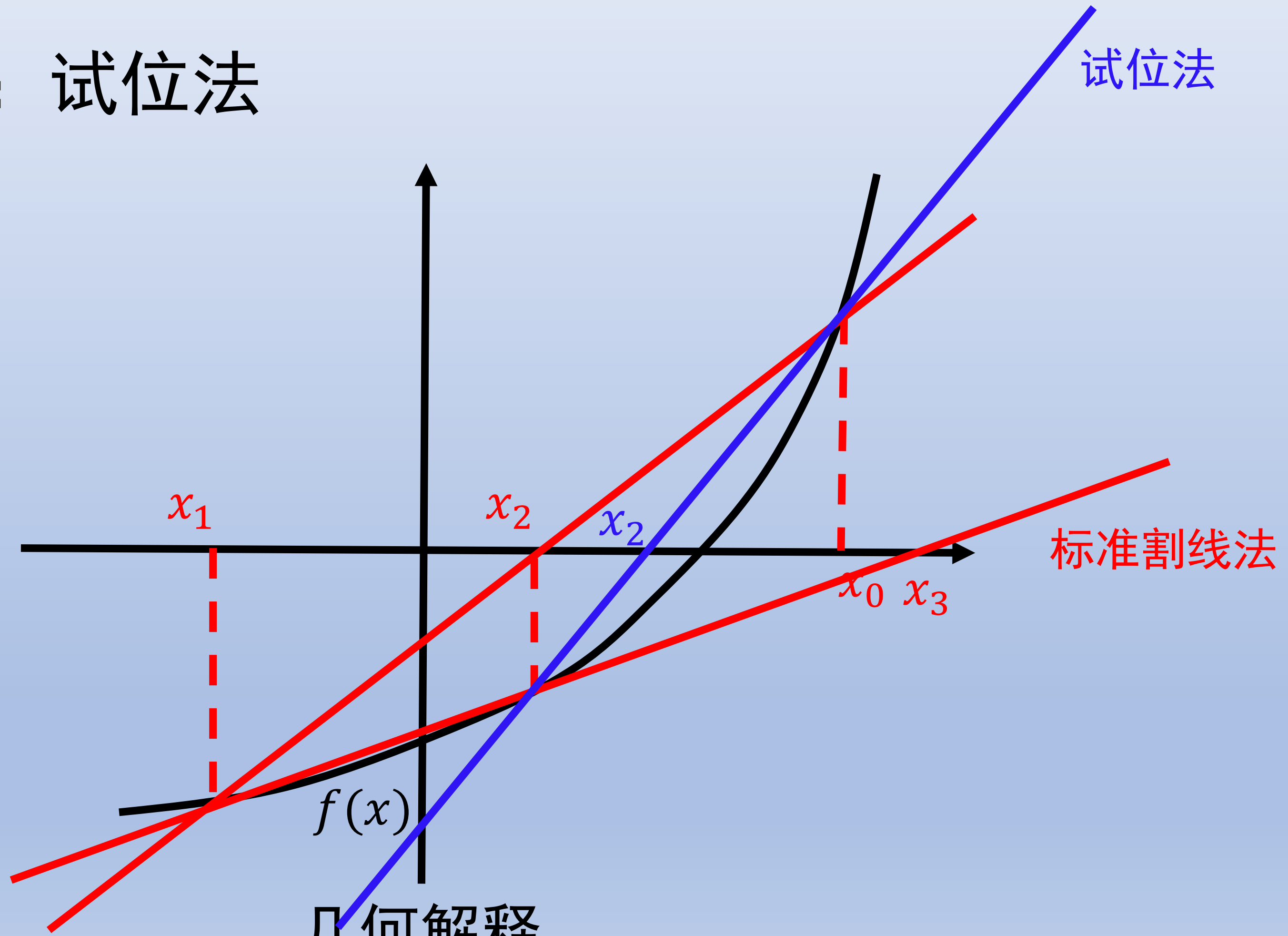
$$e_{i+1} \approx \left|\frac{f''(r)}{2f'(r)}\right|^{\alpha-1} e_i^{\alpha}$$

其中$\alpha = \frac{1+\sqrt{5}}{2} \approx 1.62$(超线性收敛)

- 牛顿法与割线法谁的效率更好？

割线法

- 割线法的变种：试位法



几何解释

- 割线法伪代码

$Given\ interval\ [a,b]$
$such\ that\ f(a)f(b)<0$
for $i=0$ **to** N

$$c \leftarrow \frac{bf(a)-af(b)}{f(a)-f(b)}$$

if $f(c)=0,$ **stop**
else
$a \leftarrow b$
$b \leftarrow c$
end if
end for

- 试位法伪代码

$Given\ interval\ [a,b]$
$such\ that\ f(a)f(b)<0$
for $i=0$ **to** N

$$c \leftarrow \frac{bf(a)-af(b)}{f(a)-f(b)}$$

if $f(c)=0,$ **stop**
else if $f(a)f(c)<0$
$b \leftarrow c$
else
$a \leftarrow c$
end if
end for

割线法

- 例：求解$x^3+x-1=0$

割线法

0	0.00000000000000
1	1.00000000000000
2	0.50000000000000
3	0.63636363636364
4	0.69005235602094
5	0.68202041964819
6	0.68232578140989
7	0.68232780435903
8	0.68232780382802
9	0.68232780382802

试位法

0	0.00000000000000
1	1.00000000000000
2	0.50000000000000
3	0.63636363636364
4	0.67119565217391
5	0.67966164639872
6	0.68169102027289
7	0.68217581596254
8	0.68229153304816
9	0.68231914840349

抛物线法

- 如何进一步提升迭代收敛速度？

1. 方案一：提供一个初始点

$$0 = f(x) = f(x_i) + f'(x_i)(x - x_i) + \frac{1}{2}f''(x_i)(x - x_i)^2 \cancel{+ \frac{1}{3!}f'''(\xi)(x - x_i)^3}$$

2. 方案二：提供三个初始点$(a, A), (b, B), (c, C)$确定通过三点的抛物线(Muller法)：

$$y = \alpha + \beta x + \gamma x^2$$

➢与x轴出现两个交点，取离c较近的点

➢与x轴没有交点时，出现复数解

抛物线法

- 逆二次插值（Inverse Quadratic Interpolation）

$$x = P(y) = a\frac{(y-B)(y-C)}{(A-B)(A-C)} + b\frac{(y-A)(y-C)}{(B-A)(B-C)} + c\frac{(y-A)(y-B)}{(C-A)(C-B)}$$

与x轴的交点

$$P(0) = c - \frac{r(r-q)(c-b) + (1-r)s(c-a)}{(q-1)(r-1)(s-1)}$$

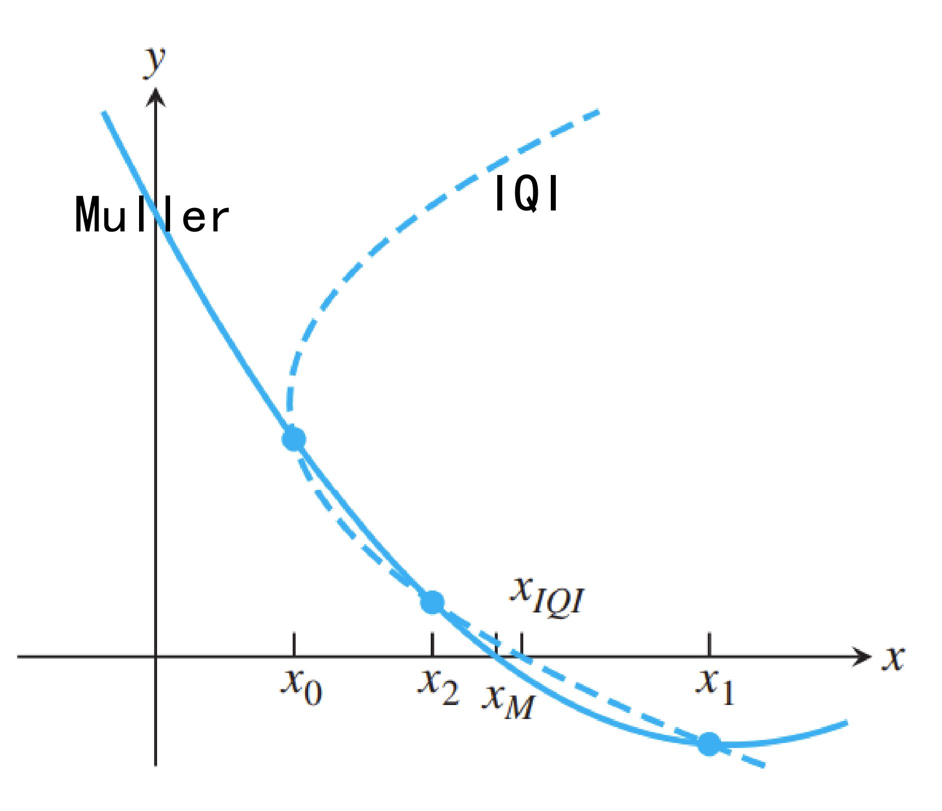
$$q = \frac{A}{B}, r = \frac{C}{B}, s = \frac{C}{A}$$

迭代格式：

$$x_{i+3} = x_{i+2} - \frac{r(r-q)(x_{i+2}-x_{i+1}) + (1-r)s(x_{i+2}-x_i)}{(q-1)(r-1)(s-1)}$$

抛物线法

- Muller方法与IQI方法的不同



Brent混合方法

- Brent混合方法的思路：兼顾收敛与稳定
  1. 记录当前点x_i（具有最优后向误差），同时记录包含根的区间$[a_i, b_i]$
  2. 尝试IQI法，如果：i.后向误差减小；ii.包含根的区间至少减半。则用新的点替换原来的一个
  3. 否则尝试割线法，如果割线法也失败则用二分法。

非线性方程组的牛顿迭代

- 包含n未知数的n个非线性方程组：

$$\begin{cases} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_2(x_1, x_2, \dots, x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{cases}$$

如何求解？

- 迭代思想：给定$X_i = \left[x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}\right]^T$递推计算$X_{i+1}$

- 线性化

$$\begin{cases} 0 = f_1(X) \approx f_1(X_i) + \dfrac{\partial f_1(X_i)}{\partial x_1}\left(x - x_1^{(i)}\right) + \dfrac{\partial f_1(X_i)}{\partial x_2}\left(x - x_2^{(i)}\right) + \cdots + \dfrac{\partial f_1(X_i)}{\partial x_n}\left(x - x_n^{(i)}\right) \\ 0 = f_2(X) \approx f_2(X_i) + \dfrac{\partial f_2(X_i)}{\partial x_1}\left(x - x_1^{(i)}\right) + \dfrac{\partial f_2(X_i)}{\partial x_2}\left(x - x_2^{(i)}\right) + \cdots + \dfrac{\partial f_2(X_i)}{\partial x_n}\left(x - x_n^{(i)}\right) \\ \vdots \\ 0 = f_n(X) \approx f_n(X_i) + \dfrac{\partial f_n(X_i)}{\partial x_1}\left(x - x_1^{(i)}\right) + \dfrac{\partial f_n(X_i)}{\partial x_2}\left(x - x_2^{(i)}\right) + \cdots + \dfrac{\partial f_n(X_i)}{\partial x_n}\left(x - x_n^{(i)}\right) \end{cases}$$

定义：$F = [f_1, f_2, \dots, f_3]^T$,Jacobi**矩阵**：$J(X) = F'(X) = \begin{pmatrix} \dfrac{\partial f_1(X)}{\partial x_1} & \cdots & \dfrac{\partial f_1(X)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_n(X)}{\partial x_1} & \cdots & \dfrac{\partial f_n(X)}{\partial x_n} \end{pmatrix}$

线性化方程可写成：$F(X_i) + J(X_i)(X - X_i) = 0$ $\boxed{\rightarrow \text{牛顿迭代：} X_{i+1} = X_i - J^{-1}(X_i)F(X_i)}$

非线性方程组的牛顿迭代

- 例：求解方程组

$$\begin{cases} f_1(u,v) = v - u^3 = 0 \\ f_2(u,v) = u^2 + v^2 - 1 = 0 \end{cases}$$

Jacobi矩阵

$$J(u,v) = \begin{bmatrix} -3u^2 & 1 \\ 2u & 2v \end{bmatrix}$$

牛顿迭代式($X = [u,v]^T, F = [f_1, f_2]$)：

$$J(u_i, v_i)(X_{i+1} - X_i) = -F(X_i)$$

取$X_0 = [1,2]$

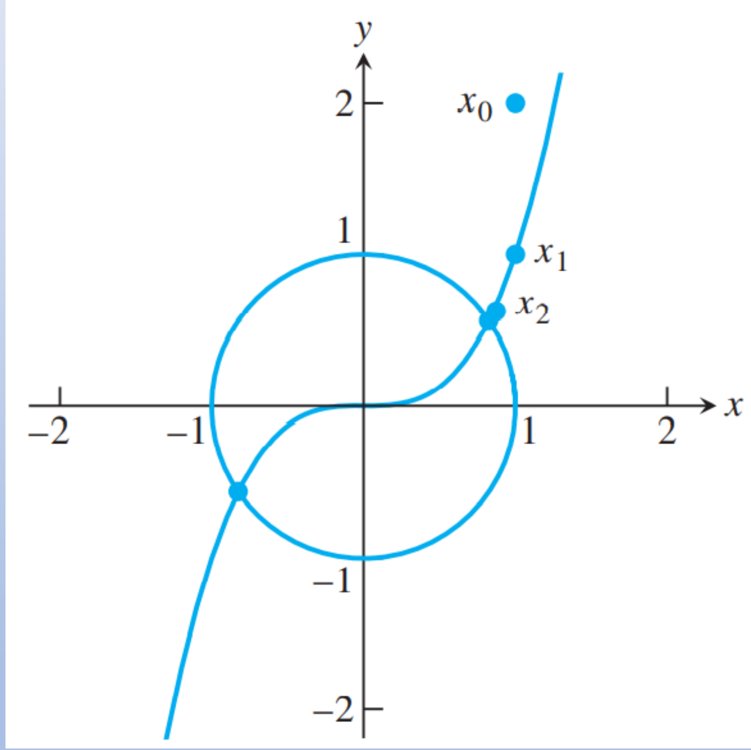
第一步迭代计算为

$$\begin{bmatrix} -3 & 1 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} u_{i+1} - u_i \\ v_{i+1} - v_i \end{bmatrix} = -\begin{bmatrix} 1 \\ 4 \end{bmatrix} \Rightarrow \begin{bmatrix} u_{i+1} \\ v_{i+1} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

非线性方程组的牛顿迭代

step	u	v
0	1.00000000000000	2.00000000000000
1	1.00000000000000	1.00000000000000
2	0.87500000000000	0.62500000000000
3	0.82903634826712	0.56434911242604
4	0.82604010817065	0.56361977350284
5	0.82603135773241	0.56362416213163
6	0.82603135765419	0.56362416216126
7	0.82603135765419	0.56362416216126

计算机迭代过程



迭代过程的图像

思考与练习

- 对以下几种计算$\sqrt[4]{2}$的算法的收敛速率从最快到最慢排序，并说明理由

1. 使用二分法求解$f(x)=x^4-2=0$
2. 使用割线法求解$f(x)=x^4-2=0$
3. 通过迭代求$g(x)=\frac{x}{2}+\frac{1}{x^3}$的不动点
4. 通过迭代求$g(x)=\frac{5x}{6}+\frac{1}{3x^3}$的不动点
5. 使用牛顿法求解$f(x)=x^4-2=0$

- 如果$f(x)\in C^m(\mathbb{R})$，并满足$f(r)=0,\ f'(r)\neq 0, f''(r)=\cdots=f^{(m-1)}(r)=0,$ $f^{(m)}(r)\neq 0\ (m>2)$,试分析标准牛顿迭代法的收敛性及误差。

练习与思考

- 找出函数$g(x) = x^2 - \frac{3}{2}x + \frac{3}{2}$的每个不动点，并确定不动点迭代是否局部收敛

- 令$f(x) = x^4 - 7x^3 + 18x^2 - 20x + 8$, 牛顿法是否会二次收敛到根$r = 2$？确定$\lim_{i\to\infty} e_{i+1}/e_i$

- 函数$f(x) = x^3 - 4x$。(a) 假设使用牛顿法求根$r = 2$, 第4步后误差是$e_4 = 10^{-6}$, 估计e_5；(b) 假设使用牛顿法求根$r = 0$, 第4步后误差是$e_4 = 10^{-6}$, 估计e_5；

- 取$(u_0, v_0) = (1,1)$，使用牛顿迭代进行两步迭代求解

$$\begin{cases} u^2 + v^2 = 1 \\ (u-1)^2 + v^2 = 1 \end{cases}$$

- 编程：使用二分法、不动点迭代、牛顿法、试位法求$\mathrm{e}^x + x = 7$的根, 结果精确到小数点后8位，并对各方法的收敛性进行比较。