FDU 数值算法 1. 非线性方程的解法

本文根据邵美悦老师课堂笔记整理而成，并参考以下教材:

- Numerical Algorithms Methods for Computer Vision, Machine Learning, and Graphics (J. Solomon) Chapter 8
- 数值逼近 (蒋尔雄 & 赵风光 & 苏仰峰, 第 2 版) 第 7 章
- 数值分析 (李庆扬 & 王能超 & 易大义, 第 5 版) 第 7 章
- 工科泛函分析基础 (孙明正、李冶岸、张建国、邹杰涛) 第 2 章

欢迎批评指正!

1.1 单变量情形

1.1.1 二分法

(中值定理, Numerical Algorithms Methods for Computer Vision 定理 8.1)
若 $f:[a,b]\mapsto\mathbb{R}$ 是连续函数，且有 $f(a)<u<f(b)$ 或 $f(a)>u>f(b)$ 成立，
则一定存在 $x_\star\in(a,b)$ 使得 $f(x_\star)=u$
(可使用闭区间套定理证明)

二分法 (bisection) 是求连续函数 $f(x)$ 实根的可靠方法.
首先找到 l_1,r_1 (假定 $l_1<r_1$) 使得 $f(l_1)f(r_1)<0$ (即 $f(l_1)$ 和 $f(r_1)$ 异号)
根据 $f(x)$ 的连续性可知 (l_1,r_1) 中必然存在 $f(x)$ 的零点.
计算 $x_1=\frac{l_1+r_1}{2}$ 和 $f(x_1)$，有以下三种可能:

- ① 若 $f(x_1)=0$，则 x_1 即为所求的根，计算终止;
  (实际计算中验证的是 $|f(x_1)|\le\varepsilon_f$ 或 $|r_1-l_1|\le\varepsilon_x$，其中 ε_f 和 ε_x 是预定义的阈值)
- ② 若 $f(l_1)f(x_1)<0$，则取 $\begin{cases}l_2=l_1\\r_2=x_1\end{cases}$
- ③ 若 $f(l_1)f(x_1)>0$，则取 $\begin{cases}l_2=x_1\\r_2=r_1\end{cases}$

若出现 ②③ 情况，则对 (l_2,r_2) 重复上述过程，以此类推，即为二分法.

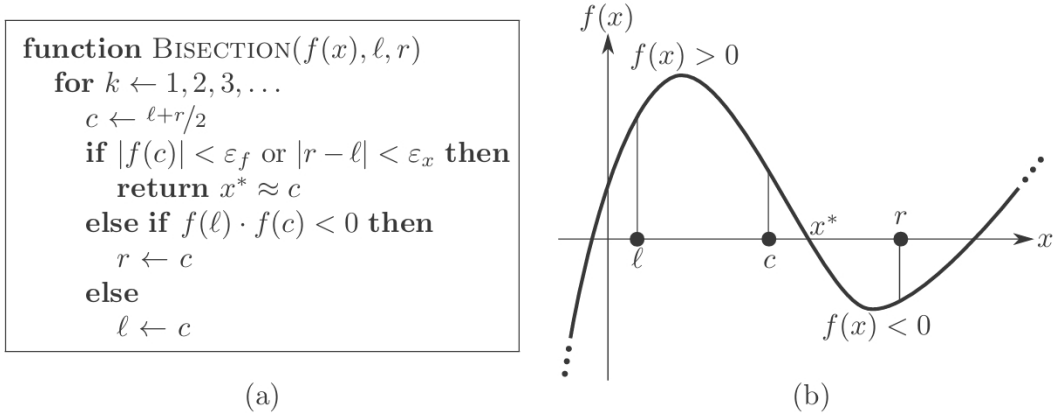


Figure 8.1 (a) Pseudocode and (b) an illustration of the bisection algorithm for finding roots of continuous $f(x)$ given endpoints $\ell,r\in\mathbb{R}$ with $f(\ell)\cdot f(r)<0$. The interval $[c,r]$ contains a root x^* because $f(c)$ and $f(r)$ have opposite sign.

二分法的优缺点:

- 优点: 近似解 x_k 的精确程度很清楚，十分可靠 (即保证收敛)，且便于并行计算几个根.
  (双精度浮点数下的机器精度为 2^{-53}，因此二分法至多迭代五十多步)
- 缺点: 二分法收敛速度比较慢 (常数为 $\frac{1}{2}$ 的线性收敛)
  且在不了解 $f(\cdot)$ 的情况下，找到 l_1,r_1 使得 $f(l_1)f(r_1)<0$ 有时会很困难 (不亚于求根问题)
  (Homework 2 Problem 5)
  此外，当 $f(l)$ 和 $f(r)$ 至少有一个比较小时，相对误差会比较大，导致符号判断不准确.

值得说明的是，我们很害怕常数接近于 1 的线性收敛 (例如 $\gamma = 0.9999$)
因为 $(1 - \frac{1}{n})^n \to e^{-1}\ (n \to \infty)$，因此 $0.9999^{10000} \approx e^{-1} = 0.368$

看到小量时要当心，因为它们有可能是两个很大的量相减得到的.
不过有一部分函数值可以算得很准，
例如使用 FMA (一次性计算 $ab \pm c$) 计算 $x^2 - 2$ 只有 eps 级别的舍入误差，
计算二阶行列式 $ab - cd$ 只有 2eps 级别的舍入误差 (Kahan 算法，关键是利用 FMA)
这样能保证它们的符号是算得准的.
如果函数值在基本精度里算得不准，就要加精度了 (使用某些库函数)

有时计算函数值很贵 (例如 Lambert W 函数)，或者我们默认它是贵的.
因此在二分法中，我们尽可能将已计算的函数值存下来.
此外，当函数值或绝对误差很小时，二分法迭代就可以停止了.

试位法 (regula falsi 法)
它与二分法唯一的不同在于，其第 k 步的近似解为:

$$x_k = \frac{1}{2}(u_k + l_k)$$
$$\Downarrow$$
$$x_k = u_k - \frac{f(u_k)(u_k - l_k)}{f(u_k) - f(l_k)} = \frac{f(u_k)l_k - f(l_k)u_k}{f(u_k) - f(l_k)}$$

本质就是割线与水平坐标轴的交点.
这个想法是合理的，因为当 u_k 和 l_k 很接近时，点 $(l_k, f(l_k))$ 和 $(u_k, f(u_k))$ 之间的函数图像可近似看作一条直线段.

1.1.2 不动点迭代

如果我们对 $f(\cdot)$ 有更多信息的话，我们就可以构造收敛速度更快的算法.
假设 $f(x)$ 可以表示为 $f(x) = g(x) - x$
其中 $g(\cdot)$ 是 c-Lipschitz 连续的，且常数 c 满足 $0 \le c < 1$
换言之，$g(\cdot)$ 是一个压缩映射，即对于任意 $x, y \in \mathrm{dom}(g)$ 都有 $|g(x) - g(y)| < |x - y|$ 成立.
于是 $f(x) = 0$ 的求根问题就变为 $g(x) = x$ 的求根问题:

$$\frac{\text{Take } x_0 \text{ as the initial guess of } x_\star,\ \text{set } k = 1}{\text{while convergence criteria is not reached, loop:}}$$
$$x_k = g(x_{k-1})$$
$$k = k + 1$$

若迭代序列 $\{x_k\}$ 收敛到 $g(\cdot)$ 的不动点 $x_\star$ (即满足 $g(x_\star) = x_\star$)，
则 $x_\star$ 即为 $f(\cdot)$ 的根 (即满足 $f(x_\star) = 0$)

> **(压缩映射定理, 又称 Banach 不动点定理, 工科泛函分析基础, 定理 $2.5.3$)**
> 设 (X, d) 为完备度量空间.
> 若 $T : X \mapsto X$ 是一个压缩映射，则 T 在 X 中有唯一的不动点.

只要 $g(\cdot)$ 存在不动点 $x_\star$，Lipschitz 常数 $c < 1$ 的 Lipschitz 连续性就能保证上述迭代格式的收敛性:
(可惜的是，呈现出的仍然是**线性收敛性**，和二分法一样)

$$\begin{aligned}
|x_k - x_\star| &= |g(x_{k-1}) - g(x_\star)| && (\text{note that } x_k = g(x_{k-1}) \text{ and } x_\star = g(x_\star)) \\
&\le c|x_{k-1} - x_\star| && (\text{since } g(\cdot) \text{ is Lipschitz continuous with constant } c \in (0, 1)) \\
&\le c^2|x_{k-2} - x_\star| \\
&\le \cdots \\
&\le c^k|x_0 - x_\star| \to 0 \quad (k \to \infty)
\end{aligned}$$

但如果 Lipschitz 常数过大 (即 $g(\cdot)$ 的斜率过大)，则不动点迭代可能会发散.
(只要初值不是 $x_\star$，误差就会随着迭代被不断放大)

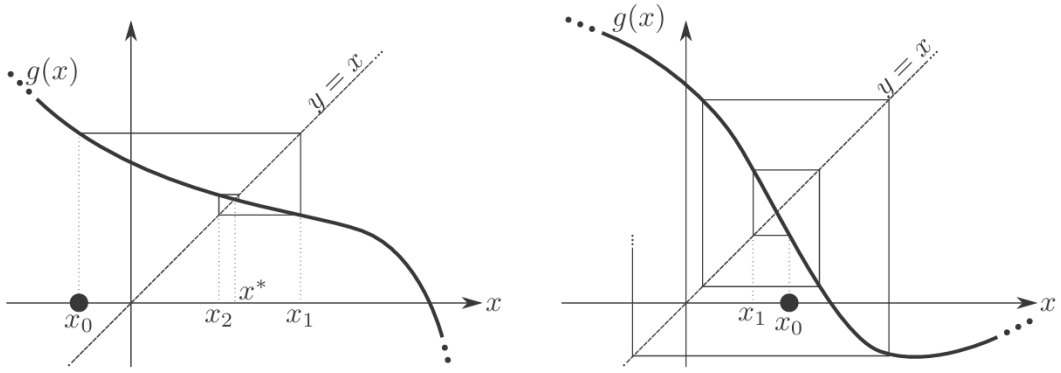


Figure 8.2 Convergence of fixed point iteration. Fixed point iteration searches for the intersection of $g(x)$ with the line $y = x$ by iterating $x_k = g(x_{k-1})$. One way to visualize this method on the graph of $g(x)$ visualized above is that it alternates between moving horizontally to the line $y = x$ and vertically to the position $g(x)$. Fixed point iteration (a) converges when the slope of $g(x)$ is small and (b) diverges otherwise.

若 $g(\cdot)$ 在 $[x_\star - \delta, x_\star + \delta]$ 中 c-Lipschitz 连续，且 $0 \le c < 1$，
则只要 $x_0 \in [x_\star - \delta, x_\star + \delta]$，不动点迭代序列 $\{x_k\}$ 就会收敛到 $x_\star$
一个简单的例子:
当 $g(\cdot)$ 一阶连续可导且 $|g'(x_\star)| < 1$ 时，根据 $g'(\cdot)$ 的连续性可知:
对于任意 $\varepsilon > 0$ 都存在 $\delta > 0$ 使得 $|g'(x)| < 1 - \varepsilon$ $(\forall\, x \in (x_\star - \delta, x_\star + \delta))$ 成立.
于是我们有:

$$\begin{aligned} |g(x) - g(y)| &= |g'(\theta)| \cdot |x - y| \quad (\text{Mean Value Theorem, for some } \theta \in [x, y]) \\ &< (1 - \varepsilon) \cdot |x - y| \end{aligned}$$

即 $g(\cdot)$ 在 $(x_\star - \delta, x_\star + \delta)$ 上 $(1 - \varepsilon)$-Lipschitz 连续.
因此只要 x_0 离 $x_\star$ 足够近，不动点迭代序列 $\{x_k\}$ 就会收敛到 $x_\star$.

在某些特殊情况下，不动点迭代可能表现为局部超线性收敛.
例如当 $g(\cdot)$ 二阶连续可导且 $g'(x_\star) = 0$ 时，不动点迭代局部二次收敛:

$$\begin{aligned} |x_k - x_\star| &= |g(x_{k-1}) - g(x_\star)| \quad (\text{note that } x_k = g(x_{k-1}) \text{ and } x_\star = g(x_\star)) \\ &= \left| g(x_\star) + \frac{1}{2} g''(x_\star)(x_{k-1} - x_\star)^2 + O((x_{k-1} - x_\star)^3) - g(x_\star) \right| \quad (\text{Taylor expansion, note that } g'(x_\star) = 0) \\ &\le \frac{1}{2}(|g''(x_\star)| + \varepsilon)|x_{k-1} - x_\star|^2 \text{ for some } \varepsilon \text{ so long as } x_{k-1} \text{ is close to } x_\star \end{aligned}$$

Example 8.2 (Fixed point iteration)**.** We can apply fixed point iteration to solving $x = \cos x$ by iterating $x_{k+1} = \cos x_k$. A numerical example starting from $x_0 = 0$ proceeds as follows:

k	0	1	2	3	4	5	6	7	8	9
x_k	0	1.000	0.540	0.858	0.654	0.793	0.701	0.764	0.722	0.750

In this case, fixed point iteration converges linearly to the root $x^* \approx 0.739085$.

The root-finding problem $x = \sin x^2$ satisfies the condition for quadratic convergence near $x^* = 0$. For this reason, fixed point iteration $x_{k+1} = \sin x_k^2$ starting at $x_0 = 1$ converges more quickly to the root:

k	0	1	2	3	4	5	6	7	8	9
x_k	1	0.841	0.650	0.410	0.168	0.028	0.001	0.000	0.000	0.000

Finally, the roots of $x = e^x + e^{-x} - 5$ do *not* satisfy convergence criteria for fixed point iteration. Iterates of the failed fixed point scheme $x_{k+1} = e^{x_k} + e^{-x_k} - 5$ starting at $x_0 = 1$ are shown below:

k	0	1	2	3	4	5	6	7
x_k	1	-1.914	1.927	2.012	2.609	8.660	5760.375	$\cdots$

1.1.3 Newton 法

假设 $f(\cdot)$ 一阶连续可导，则 x_k 处的近似值为:

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

令右式等于零并解出 x 即得 Newton 法的迭代格式:

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

定义 $g(x) := x - \frac{f(x)}{f'(x)}$，则 Newton 法相当于 $g(\cdot)$ 的不动点迭代.

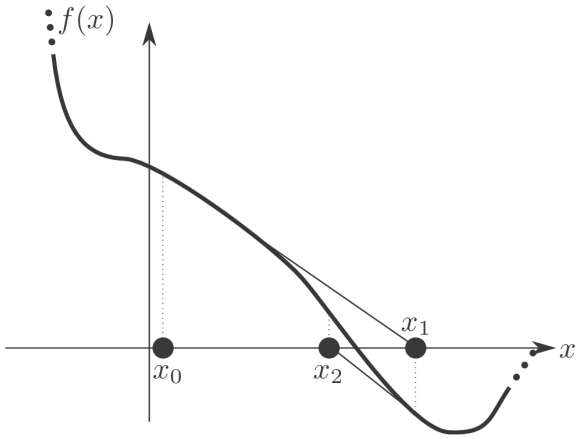


Figure 8.3 Newton's method iteratively approximates $f(x)$ with tangent lines to find roots of a differentiable function $f(x)$.

假设 $f(\cdot)$ 二阶连续可导，考虑 $g(\cdot)$ 的导函数:

$$g'(x) = 1 - \frac{(f'(x))^2 - f(x)f''(x)}{(f'(x))^2} = \frac{f(x)f''(x)}{(f'(x))^2}$$

若 $x_\star$ 是 $f(x)$ 的一个单重根，则 $f(x_\star) = 0$ 且 $f'(x_\star) \neq 0$
于是我们有:

$$g'(x_\star) = \frac{f(x_\star)f''(x_\star)}{(f'(x_\star))^2} = 0$$

根据 1.1.2 的结论可知 Newton 法局部二次收敛.
但若 $x_\star$ 是 $f(x)$ 的多重根 (这就需要变形或其他处理，参考 Homework 02 Problem 03 (c)),
则 Newton 法的收敛性会退化为线性收敛 (甚至更糟)

Example 8.3 (Newton's method)**.** The last part of Example 8.2 can be expressed as a root-finding problem on $f(x) = e^x + e^{-x} - 5 - x$. The derivative of $f(x)$ in this case is $f'(x) = e^x - e^{-x} - 1$, so Newton's method can be written

$$x_{k+1} = x_k - \frac{e^{x_k} + e^{-x_k} - 5 - x_k}{e^{x_k} - e^{-x_k} - 1}.$$

This iteration quickly converges to a root starting from $x_0 = 2$:

k	0	1	2	3	4
x_k	2	1.9161473	1.9115868	1.9115740	1.9115740

Example 8.4 (Newton's method failure)**.** Suppose $f(x) = x^5 - 3x^4 + 25$. Newton's method applied to this function gives the iteration

$$x_{k+1} = x_k - \frac{x_k^5 - 3x_k^4 + 25}{5x_k^4 - 12x^3}.$$

These iterations converge when x_0 is sufficiently close to the root $x^* \approx -1.5325$. For instance, the iterates starting from $x_0 = -2$ are shown below:

k	0	1	2	3	4
x_k	-2	-1.687500	-1.555013	-1.533047	-1.532501

Farther away from this root, however, Newton's method can fail. For instance, starting from $x_0 = 0.25$ gives a divergent set of iterates:

k	0	1	2	3	4
x_k	0.25	149.023256	119.340569	95.594918	76.599025

有趣的事实:

- ① 有工程界的人说三角函数中的角度制没有存在的必要，他所看到的应用场景都用的是弧度制.
  这种说法是不对的，例如进行 36 次 10° 顺时针旋转最终会转回来 (整数运算没有舍入误差),
  但进行 36 次 $\frac{\pi}{18}$ 顺时针旋转会由于舍入误差的影响而转不回来 (浮点数运算有舍入误差)
- ② 现在的 CPU 和 GPU 支持 $f(x) = \frac{1}{\sqrt{x}}$ 的运算.
  曾经有人开发出快速算法 (Fast inverse square root in Quake III Arena):

```
float InvSqrt(float x)
{
    float x_half = 0.5f * x;
    int i = *(int*)&x;              // get bits for floating value
    i = 0x5f3759df - (i>>1);        // gives initial guess
    x = *(float*)&i;                // convert bits back to float
    x = x*(1.5f - x_half*x*x);      // Newton step, repeating increases accuracy
    return x;
}
```

下面我们推导 Newton 迭代格式:

$$
\begin{aligned}
y = \frac{1}{\sqrt{\alpha}} = \alpha^{-\frac{1}{2}} \quad (\alpha > 0) \\
\Updownarrow \\
y^{-2} - \alpha = 0 \\
\Updownarrow \\
y_{k+1} = y_k - \frac{y_k^{-2} - \alpha}{-2y_k^{-3}} \\
= y_k + \frac{1}{2}(y_k - \alpha y_k^3) \\
= y_k \left(\frac{3}{2} - \frac{1}{2}\alpha y_k^2 \right)
\end{aligned}
$$

(Schröder's Method)
Newton 法的修正，即对 $\frac{f}{f'}$ 应用 Newton 法:

$$
\begin{aligned}
x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} \Big\backslash \frac{f'(x_n) \cdot f'(x_n) - f(x_n) \cdot f''(x_n)}{(f'(x_n))^2} \\
&= x_n - \frac{f(x_n) f'(x_n)}{(f'(x_n))^2 - f(x_n) f''(x_n)}
\end{aligned}
$$

Newton 法的一种简化策略是用 $f'(x_0)$ 代替 $f'(x_k)$:

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

这可能会牺牲收敛性，通常表现为局部线性收敛.

1.1.4 割线法

在某些情况下，计算导数是很昂贵的.

> **Example 8.5** (Rocket design)**.** Suppose we are designing a rocket and wish to know how much fuel to add to the engine. For a given number of gallons x, we can write a function $f(x)$ giving the maximum height of the rocket during flight; our engineers have specified that the rocket should reach a height h, so we need to solve $f(x) = h$. Evaluating $f(x)$ involves simulating a rocket as it takes off and monitoring its fuel consumption, which is an expensive proposition. Even if f is differentiable, we might not be able to evaluate f' in a practical amount of time.

我们可以使用差分来近似导数:

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

由于我们在上一轮迭代中已经计算了 $f(x_{k-1})$，故上述近似的开销很低.
当 $\{x_k\}$ 接近收敛并且彼此相近时，上述近似的效果很好.
将其代入 Newton 法的迭代格式中，便得到了**割线法** (Secant Method):

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

不过我们必须提供 x_0 和 x_1 两个初值 (其中 x_1 可通过对 x_0 应用一次 Newton 法迭代得到)

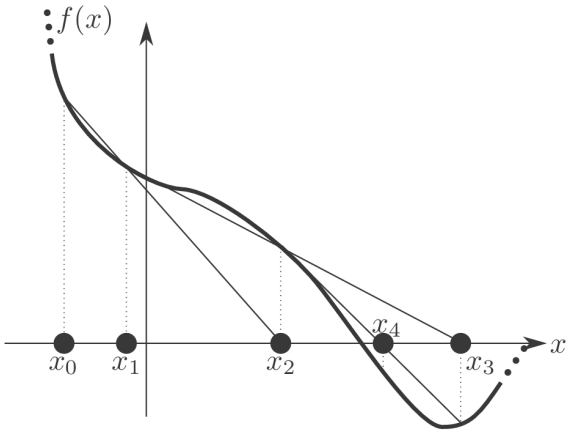


Figure 8.4 The secant method is similar to Newton's method (Figure 8.3) but approximates tangents to $f(x)$ as the lines through previous iterates. It requires both x_0 and x_1 for initialization.

收敛性分析表明，割线法以 $\frac{1+\sqrt{5}}{2} \approx 1.618$ (黄金分割比例) 的速率收敛，这介于线性收敛和二次收敛之间.
由于割线法的收敛性接近于 Newton 法，且节省了计算导数 $f'(\cdot)$ 的开销，因此它是 Newton 法强有力的替代.
(但如果初值 x_0 离 $x_\star$ 较远，则割线法可能发散)

Example 8.6 (Secant method)**.** Suppose $f(x) = x^4 - 2x^2 - 4$. Iterates of Newton's method for this function are given by

$$x_{k+1} = x_k - \frac{x_k^4 - 2x_k^2 - 4}{4x_k^3 - 4x_k}.$$

Contrastingly, iterates of the secant method for the same function are given by

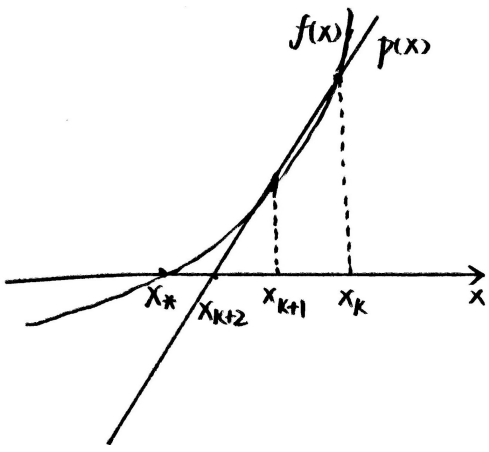
$$x_{k+1} = x_k - \frac{(x_k^4 - 2x_k^2 - 4)(x_k - x_{k-1})}{(x_k^4 - 2x_k^2 - 4) - (x_{k-1}^4 - 2x_{k-1}^2 - 4)}.$$

By construction, a less expensive way to compute these iterates is to save and reuse $f(x_{k-1})$ from the previous iteration. We can compare the two methods starting from $x_0 = 3$; for the secant method we also choose $x_{-1} = 2$:

k	0	1	2	3	4	5	6
x_k (Newton)	3	2.385417	2.005592	1.835058	1.800257	1.798909	1.798907
x_k (secant)	3	1.927273	1.882421	1.809063	1.799771	1.798917	1.798907

The two methods exhibit similar convergence on this example.

割线法的收敛性: (reference)



设 $f(x)$ 的一个根为 $x_\star$
根据 $(x_k, f(x_k)), (x_{k+1}, f(x_{k+1}))$ 构造的割线记为 $p(x)$，与 x 轴交点记为 x_{k+2}
则根据 Lagrange 中值定理我们有:

$$p(x)=\frac{f(x_{k+1})-f(x_k)}{x_{k+1}-x_k}(x-x_{k+2})=f'(\eta)(x-x_{k+2})\text{ where }\eta\in(x_k,x_{k+1})$$

这相当于根据节点 $(x_k,f(x_k)),(x_{k+1},f(x_{k+1}))$ 对 $f(x)$ 进行一次多项式插值.
根据 **FDU 数值算法 2. 插值与拟合**的结论，插值误差如下:

$$f(x)-p(x)=\frac{f''(\xi)}{2}(x-x_k)(x-x_{k+1})\text{ where }\xi\in(x_k,x_{k+1})$$

代入 $x=x_\star$ 可得:

$$\begin{gathered}f(x_\star)-p(x_\star)=\frac{f''(\xi)}{2}(x_\star-x_k)(x_\star-x_{k+1})\\\Updownarrow\\0-f'(\eta)(x_\star-x_{k+2})=\frac{f''(\xi)}{2}(x_\star-x_k)(x_\star-x_{k+1})\\\Updownarrow\\\varepsilon_{k+2}=\frac{f''(\xi)}{2f'(\eta)}\varepsilon_k\varepsilon_{k+1}\\\text{where }\varepsilon_k:=|x_\star-x_k|\end{gathered}$$

于是我们有:

$$\begin{gathered}\varepsilon_{k+2}\le M\varepsilon_k\varepsilon_{k+1}\\\text{where }M:=\frac{\sup_{x_k<x<x_{k+1}}f''(x)}{2\inf_{x_k<x<x_{k+1}}f'(x)}\\\Downarrow\\M\varepsilon_{k+2}\le(M\varepsilon_{k+1})(M\varepsilon_k)\\\Downarrow\\\log(M\varepsilon_{k+2})\le\log(M\varepsilon_{k+1})+\log(M\varepsilon_k)\end{gathered}$$

定义 Fibonacci 递推数列: (注意初值 a_0,a_1 不是 $0,1$)

$$\begin{gathered}a_0=\log(M\varepsilon_0)\\a_1=\log(M\varepsilon_1)\\\text{suppose that }a_0,a_1<0\\a_{n+2}=a_{n+1}+a_n\ (n\in\mathbb{N})\end{gathered}$$

显然有 $\log(M\varepsilon_n)\le a_n\ (n\in\mathbb{N})$ 成立.
容易验证数列 $\{a_n\}$ 的渐近增长率为 $\frac{1+\sqrt5}{2}\approx1.618$:

$$\begin{gathered}\text{Denote }\phi:=\lim_{n\to\infty}\frac{a_{n+1}}{a_n}\\\text{Note that }\phi=\lim_{n\to\infty}\frac{a_{n+1}}{a_n}=\lim_{n\to\infty}\frac{a_n+a_{n-1}}{a_n}=1+\lim_{n\to\infty}\frac{a_{n-1}}{a_n}=1+\frac{1}{\phi}\\\text{By solving }\phi^2=\phi+1\text{ we obtain }\phi_{1,2}=\frac{1\pm\sqrt5}{2}\\\text{Therefore }\lim_{n\to\infty}\frac{a_{n+1}}{a_n}=\frac{1+\sqrt5}{2}\end{gathered}$$

因此 $\{\varepsilon_n\}$ 至少是以 $\frac{1+\sqrt5}{2}$ 的速率超线性收敛于 0 的 **(存疑)**

> 我们可以推导数列 $\{a_n\}$ 的通项公式:

$$
\begin{aligned}
\begin{bmatrix} a_n \\ a_{n-1} \end{bmatrix} &= \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a_{n-1} \\ a_{n-2} \end{bmatrix} \\
&= \cdots \\
&= \left(\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \right)^{n-1} \begin{bmatrix} a_1 \\ a_0 \end{bmatrix} \\
&= \begin{bmatrix} \phi_1 & \phi_2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \phi_1^{n-1} & \\ & \phi_2^{n-1} \end{bmatrix} \begin{bmatrix} \phi_1 & \phi_2 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} a_1 \\ a_0 \end{bmatrix} \\
&= \begin{bmatrix} \phi_1 & \phi_2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \phi_1^{n-1} & \\ & \phi_2^{n-1} \end{bmatrix} \left(\frac{1}{\phi_1 - \phi_2} \begin{bmatrix} 1 & -\phi_2 \\ -1 & \phi_1 \end{bmatrix} \right) \begin{bmatrix} a_1 \\ a_0 \end{bmatrix} \\
&= \begin{bmatrix} \phi_1^n & \phi_2^n \\ \phi_1^{n-1} & \phi_2^{n-1} \end{bmatrix} \left(\frac{1}{\phi_1 - \phi_2} \begin{bmatrix} a_1 - a_0\phi_2 \\ a_0\phi_1 - a_1 \end{bmatrix} \right) \\
&= \frac{1}{\phi_1 - \phi_2} \begin{bmatrix} a_1(\phi_1^n - \phi_2^n) - a_0\phi_1\phi_2(\phi_1^{n-1} - \phi_2^{n-1}) \\ a_1(\phi_1^{n-1} - \phi_2^{n-1}) - a_0\phi_1\phi_2(\phi_1^{n-2} - \phi_2^{n-2}) \end{bmatrix} \quad (\text{note that } \phi_1\phi_2 = -1) \\
&= \frac{1}{\phi_1 - \phi_2} \begin{bmatrix} a_1(\phi_1^n - \phi_2^n) + a_0(\phi_1^{n-1} - \phi_2^{n-1}) \\ a_1(\phi_1^{n-1} - \phi_2^{n-1}) + a_0(\phi_1^{n-2} - \phi_2^{n-2}) \end{bmatrix}
\end{aligned}
$$

$$
\left(\text{where } \begin{cases} \phi_1 = \frac{1+\sqrt{5}}{2} \\ \phi_2 = \frac{1-\sqrt{5}}{2} \end{cases} \text{ are the solutions to the characteristic equation } \phi^2 = \phi + 1 \right)
$$

因此 $a_n = \frac{a_1(\phi_1^n - \phi_2^n) + a_0(\phi_1^{n-1} - \phi_2^{n-1})}{\phi_1 - \phi_2}$

Steffensen 方法以另一种方式近似导数:

$$
f'(x_k) \approx \frac{f(x_k + f(x_k)) - f(x_k)}{f(x_k)}
$$

于是迭代格式变为:

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

(Homework 2 Problem 6 & Final Exam Problem 2)
其收敛速度仍为局部二次收敛，每轮迭代不需计算导数值，但需要计算两次函数值.

1.1.5 Dekkers 方法

我们可以结合不同根求解算法的优点，得到一种混合算法.
注意到:

- 二分法保证全局线性收敛;
- 割线法具有更快的收敛速率 (速率为 $\frac{1+\sqrt{5}}{2} \approx 1.618$)，但如果初值 x_0 离 $x_\star$ 较远，则割线法可能发散;

假设 $f(\cdot)$ 的一个根落在区间 $[l_k, r_k]$ 的内部.
给定近似解 x_{k-1} 和 x_k，则我们可以这样计算下一个近似解 x_{k+1}:

- ① 进行一步割线法迭代，得到 $\tilde{x}_{k+1} := x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})} = \frac{f(x_k)x_{k-1} - f(x_{k-1})x_k}{f(x_k) - f(x_{k-1})}$
  这看上去很有道理，因为当 x_k 和 x_{k-1} 很近时，我们可以近似认为那一段函数图像是直线.
- ② 若 $\tilde{x}_{k+1}$ 落在 (l_k, r_k) 里，则取 $x_{k+1} = \tilde{x}_{k+1}$; 否则取 $x_{k+1} = \frac{l_k + r_k}{2}$

最后和二分法一样，根据 $f(x_{k+1})$ 的符号来选取 l_{k+1} 和 r_{k+1}.

上述混合方法称为 **Dekkers 方法**.
它试图将二分法的全局收敛性与割线法的更强根估计相结合.
在许多情况下，它是成功的，但其收敛速率比较难分析.
在特定的例子上，该方法的收敛性可能会降至线性收敛或更差.

1.2 多变量情形

多变量的问题要复杂得多，像二分法这样的策略难以扩展.

1.2.1 Newton 法

假设映射 $f:\mathbb{R}^n\mapsto\mathbb{R}^m$ 连续可导，则 f 在 x_k 处的一阶 Taylor 近似为:

$$f(x)\approx f(x_k)+Df(x_k)(x-x_k)\text{ where }Df(x):=\left[\frac{\partial f_i(x)}{\partial x_j}\right]\in\mathbb{R}^{m\times n}$$

令右式 $=0_m$ 即得到:

$$Df(x_k)(x_{k+1}-x_k)=-f(x_k)$$

- ① 当 $m<n$ 时，可能有多个解，我们可使用 Moore-Penrose 伪逆找到其中一个解. **(存疑)**
- ② 当 $m>n$ 时，可以尝试最小二乘解，但这种方法很可能不收敛.
- ③ 当 $m=n$ 且 $Df(x_k)$ 非奇异时，我们有:

  $$x_{k+1}=x_k-[Df(x_k)]^{-1}f(x_k)$$

  实际计算时我们通过求解线性方程组来代替逆矩阵 $[Df(x_k)]^{-1}$:

  $$\begin{gathered}\text{Solve }Df(x_k)\cdot z=f(x_k)\text{ to obtain }z_k=[Df(x_k)]^{-1}f(x_k)\\x_{k+1}=x_k-z_k\end{gathered}$$

  定义 $g(x):=x-[Df(x)]^{-1}f(x)$，其 Jacobi 矩阵为: **(存疑)**

  $$\begin{aligned}Dg(x)&=I_n+[Df(x)]^{-1}D^2f(x)[Df(x)]^{-1}f(x)-[Df(x)]^{-1}Df(x)\\&=[Df(x)]^{-1}D^2f(x)[Df(x)]^{-1}f(x)\end{aligned}$$

  若 $Dg(x_\star)$ 的谱半径小于 1，则只要初值 x_0 离 $x_\star$ 足够近，Newton 法就一定收敛;
  若 $Df(x_\star)$ 非奇异，则 Newton 法局部二次收敛.

阻尼 Newton 法 (Damped Newton's Method)

$$\begin{gathered}\text{Solve }Df(x_k)\cdot z=f(x_k)\text{ to obtain }z_k=[Df(x_k)]^{-1}f(x_k)\\x_{k+1}=x_k-t_kz_k\end{gathered}$$

我们可以通过动态调整步长 t_k $(0<t_k\leq1)$，从而克服一些全局收敛性的障碍.

1.2.2 拟 Newton 法

我们可以通过某种近似来节省计算 Jacobi 矩阵 $Df(x)$ 的开销.
设非奇异阵 J_k 是对 $Df(x_k)$ 的近似，设 x_{k-1},J_{k-1} 和 J_{k-1}^{-1} 已知，并记:

$$\begin{aligned}x_k&=x_{k-1}-J_k^{-1}f(x_{k-1})\\\Delta x_k&=x_k-x_{k-1}\\\Delta f_k&=f(x_k)-f(x_{k-1})\end{aligned}$$

我们令它满足割线方程:

$$\begin{gathered}J_k\Delta x_k=\Delta f_k\\\Updownarrow\\\Delta x_k=J_k^{-1}\Delta f_k\end{gathered}$$

如果 J_k 不满秩，则 J_k 有无穷多种可能，因此我们要对它做一定限制.

(Good Broyden's Method)
Broyden 令 J_k 是 J_{k-1} 的秩 1 修正 (即 $J_k=J_{k-1}+uv^{\mathrm{T}}$)
代入第一种割线方程 $J_k\Delta x_k=\Delta f_k$ 即得:

$$uv^{\mathrm{T}}\Delta x_k=\Delta f_k-J_{k-1}\Delta x_k$$

容易看出 u 与向量 $\Delta f_k - J_{k-1}\Delta x_k$ 线性相关.
令 $v = \Delta x_k$ (这可保证 v 与 Δx_k 的内积为正值) 即得 $u = \frac{\Delta f_k - J_{k-1}\Delta x_k}{\|\Delta x_k\|^2}$
事实上，这样取相当于用 Moore-Penrose 逆求解:

$$\begin{aligned} uv^{\mathrm{T}} &= (\Delta f_k - J_{k-1}\Delta x_k)(\Delta x_k)^{\dagger} \\ &= (\Delta f_k - J_{k-1}\Delta x_k) \cdot \frac{\Delta x_k^{\mathrm{T}}}{\Delta x_k^{\mathrm{T}}\Delta x_k} \\ &= \frac{(\Delta f_k - J_{k-1}\Delta x_k)\Delta x_k^{\mathrm{T}}}{\|\Delta x_k\|^2} \end{aligned}$$

于是我们有:

$$\begin{aligned} J_k &= J_{k-1} + uv^{\mathrm{T}} \\ &= J_{k-1} + \frac{(\Delta f_k - J_{k-1}\Delta x_k)\Delta x_k^{\mathrm{T}}}{\|\Delta x_k\|^2} \\ &= \arg\min_{\mathrm{rank}(J-J_{k-1})\le 1} \|J - J_{k-1}\|_{\mathrm{F}} \quad (\text{Properties of the Moore-Penrose inverse}) \end{aligned}$$

根据 Sherman-Morrison-Woodbury 公式可知:

$$\begin{aligned} J_k^{-1} &= J_{k-1}^{-1} - \frac{J_{k-1}^{-1}uv^{\mathrm{T}}J_{k-1}^{-1}}{1 + v^{\mathrm{T}}J_{k-1}^{-1}u} \\ &= J_{k-1}^{-1} - \frac{J_{k-1}^{-1} \cdot \frac{\Delta f_k - J_{k-1}\Delta x_k}{\|\Delta x_k\|^2}\Delta x_k^{\mathrm{T}} \cdot J_{k-1}^{-1}}{1 + \Delta x_k^{\mathrm{T}}J_{k-1}^{-1} \cdot \frac{\Delta f_k - J_{k-1}\Delta x_k}{\|\Delta x_k\|^2}} \\ &= J_{k-1}^{-1} + \frac{(\Delta x_k - J_{k-1}^{-1}\Delta f_k)\Delta x_k^{\mathrm{T}}J_{k-1}^{-1}}{\Delta x_k^{\mathrm{T}}J_{k-1}^{-1}\Delta f_k} \end{aligned}$$

多步秩 1 修正可以变为秩 k 修正，用 Sherman-Morrison-Woodbury 公式一步运算得到结果:

$$\begin{aligned} &U_k = [u_1, u_2, \dots, u_k] \in \mathbb{R}^{n\times k} \\ &V_k = [v_1, v_2, \dots, v_k] \in \mathbb{R}^{n\times k} \\ \hline J_k^{-1} &= (J_0 + U_kV_k^{\mathrm{T}})^{-1} \\ &= J_0^{-1} - J_0^{-1}U_k(I_k + V_k^{\mathrm{T}}J_0^{-1}U_k)^{-1}V_k^{\mathrm{T}}J_0^{-1} \end{aligned}$$

这在当 J_0^{-1} 是大型稀疏矩阵时会很有用:
存储 U_k 和 V_k 的代价是 $O(kn)$，远小于存储 J_{k-1}^{-1} (可能是大型稠密矩阵) 的代价 (如果 $k \ll n$ 的话)

> **Sherman-Morrison-Woodbury 公式:**
> 设 $\begin{cases} A \in \mathbb{C}^{n\times n} \\ U, V \in \mathbb{C}^{n\times k} \\ B \in \mathbb{C}^{k\times k} \\ n \ge k \end{cases}$ 且 A, B 均为可逆矩阵，
> 则 $(A + UBV^{\mathrm{H}})$ 可逆的充要条件是 $(B^{-1} + V^{\mathrm{H}}A^{-1}U)$ 可逆，
> 此时逆矩阵 $(A + UBV^{\mathrm{H}})^{-1} = A^{-1} - A^{-1}U(B^{-1} + V^{\mathrm{H}}A^{-1}U)^{-1}V^{\mathrm{H}}A^{-1}$
>
> - 上述公式给出了 A 进行秩 k 更新得到的矩阵 $A + UBV^{\mathrm{H}}$ 的逆矩阵.
> 若 A^{-1} 已知，则我们可以很方便地计算 $(A + UBV^{\mathrm{H}})^{-1}$
> - 特殊地，当 $B = I_k$ 且 $(I_k + V^{\mathrm{H}}A^{-1}U)$ 非奇异时，
> 公式化为 $(A + UV^{\mathrm{H}})^{-1} = A^{-1} - A^{-1}U(I_k + V^{\mathrm{H}}A^{-1}U)^{-1}V^{\mathrm{H}}A^{-1}$
> (这便是实用意义下的 Sherman-Morrison-Woodbury 公式)
> - 进一步，当 $k = 1$ 时，记 $u, v \in \mathbb{C}^n$ (假设 $1 + v^{\mathrm{H}}A^{-1}u \ne 0$)
> 公式化为 $(A + uv^{\mathrm{H}})^{-1} = A^{-1} - \frac{A^{-1}uv^{\mathrm{H}}A^{-1}}{1+v^{\mathrm{H}}A^{-1}u}$
> (这便是最早的 Sherman–Morrison 公式)

(Bad Broyden's Method)
另一种方法是令 J_k^{-1} 是 J_{k-1}^{-1} 的秩 1 修正 (即 $J_k^{-1} = J_{k-1}^{-1} + uv^{\mathrm{T}}$)
代入第二种割线方程 $\Delta x_k = J_k^{-1}\Delta f_k$ 即得:

$$uv^{\mathrm{T}}\Delta f_k = \Delta x_k - J_{k-1}^{-1}\Delta f_k$$

容易看出 u 与向量 $\Delta x_k - J_{k-1}^{-1}\Delta f_k$ 线性相关.

令 $v = \Delta f_k$ (这可保证 v 与 Δx_k 的内积为正值) 即得 $u = \frac{\Delta x_k - J_{k-1}^{-1}\Delta f_k}{\|\Delta f_k\|^2}$

事实上，这样取相当于用 Moore-Penrose 逆求解:

$$\begin{aligned} uv^{\mathrm{T}} &= (\Delta x_k - J_{k-1}^{-1}\Delta f_k)(\Delta f_k)^{\dagger} \\ &= (\Delta x_k - J_{k-1}^{-1}\Delta f_k) \cdot \frac{\Delta f_k^{\mathrm{T}}}{\Delta f_k^{\mathrm{T}}\Delta f_k} \\ &= \frac{(\Delta x_k - J_{k-1}^{-1}\Delta f_k)\Delta f_k^{\mathrm{T}}}{\|\Delta f_k\|^2} \end{aligned}$$

于是我们有:

$$\begin{aligned} J_k^{-1} &= J_{k-1}^{-1} + uv^{\mathrm{T}} \\ &= J_{k-1}^{-1} + \frac{(\Delta x_k - J_{k-1}^{-1}\Delta f_k)\Delta f_k^{\mathrm{T}}}{\|\Delta f_k\|^2} \\ &= \arg\min_{\mathrm{rank}(J - J_{k-1}^{-1}) \le 1} \|J - J_{k-1}^{-1}\|_{\mathrm{F}} \quad (\text{Properties of the Moore-Penrose inverse}) \end{aligned}$$

The End