非线性方程组的牛顿迭代法

姓名: 李徐瑾

学号: 202021110109　　序号: 66

电子科技大学数学科学学院

日期: 2020 年 11 月 7 日

摘　　要

非线性方程组在实际问题中经常出现，并且在科学计算中的地位越来越重要，很多常见的线性模型都是在一定条件下由非线性问题简化得到的，为了得到更符合实际的解答，往往需要直接研究非线性模型，然而从线性到非线性是一个质的飞跃，方程的性质不同，所使用的求解方法也有很大的差别，本文主要先给出由非线性方程到一般非线性方程组牛顿迭代法的分析及推导过程，而后使用牛顿迭代法求解非线性方程组(2.1)，并给出由 MATLAB 编程分析实现的过程.

关键词: 非线性方程组，牛顿迭代法，雅可比矩阵，MATLAB

1　非线性方程组与牛顿迭代法

1.1　非线性方程组

非线性方程组是非线性科学的重要组成部分.

考虑方程组

$$\begin{cases} f_1(x_1, x_2, \cdots, x_n) = 0, \\ f_2(x_1, x_2, \cdots, x_n) = 0, \\ \qquad\vdots \\ f_n(x_1, x_2, \cdots, x_n) = 0. \end{cases} \tag{1.1}$$

其中 $f_1, f_2, \cdots, f_n$ 均是 $x_1, x_2, \cdots, x_n$ 的多元函数.

如果采用向量记法 $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^{\mathrm{T}} \in \mathbb{R}^n$, $\boldsymbol{F} = (f_1, f_2, \cdots, f_n)^{\mathrm{T}}$, 方程组(1.1)可表示为

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}. \tag{1.2}$$

当 $n \geqslant 2$ 且 $f_i(i = 1, 2, \cdots, n)$ 中至少有一个是自变量 $x_i(i = 1, 2, \cdots, n)$ 的非线性函数时，则称方程组(1.1)为**非线性方程组**.

非线性方程组(1.1)的求解无论是在理论还是在实际解法上均比线性方程组和单个方程的求解要复杂和困难，可能无解也可能有一个或者多个解.

求方程组(1.1)的根可直接将单个方程 $(n=1)$ 的求解方法加以推广，实际上只要把单变量函数 $f(x)$ 看成向量函数 $\boldsymbol{F}(\boldsymbol{x})$，将方程组(1.1)改写成方程组(1.2)，就可将之前讨论的求根方法用于求解方程组(1.2)的根，假设向量函数 $\boldsymbol{F}(\boldsymbol{x})$ 是定义在区域 $D \subset \mathbb{R}^n$，若对于 $\boldsymbol{x}_0 \in D$，有

$$\lim_{\boldsymbol{x}\to\boldsymbol{x}_0} \boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{x}_0),$$

则称 $\boldsymbol{F}(\boldsymbol{x})$ 在 $\boldsymbol{x}_0$ 处**连续**，即 $\forall \varepsilon > 0, \exists \delta > 0$，当 $0 < \|\boldsymbol{x} - \boldsymbol{x}_0\| < \delta$ 时，使得

$$\|\boldsymbol{F}(\boldsymbol{x}) - \boldsymbol{F}(\boldsymbol{x}_0)\| < \varepsilon.$$

如果 $\boldsymbol{F}(\boldsymbol{x})$ 在 D 上的每一点均连续，则称 $\boldsymbol{F}(\boldsymbol{x})$ 在域 D 中连续.

向量函数 $\boldsymbol{F}(\boldsymbol{x})$ 的导数 $\boldsymbol{F}'(\boldsymbol{x})$ 称为 $\boldsymbol{F}$ 的**雅克比矩阵**，通常记为 $\mathbf{J}(\boldsymbol{x})$，即

$$\mathbf{J}(\boldsymbol{x}) := \boldsymbol{F}'(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \frac{\partial f_1(x)}{\partial x_2} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \frac{\partial f_2(x)}{\partial x_1} & \frac{\partial f_2(x)}{\partial x_2} & \cdots & \frac{\partial f_2(x)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n(x)}{\partial x_1} & \frac{\partial f_n(x)}{\partial x_2} & \cdots & \frac{\partial f_n(x)}{\partial x_n} \end{bmatrix}. \tag{1.3}$$

1.2 牛顿迭代法

事实上，将单个方程的牛顿迭代法直接用于方程组(1.2)，则可得到解非线性方程组的牛顿迭代法，因此我们先分析非线性方程的数值解法，然后再将其延伸到方程组的解法.

1.2.1 牛顿迭代法的思想

牛顿迭代法实质上是一种线性化方法，基本思想是将非线性方程 $f(x)=0$ 逐步归结为某种线性方程来求解.

设已知方程 $f(x)=0$ 有近似根 $x_k (f'(x_k) \neq 0)$，将函数 $f(x)$ 在点 x_k 展开，则有

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

于是方程 $f(x)=0$ 可近似地表示为

$$f(x_k) - f'(x_k)(x - x_k) = 0.$$

这是一个线性方程，记其根为 x_{k+1}，则 x_{k+1} 的计算公式为

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, \cdots. \tag{1.4}$$

这便是**牛顿迭代法**，亦称为**切线法**.

关于牛顿迭代法的收敛性，根据式(1.4)知迭代公式为

$$\varphi(x) = x - \frac{f(x)}{f'(x)},$$

因此

$$\varphi'(x) = \frac{f(x)f''(x)}{[f'(x)]^2}.$$

假设 $x^\star$ 是 $f(x)$ 的单根，即 $f(x^\star) = 0, f'(x^\star) \neq 0$，故牛顿迭代法是平方收敛的. 即

$$\lim_{k\to\infty} \frac{x_{k+1} - x^\star}{(x_k - x^\star)^2} = \frac{f''(x^\star)}{2f'(x^\star)}.$$

牛顿迭代法有明显的几何意义，方程 $f(x) = 0$ 的根 $x^\star$ 可解释为曲线 $y = f(x)$ 与 x 轴交点的横坐标，即

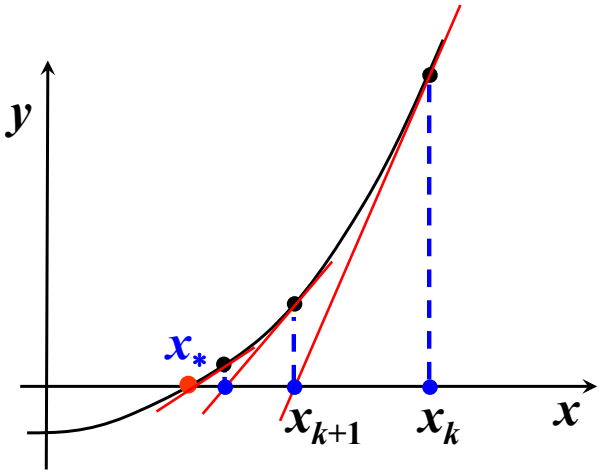


图 1.1: 牛顿迭代法的几何意义

定理 1.1. *若 $f(x)$ 在 $x^\star$ 的某邻域内具有连续的二阶导数，且满足 $f(x^\star) = 0, f'(x^\star) \neq 0$，则对充分靠近 $x^\star$ 的初始值 x_0，牛顿迭代法生成的序列 $\{x_n\}$ 至少平方收敛于 $x^\star$.*

定理 1.2 (非局部收敛定理)**.** *给定方程 $f(x) = 0$，如果函数 $f(x) \in C^2[a,b]$ 且满足:*

(1) $f(a)f(b) < 0$,

(2) $f'(x), f''(x)$ *在* $[a,b]$ *上连续且不变号，*

(3) $\exists x_0 \in [a,b], \text{s.t. } f(x_0)f''(x_0) > 0.$

则方程 $f(x) = 0$ 在 $[a,b]$ 上有唯一的根 $x^\star$.

现在我们可将牛顿迭代法的思想归结为下列的步骤:

步一 任取迭代初始值 x_0;

步二 计算 $x_1 = x_0 - f(x_0)/f'(x_0)$;

步三 判断收敛性，若 $|x_1 - x_0| < \varepsilon$ 或 $|f(x_0)| < \varepsilon$，则算法收敛，停止计算，输出近似解 x_1;

步四 令 $x_0 \leftarrow x_1$，返回**步二**.

1.2.2 牛顿迭代法的改进

我们已经知道牛顿迭代法是平方收敛的，因而其优点是收敛快. 但缺点也较为明显，主要表现为以下两点:

- 每一步迭代均需要计算 $f(x_k), f'(x_k)$ 且有时 $f'(x_k)$ 计算量偏大，
- 初始值 x_0 仅在根 $x^\star$ 附近才能保证收敛，选取不恰当的初始值可能导致不收敛.

为了克服上述两个缺点，我们通常可以采用**简化牛顿法**和**牛顿下山法**.

(1) **简化牛顿法**的迭代公式为

$$x_{k-1} = x_k - Cf(x_k), \quad C \neq 0, k = 0, 1, \cdots . \tag{1.5}$$

迭代函数 $\varphi(x) = x - Cf(x)$.

若 $|\varphi'(x)| = |1 - Cf'(x)| < 1$，即当 $0 < Cf'(x) < 2$ 时，在根 $x^\star$ 附近成立，此时迭代公式(1.5)**局部收敛**. 若取 $C = 1/f'(x_0)$，则称为**简化牛顿法**，此方法的几何意义是用斜率为 $f'(x_0)$ 的平行弦与 x 轴的交点作为 $x^\star$ 的近似，因此亦称为**平行弦法**，此种方法计算量相对较小，但只是线性收敛.

(2) **牛顿下山法**的收敛性依赖于初始值 x_0 的选取，如果 x_0 偏离根 $x^\star$ 较远，则可能导致发散. **下山法**意即对迭代过程添加附加条件

$$|f(x_{k+1})| < |f(x_k)|.$$

将下山法与牛顿法结合，以此在保证函数值稳定下降的前提下加快牛顿法的收敛速度. 牛顿下山法的迭代公式为

$$x_{k+1} = x_k - \lambda \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, \cdots .$$

其中 $\lambda (0 < \lambda \leqslant 1)$ 称为**下山因子**.

接下来我们考虑重根的情形，即当方程 $f(x) = 0$ 有 m 重根时，有

$$f(x^\star) = f'(x^\star) = \cdots = f^{(k-1)}(x^\star) = 0, \quad f^{(m)}(x^\star) \neq 0.$$

只有 $f'(x_k) \neq 0$ 仍可使用牛顿迭代法公式(1.4)计算. 此时迭代函数的导数满足

$$\varphi'(x^\star) = 1 - \frac{1}{m}, \quad |\varphi'(x^\star)| < 1.$$

故牛顿迭代法求重根只是线性收敛的. 不妨取

$$\varphi(x) = x - m\frac{f(x)}{f'(x)},$$

则 $\varphi'(x^\star) = 0$. 采用迭代公式

$$x_{k+1} = x_k - m\frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, \cdots .$$

这是二阶收敛的，但我们需要求出 $x^\star$ 的重数 m.

不妨令 $\mu(x) = f(x)/f'(x)$，如果 $x^\star$ 是 $f(x) = 0$ 的 m 重根，则

$$\mu(x) = \frac{(x - x^\star)g(x)}{mg(x) \mid (x - x^\star)g'(x)},$$

故 $x^\star$ 是 $\mu(x) = 0$ 的单根，因而可构造迭代函数

$$x_{k+1} = x_k - \frac{f(x_k)f'(x_k)}{[f'(x_k)]^2 - f(x_k)f''(x_k)}, \quad k = 0, 1, \cdots .$$

1.2.3 二元非线性方程组牛顿迭代法的推导

现在我们考虑二元非线性方程组

$$\begin{cases} f(x,y)=0, \\ g(x,y)=0. \end{cases} \tag{1.6}$$

的迭代的方程，我们先给出**一元函数的** Taylor **公式**和**二元函数的** Taylor **公式**.

定理 1.3 (一元函数的 Taylor 公式)**.** *若函数 $f(x)$ 在含有 x_0 的某开区间 (a,b) 内具有直至 $(n+1)$ 阶的导数，则对任一 $x\in(a,b)$，有*

$$f(x)=f(x_0)+f'(x_0)(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\cdots+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n+R_n(x).$$

其中 $R_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1},\xi\in(x_0,x)$.

定理 1.4 (二元函数的 Taylor 公式)**.** *设 $z=f(x,y)$ 在点 (x_0,y_0) 的某邻域内连续且有直至 $(n+1)$ 阶的连续偏导数，(x_0+h,y_0+h) 为此邻域内任意一点，则有*

$$\begin{aligned} f(x_0+h,y_0+h)=&f(x_0,y_0)+\Big(h\frac{\partial}{\partial x}+k\frac{\partial}{\partial y}\Big)f(x_0,y_0)\\ &+\frac{1}{2!}\Big(h\frac{\partial}{\partial x}+k\frac{\partial}{\partial y}\Big)^2 f(x_0,y_0)+\cdots+\frac{1}{n!}\Big(h\frac{\partial}{\partial x}+k\frac{\partial}{\partial y}\Big)^n f(x_0,y_0)\\ &+\frac{1}{(n+1)!}\Big(h\frac{\partial}{\partial x}+k\frac{\partial}{\partial y}\Big)^{n+1} f(x_0+\theta h,y_0+\theta k),\quad \theta\in(0,1). \end{aligned}$$

其中

$$\Big(h\frac{\partial}{\partial x}+k\frac{\partial}{\partial y}\Big)^m f(x_0,y_0)=\sum_{p=0}^{m}\binom{m}{p}h^p k^{m-p}\frac{\partial^m f}{\partial x^p\partial y^{m-p}}\bigg|_{(x_0,y_0)}.$$

接下来我们来推导二元函数(1.6)的牛顿迭代法，假设 $z=f(x,y)$ 在点 (x_0,y_0) 的某邻域内连续且有直至 2 阶的连续偏导数，(x_0+h,y_0+k) 为此邻域内任意一点，则有

$$\begin{aligned} f(x_0+h,y_0+k)\approx& f(x_0,y_0)\\ &+\Big(h\frac{\partial}{\partial x}f(x,y)\Big|_{x=x_0}+k\frac{\partial}{\partial y}f(x,y)\Big|_{y=y_0}\Big),\quad h=x-x_0,k=y-y_0. \end{aligned}$$

于是方程 $f(x,y)=0$ 可近似的表示为

$$f(x_k,y_k)+\Big(h\frac{\partial}{\partial x}f(x,y)\Big|_{x=x_k}+k\frac{\partial}{\partial y}f(x,y)\Big|_{y=y_k}\Big)=0,$$

即 $f(x_k,y_k)+(x-x_k)f_k(x_k,y_k)+(y-y_k)f_y(x_k,y_k)=0$.

同理设 $z=g(x,y)$ 在点 (x_0,y_0) 的某邻域内连续且有直至 2 阶的连续偏导数，(x_0+h,y_0+k) 为此邻域内任意一点，则有

$$\begin{aligned} g(x_0+h,y_0+k)\approx& g(x_0,y_0)\\ &+\Big(h\frac{\partial}{\partial x}g(x,y)\Big|_{x=x_0}+k\frac{\partial}{\partial y}g(x,y)\Big|_{y=y_0}\Big),\quad h=x-x_0,k=y-y_0. \end{aligned}$$

于是方程 $g(x,y)=0$ 可近似的表示为

$$g(x_k,y_k)+\Big(h\frac{\partial}{\partial x}g(x,y)\Big|_{x=x_k}+k\frac{\partial}{\partial y}g(x,y)\Big|_{y=y_k}\Big)=0,$$

即 $g(x_k,y_k)+(x-x_k)g_k(x_k,y_k)+(y-y_k)g_y(x_k,y_k)=0.$

于是得到方程组

$$\begin{cases} f(x_k,y_k)+(x-x_k)f_k(x_k,y_k)+(y-y_k)f_y(x_k,y_k)=0, \\ g(x_k,y_k)+(x-x_k)g_k(x_k,y_k)+(y-y_k)g_y(x_k,y_k)=0. \end{cases} \tag{1.7}$$

接下来我们求解方程组(1.7):

当 $g_x(x_k,y_k)f_y(x_k,y_k)-f_x(x_k,y_k)g_y(x_k,y_k)\neq 0$ 时，则有

$$\begin{aligned} x &= x_k \\ &\quad + \frac{f(x_k,y_k)g_y(x_k,y_k)-g(x_k,y_k)f_y(x_k,y_k)}{g_x(x_k,y_k)f_y(x_k,y_k)-f_x(x_k,y_k)g_y(x_k,y_k)}, \\ y &= y_k \\ &\quad + \frac{g(x_k,y_k)f_x(x_k,y_k)-f(x_k,y_k)g_x(x_k,y_k)}{g_x(x_k,y_k)f_y(x_k,y_k)-f_x(x_k,y_k)g_y(x_k,y_k)}. \end{aligned}$$

不妨记

$$\begin{aligned} gf_x-fg_x|_{(x_k,y_k)} &= g(x_k,y_k)f_x(x_k,y_k)-f(x_k,y_k)g_x(x_k,y_k), \\ fg_y-gf_y|_{(x_k,y_k)} &= f(x_k,y_k)g_y(x_k,y_k)-g(x_k,y_k)f_y(x_k,y_k), \\ g_xf_y-f_xg_y|_{(x_k,y_k)} &= g_x(x_k,y_k)f_y(x_k,y_k)-f_x(x_k,y_k)g_y(x_k,y_k). \end{aligned}$$

因此

$$\begin{cases} x = x_k+\frac{fg_y-gf_y}{g_xf_y-f_xg_y}\Big|_{(x_k,y_k)}, \\ y = y_k+\frac{gf_x-fg_x}{g_xf_y-f_xg_y}\Big|_{(x_k,y_k)}. \end{cases}$$

迭代公式为

$$\begin{cases} x_{k+1} = x_k+\frac{fg_y-gf_y}{g_xf_y-f_xg_y}\Big|_{(x_k,y_k)}, \\ y_{k+1} = y_k+\frac{gf_x-fg_x}{g_xf_y-f_xg_y}\Big|_{(x_k,y_k)}. \end{cases} \tag{1.8}$$

通过迭代公式(1.8)可迭代出当 $k=1,2,\cdots$ 时 (x_k,y_k) 的值，当 $|(x_{k+1},y_{k+1})|<\delta$ 时，方程组(1.6)的根即为 (x_k,y_k)，这就是**二元函数的牛顿迭代法**.

1.2.4 非线性方程组的牛顿迭代法

现在我们将非线性方程的牛顿迭代法推广到非线性方程组(1.2)，则可得到非线性方程组(1.2)的牛顿迭代法

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}-\mathbf{J}\left(\boldsymbol{x}^{(k)}\right)^{-1}\boldsymbol{F}\left(\boldsymbol{x}^{(k)}\right),\quad k=0,1,\cdots.$$

其中 $\mathbf{J}\left(\boldsymbol{x}^{(k)}\right)^{-1}$ 是雅克比矩阵(1.3)的逆矩阵，不妨记 $\boldsymbol{x}^{(k-1)}-\boldsymbol{x}^{(k)}=\Delta\boldsymbol{x}^{(k)}$，则有

$$\mathbf{J}\left(\boldsymbol{x}^{(k)}\right)\Delta\boldsymbol{x}^{(k)}=-\boldsymbol{F}\left(\boldsymbol{x}^{(k)}\right).$$

这是一个线性方程组，可用**高斯消去法**求解. 求出向量 $\Delta\boldsymbol{x}^{(k)}$，再令 $\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+\Delta\boldsymbol{x}^{(k)}$，每步需要计算向量函数 $\boldsymbol{F}\left(\boldsymbol{x}^{(k)}\right)$ 和雅可比矩阵 $\mathbf{J}\left(\boldsymbol{x}^{(k)}\right)$. 通常可用 $\left\|\boldsymbol{x}^{(k)}\right\|<\varepsilon$ 或

$\left\|\boldsymbol{F}\left(\boldsymbol{x}^{(k)}\right)\right\| < \delta$ 作为牛顿迭代法终止迭代的条件，其中 ε, δ 为预先给定的精度要求，有时也可用预先确定的最大迭代次数 M 作为终止迭代的条件.

定理 1.5 (收敛性定理)**.** 设 $\boldsymbol{F}(\boldsymbol{x})$ 的定义域为 $D \subset \mathbb{R}^n$，且 $\boldsymbol{F}(\boldsymbol{x}^\star) = \boldsymbol{0}, \boldsymbol{x}^\star \in D$. 若在 $\boldsymbol{x}^\star$ 的开邻域 $S_0 \subset D$ 上 $\mathbf{J}(\boldsymbol{x})$ 存在且连续，$\det \mathbf{J}(\boldsymbol{x}^\star) \neq 0$，则牛顿迭代法生成的序列 $\left\{\boldsymbol{x}^{(k)}\right\}$ 在闭域 $S \subset S_0$ 上**超线性收敛**于 $\boldsymbol{x}^\star$，若存在常数 $L > 0$，使得

$$\|\mathbf{J}(\boldsymbol{x}) - \mathbf{J}(\boldsymbol{x}^\star)\| \leqslant L\|\boldsymbol{x} - \boldsymbol{x}^\star\|, \quad \forall \boldsymbol{x} \in S.$$

现在我们将非线性方程组(1.2)的牛顿迭代法归结为下列的步骤:

步一 给定初始向量 $\boldsymbol{x}^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)}\right)$;

步二 计算 $\mathbf{J}\left(\boldsymbol{x}^{(0)}\right), \boldsymbol{F}\left(\boldsymbol{x}^{(0)}\right)$;

步三 计算向量 $\boldsymbol{y}^{(0)} = \left(y_1^{(0)}, y_2^{(0)}, \cdots, y_n^{(0)}\right)^{\mathrm{T}}$，满足 $\mathbf{J}\left(\boldsymbol{x}^{(0)}\right)\boldsymbol{y}^{(0)} = -\boldsymbol{F}\left(\boldsymbol{x}^{(0)}\right)$，且

$$\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} - \mathbf{J}\left(\boldsymbol{x}^{(k-1)}\right)^{-1}\boldsymbol{F}\left(\boldsymbol{x}^{(k-1)}\right) = \boldsymbol{x}^{(k-1)} - \boldsymbol{y}^{(k-1)}.$$

步四 一旦确定了 $\boldsymbol{y}^{(0)}$，我们便可计算 $\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} + \boldsymbol{y}^{(0)}$;

步五 重复上述过程，直至 $\boldsymbol{x}^{(k)}$ 收敛于 $\boldsymbol{x}^\star$.

下面证明非线性方程组的牛顿迭代法的收敛性:

证明. 假设初始值 $\boldsymbol{x}^{(0)}$ 是足够接近方程(1.2)精确解 $\boldsymbol{x}^\star$，且 $\det \mathbf{J}(\boldsymbol{x})^{-1} \neq 0$，目的是说明

$$\frac{\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^\star\|}{\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^\star\|^2} = \lambda,$$

其中常数 $\lambda > 0$. 已知

$$\begin{aligned}\left\|\boldsymbol{e}^{(k+1)}\right\| &= \left\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^\star\right\| \\ &= \left\|\boldsymbol{x}^\star - \mathbf{J}\left(\boldsymbol{x}^{(k)}\right)^{-1}\boldsymbol{F}\left(\boldsymbol{x}^{(k)}\right) - \boldsymbol{x}^\star\right\|,\end{aligned}$$

不妨令 $\left\|\boldsymbol{e}^{(k)}\right\| = \left\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^\star\right\|$，则有

$$\left\|\boldsymbol{e}^{(k+1)}\right\| = \left\|\boldsymbol{e}^{(k)} - \mathbf{J}\left(\boldsymbol{x}^{(k)}\right)^{-1}\boldsymbol{F}\left(\boldsymbol{x}^{(k)}\right)\right\|. \tag{1.9}$$

根据定理1.4，则有

$$\boldsymbol{F}\left(\boldsymbol{x}^{(k)}\right) \approx \boldsymbol{F}\left(\boldsymbol{x}^\star\right) + \mathbf{J}\left(\boldsymbol{x}^{(k)}\right)\boldsymbol{e}^{(k)} + \frac{1}{2}\left(\boldsymbol{e}^{(k)}\right)^{\mathrm{T}}\mathbf{H}\left(\boldsymbol{e}^{(k)}\right),$$

其中 $\mathbf{H}$ 为黑塞矩阵，即 $\mathbf{H} = \left(\partial^2 f / \partial x_i \partial x_j\right)_{ij}$. 因此

$$\begin{aligned}\mathbf{J}^{-1}\left(\boldsymbol{F}\left(\boldsymbol{x}^{(k)}\right)\right) &\approx \mathbf{J}^{-1}\left[\boldsymbol{F}\left(\boldsymbol{x}^\star\right) + \mathbf{J}\left(\boldsymbol{x}^{(k)}\right)\boldsymbol{e}^{(k)} + \frac{1}{2}\left(\boldsymbol{e}^{(k)}\right)^{\mathrm{T}}\mathbf{H}\left(\boldsymbol{e}^{(k)}\right)\right] \\ &= \boldsymbol{e}^{(k)} + \frac{\mathbf{J}^{-1}}{2}\left(\boldsymbol{e}^{(k)}\right)^{\mathrm{T}}\mathbf{H}\left(\boldsymbol{e}^{(k)}\right).\end{aligned} \tag{1.10}$$

根据式(1.9)和式(1.10)，则有

$$\begin{aligned}\left\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^\star\right\| &= \left\|\boldsymbol{e}^{(k+1)}\right\| \\ &= \left\|\frac{\mathbf{J}^{-1}}{2}\left(\boldsymbol{e}^{(k)}\right)^{\mathrm{T}}\mathbf{H}\left(\boldsymbol{e}^{(k)}\right)\right\| \leqslant \frac{\|\mathbf{J}^{-1}\|\|\mathbf{H}\|}{2}\left\|\boldsymbol{e}^{(k)}\right\|^2.\end{aligned}$$

□

2 方法应用

采用牛顿迭代法求解给定的非线性方程组(2.1).

$$\begin{cases} (x_1+3)(x_2^3-7)+18=0, \\ \sin(x_2 e^{x_1}-1)=0. \end{cases} \tag{2.1}$$

显然方程组(2.1)有整数解 $(0,1)$. 在 MATLAB R2020a 中采用`fimplicit`函数绘制的方程组(2.1)图像如图2.1所示:

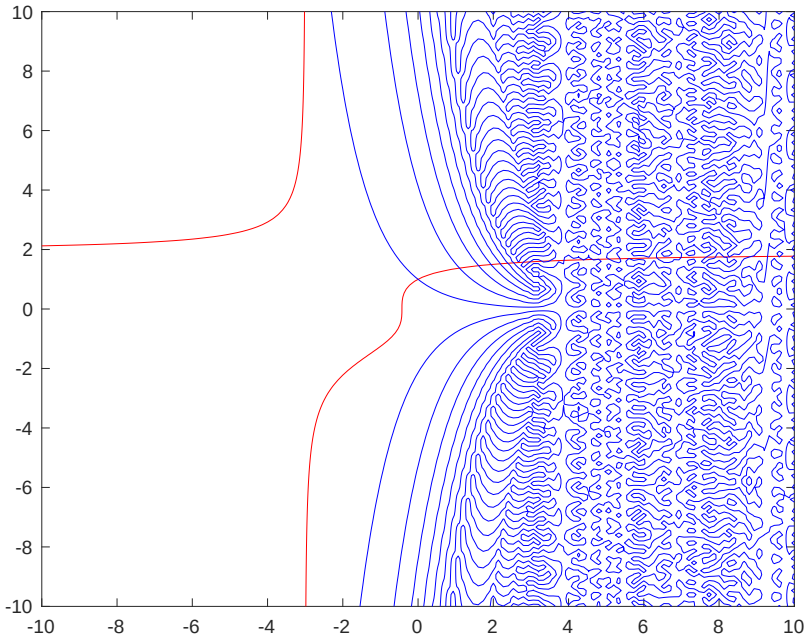


图 2.1: MATLAB 绘制方程组(2.1)图像的示意图

图2.1的绘制采用如下的 MATLAB 代码:

```
f=@(x,y) (x+3)*(y^3-7)+18;
fimplicit(f,[-10 25 -10 25],'r')

hold on

g=@(x,y) sin(y*exp(x)-1);
fimplicit(g,[-10 25 -10 25],'b')

hold off
```

在 Python 3.7.7 中利用 `matplotlib` 和 `numpy` 绘制的方程组(2.1)图像如图2.2所示:

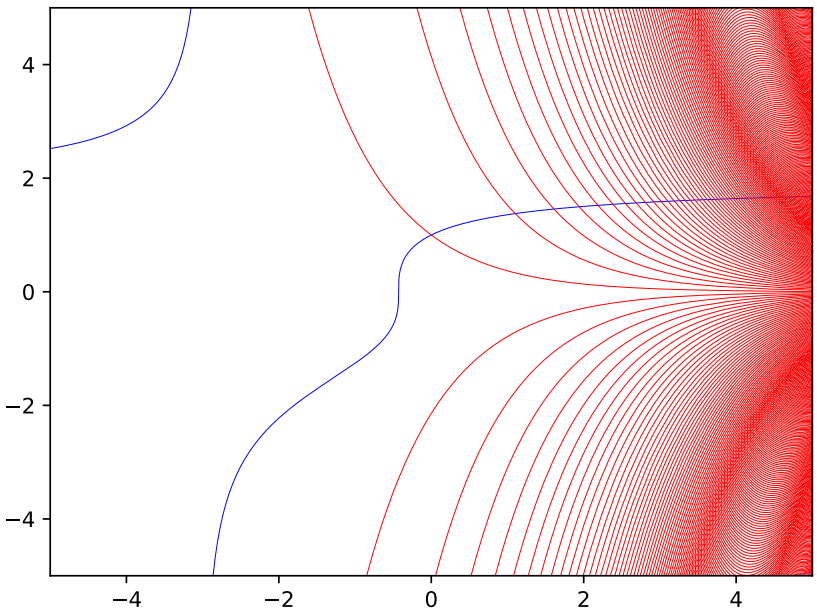


图 2.2: Python 绘制方程组(2.1)图像的示意图

图2.2的绘制采用如下的 Python 代码:

```
import matplotlib.pyplot as plt
import numpy as np

def f(x,y):
    return (x+3)*(y**3-7)+18
def g(x,y):
    return np.sin(y*np.exp(x)-1)

n=500
x=np.linspace(-5,5,500)
y=np.linspace(-5,5,500)

X,Y=np.meshgrid(x,y)

plt.contour(X,Y,f(X,Y),1,colors='b',linewidths=0.4)
plt.contour(X,Y,g(X,Y),1,colors='r',linewidths=0.4)

plt.show()
```

根据图2.1和图2.2的结果，选取交点附近的坐标值作为初始值进行迭代. 采用如下的 MATLAB 代码对方程组(2.1)交点附近的值进行牛顿迭代.

子程序一 `F.m` 用于求每一步的 $\boldsymbol{F}(\boldsymbol{x})$.

```
function[out]=F(x,y)
syms x1 x2;
f1=(x1+3)*(x2^3-7)+18;
f2=sin(x2*exp(x1)-1);
Y=[f1;f2];
x1=x;
x2=y;
out=subs(Y);
end
```

子程序二 `JF.m` 用于求每一步的雅克比矩阵.

```
function[y]=JF(x,y)
syms x1 x2
f1=(x1+3)*(x2^3-7)+18;
f2=sin(x2*exp(x1)-1);
df1x=diff(sym(f1),'x1');
df1y=diff(sym(f1),'x2');
df2x=diff(sym(f2),'x1');
df2y=diff(sym(f2),'x2');
j=[df1x,df1y;df2x,df2y];
x1=x;
x2=y;
y=double(subs(j));
end
```

主程序 `Newton.m` 是对方程组(2.1)进行牛顿迭代.

```
function[Z,P,k,E,S]=Newton(P,e0)
Z=F(P(1),P(2));
J=JF(P(1),P(2));
Q=P-J\Z;
e=norm((Q-P),inf);
P=Q;
Z=F(P(1),P(2));
S=[Q];
E=[e];
k=1;
while e>=e0
   J=JF(P(1),P(2));
   Q=P-J\Z;
   e=norm((Q-P),inf);
   P=Q;
   Z=F(P(1),P(2));
```

```
    S=[S P];
    E=[E e];
    k=k+1;
end
Z=vpa(Z)
P=vpa(P);
S=vpa(S);
E=vpa(E);
end
```

设置精度为 $e = 10^{-5}$，在 MATLAB R2020a 分别取如表2.1所示的初始值运行程序 `F.m` 、 `JF.m` 和 `Newton.m` . 并得到接下来的六组迭代结果:

表 2.1: 6 次迭代选取的初始值

迭代结果	表2.2	表2.3	表2.4	表2.5	表2.6	表2.7
迭代初始值	-0.040	1.075	1.604	1.935	2.177	2.688
(x_1^0, x_2^0)	0.96	1.34	1.42	1.46	1.49	1.54

表 2.2: 初始值为 $(-0.040, 0.96)$ 的迭代结果

迭代次数	x_1	x_2	e
1	0.00018764791983762 444721611681478909	1.002393408614774 1826527201579592	0.0423934086147741 82652720157959161
2	-0.00000324227614454 1249896480214001084	1.00000371102556 48227175119899779	0.0023896975892093 599352081679812331
3	-0.000000000000992216042 57619582046569943993453	1.000000000003146 2105318997286378	0.000003711022418612 1856122613401583001

表 2.3: 初始值为 $(1.075, 1.34)$ 的迭代结果

迭代次数	x_1	x_2	e
1	1.102024869036446 6056395111699049	1.378461462387132 9155940821227552	0.0384614623871329 15594082122755166
2	1.101168345107749 7215327639922375	1.377008371470982 9065809546580189	0.0014530909161500 090131274647362467
3	1.101168436301883 0651330315890114	1.377006550054511 0421307263105578	0.000001821416471864 4502283474611172107

表 2.4: 初始值为 (1.604, 1.42) 的迭代结果

迭代次数	x_1	x_2	e
1	1.608964997342521 8249898739813013	1.458274363802354 2205596517677181	0.0382743638023542 20559651767718068
2	1.609013900511893 8845434633231169	1.457254768660919 1167963985250797	0.0010195951414351 037632532426383762
3	1.609014377222653 6136092463460571	1.457254129187330 5584033693427254	0.000006394735885 58 39302918235430635174

表 2.5: 初始值为 (1.935, 1.46) 的迭代结果

迭代次数	x_1	x_2	e
1	1.941043297812879 3617050840017134	1.498344996495315 1981785313276591	0.0383449964953151 98178531327659099
2	1.940535487293205 9732938724667489	1.497280415079132 5703198137357562	0.0010645814161826 278587175919028949
3	1.940535631232127 7062195660593209	1.497279563996578 3454833642453727	0.000000851082554224 83644949038346122073

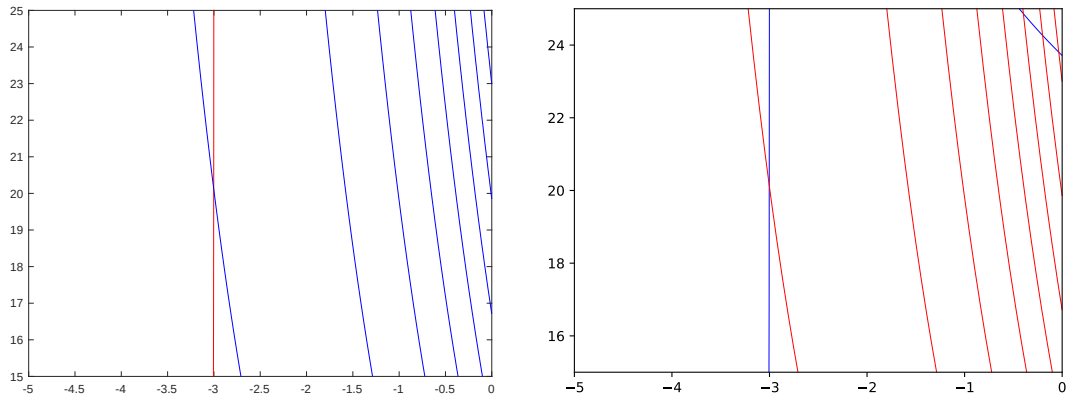
表 2.6: 初始值为 (2.177, 1.49) 的迭代结果

迭代次数	x_1	x_2	e
1	2.188893030333566 2304580607740597	1.523574396024620 5541318038807763	0.0335743960246205 54131803880776286
2	2.187171863987524 6189111099797759	1.522606834111617 4938342254789248	0.0017211663460416 115469507942837653
3	2.18717077695660 865007833770624	1.522605792985845 9530825293565194	0.000001087030915968 8327722735358985321

表 2.7: 初始值为 (2.688, 1.54) 的迭代结果

迭代次数	x_1	x_2	e
1	2.687419770586015 9720618844782872	1.565698197765964 7484947454132387	0.0256981977659647 48494745413238669
2	2.687059806168365 8036809129748115	1.565256347405070 1238022869460335	0.00044185036089462 469245846720520842
3	2.687059785059851 0707426894123864	1.565256193072133 7807870572778944	0.0000001543329363 43 01522966813907154719

观察图2.1和图2.2，我们可以发现在 y 轴正方向可能会有交点，因此我们选取区域 $[-5, 0] \times [15, 25]$ 分别采用 MATLAB 和 Python 作出方程组(2.1)在此区域的图像如图2.3所示.



(a) MATLAB 绘制的图像　　(b) Python 绘制的图像

图 2.3: 方程组(2.1)在点 $(-3.006, 20.23)$ 附近的交点示意图

根据图2.3所示，我们可以看出在区域 $[-5, 0] \times [15, 25]$ 方程组(2.1)的确有交点，选取 $(-3.006, 20.23)$ 作为初始值进行迭代，有如下的迭代结果:

表 2.8: 初始值为 $(-3.006, 20.23)$ 的迭代结果

迭代次数	x_1	x_2	e
1	-3.00226427175802 79666497409029065	20.13083861550757 5239629863760843	0.0991613844924247 60370136239157416
2	-3.00220860943217 44145494655474207	20.12994703530901 7541420661938202	0.00089158019855769 820920182264087115
3	-3.00220860205891 79266550165537932	20.12994690532731 5207307796741835	0.0000001299817023 34 11286519636643249399

3 牛顿迭代法的优缺点

牛顿迭代法是梯度下降法的进一步发展，梯度下降法以目标函数的一阶偏导数作为信息、以负梯度方向作为搜索方向，只考虑目标函数在迭代点的局部性质；而牛顿迭代法不仅使用目标函数的一阶偏导数，而且进一步利用目标函数的二阶偏导数，考虑了梯度变化的趋势，因而能更全面地确定合适的搜索方向加快收敛.

但牛顿迭代法存在以下两个缺点:

1. 对目标函数有较严格的要求. 函数必须具有连续的一、二阶偏导数，黑塞矩阵必须正定.
2. 计算复杂，除需计算梯度外，仍需计算雅可比矩阵及其逆矩阵. 计算量、存储量均很大，且均以维数 n 的平方增加，当 n 很大时问题更加突出.

为了克服上述问题，人们提出了**拟牛顿法**. 此方法的基本思想是: 不用二阶偏导数而构造出近似的正定对称的黑塞矩阵或黑塞矩阵的逆矩阵，在拟牛顿的条件下优化目标函数. 不同的构造方法会产生不同的拟牛顿法. 即对算法中用来计算搜索方向的黑塞矩阵或黑塞矩阵的逆矩阵作近似计算.

参考文献

[1] KELLEY C T. Solving Nonlinear Equations with Newton's Method[M]. Philadelphia: Society for Industrial Applied Mathematics, 2003: 39-41.

[2] REMANI C. Numerical methods for solving systems of nonlinear equations[J]. Lakehead University Thunder Bay, Ontario, Canada, 2013: 6-10.

[3] 李庆扬, 王能超, 易大义. 数值分析[M]. 第五版. 北京: 清华大学出版社, 2008: 222-226, 234-236.

[4] 钟尔杰, 黄廷祝. 数值分析[M]. 北京: 高等教育出版社, 2004: 30-35.

[5] 谢世坤, 段芳, 李强征, 等. 非线性方程组求解的三种 Newton 法比较[J]. 井冈山学院学报 (自然科学), 2006, 27(08): 8-11.

[6] 徐瑞民. 二元非线性方程组求根的牛顿迭代法[J]. 山东轻工业学院学报, 2009, 23(04): 89-91.