数学实验：第四次作业

计算机系　计 73　2017011620　李家昊

2020 年 3 月 26 日

1　实验目的

- 掌握用 MATLAB 软件求解非线性方程和方程组的基本方法，并对结果作初步分析。
- 练习用非线性方程和方程组建立实际问题的模型并进行求解。

2　问题求解

2.1　Chap6-Ex3 利率（应用题）

2.1.1　问题分析

题目给出了按揭贷款的本金总额，还款期数，以及每期还款金额，需要求出贷款利率，这是一个非线性方程求解问题。

2.1.2　模型假设

考虑到实际情况，该模型基于以下假设，

1. 借款人能够按时按量还款。
2. 偿还过程中本金利率不变。

2.1.3　模型建立

数学模型　设贷款总额为 Q，每期贷款利率为 $x > 0$，第 n 期还款后剩余本金为 a_n，每期还款金额为 q，共分 N 期还清，则序列 $\{a_n\}$ 满足差分方程，

$$a_{n+1} = (1+x)a_n - q, \quad n = 0, 1, 2, \cdots, N-1 \tag{1}$$

可解得其通项为，

$$a_n = (1+x)^n \left(a_0 - \frac{q}{x}\right) + \frac{q}{x}, \quad n = 0, 1, 2, \cdots, N \tag{2}$$

由条件，有 $a_0 = Q$，$a_N = 0$，带入 $n = N$ 到方程 (2)，得到，

$$0 = (1+x)^N \left(Q - \frac{q}{x}\right) + \frac{q}{x} \tag{3}$$

进一步化简得，

$$N \ln(1+x) + \ln\left(1 - \frac{Q}{q}x\right) = 0, \quad x \in \left(0, \frac{q}{Q}\right) \tag{4}$$

记方程 (4) 左端为 $f(x)$，则有 $f(x) = 0$，即为本题的模型。

零点分析 那么，$f(x)$ 有没有零点呢？如果有，则有多少个零点？为了进一步研究 $f(x)$ 的零点，这里首先求出它的导函数，

$$f'(x) = \frac{(Nq - Q) - (N+1)Qx}{(1+x)(q - Qx)} \tag{5}$$

令 $f'(x) = 0$，解得，

$$x_1 = \frac{Nq - Q}{(N+1)Q} \tag{6}$$

由于利率的存在，总还款金额必定大于贷款总额，即 $Nq > Q$，因此 $0 < x_1 < q/Q$，且由于 $0 < x < q/Q$，因此 $q - Qx > 0$。

由此可得，当 $x \in (0, x_1)$ 时，$f'(x) > 0$，$f(x)$ 递增；当 $x \in (x_1, q/Q)$ 时，$f'(x) < 0$，$f(x)$ 递减。

由于 $f(0) = 0$，因此 $f(x_1) > 0$，由于 $\lim_{x \to q/Q^-} f(x) = -\infty$，且 $f(x)$ 连续，由零点定理知，$f(x)$ 在区间 $(x_1, q/Q)$ 中必有一个零点，由上文分析的 $f(x)$ 的单调性可知，该零点为 $f(x)$ 的唯一零点。

2.1.4 算法设计

首先应当画出 $f(x)$ 在 $x \in (0, q/Q)$ 的图像，确定函数零点的大致位置，将其作为初值。由 $f(x)$ 的单调性可知零点两侧异号，因此可利用 `fzero` 命令求解。

2.1.5 Matlab 程序

请参见附录4.1。

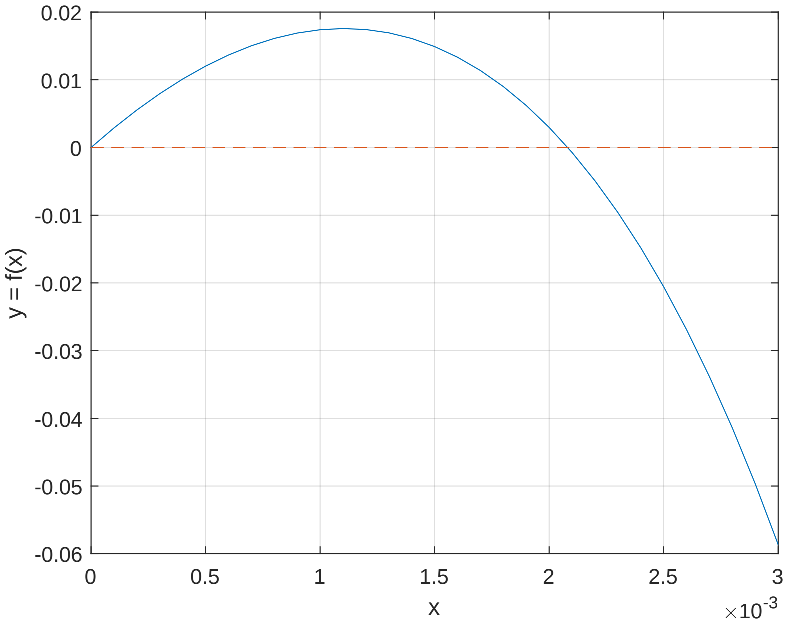


图 1: $f(x)$ 在零点附近的图像

2.1.6 计算结果

第一小问 由题意，此处贷款月利率为 x，应还款总额 $Q = 15$ 万元，每月还款 $q = 0.1$ 万元，分 $N = 180$ 个月还清，带入方程 (4) 中，得到，

$$f(x) = 180\ln(1+x) + \ln(1-150x) \tag{7}$$

画出 $f(x)$ 的图像，如图 1所示，可见 $f(x)$ 的唯一零点在 0.002 附近，将其作为初值，用 `fzero` 求解得到结果为 $\hat{x} = 0.2081\%$，其对应的绝对误差为 $e = f(\hat{x}) = -3.1086 \times 10^{-15}$。

第二小问 按月还款时，贷款月利率为 x，应还款总额 $Q = 50$ 万元，每月还款 $q = 0.45$ 万元，分 $N = 180$ 个月还清，带入方程 (4) 中，得到，

$$f(x) = 180\ln(1+x) + \ln\left(1 - \frac{1000}{9}x\right) \tag{8}$$

画出 $f(x)$ 的图像，如图 2所示，可见 $f(x)$ 的唯一零点在 0.006 附近，将其作为初值，用 `fzero` 求解得到结果为 $\hat{x} = 0.5851\%$，其对应的绝对误差为 $e = f(\hat{x}) = -1.7542 \times 10^{-14}$。

按年还款时，贷款年利率为 x，应还款总额 $Q = 50$ 万元，每年还款 $q = 4.5$ 万元，分 $N = 20$ 年还清，带入方程 (4) 中，得到，

$$f(x) = 20\ln(1+x) + \ln\left(1 - \frac{100}{9}x\right) \tag{9}$$

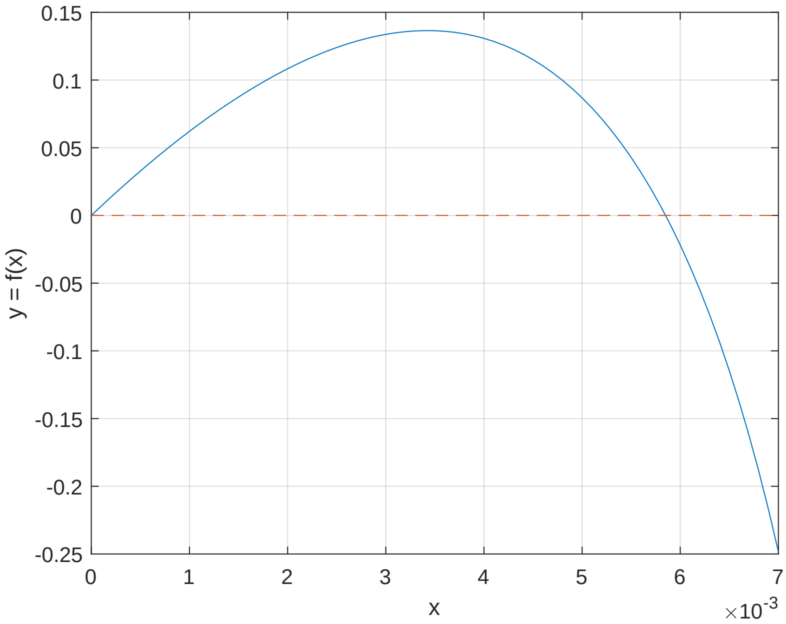


图 2: $f(x)$ 在零点附近的图像

画出 $f(x)$ 的图像，如图 3所示，可见 $f(x)$ 的唯一零点在 0.065 附近，将其作为初值，用 `fzero` 求解得到结果为 $\hat{x} = 6.3949\%$，其对应的绝对误差为 $e = f(\hat{x}) = -4.4409 \times 10^{-15}$。

2.1.7 结果的数学分析

在三个方程中，方程的绝对误差均控制在 10^{-13} 以内，十分精确，具有实际参考价值。

2.1.8 结果的实际意义

对于第二小问，第一家银行的贷款月利率为 0.5851%，第二家银行的贷款年利率为 6.3949%，按照年利率等于 12 倍月利率计算，则第二家银行的贷款月利率为 0.5329%，比第一家银行的月利率更低，因此，从利率角度看，第二家银行更优惠。

如果从还款总额来看，第一家银行的还款总额为 81 万，第二家银行的还款总额为 90 万，若不考虑通货膨胀等因素，第一家银行将比第二家银行更优惠。因此，在向银行贷款时，应当综合多方面因素考虑其贷款政策，做出合理的决定。

本题根据按揭贷款的本金总额，还款期数，以及每期还款金额，求出了贷款利率，该结果具有一定的参考价值。然而，在现实生活中，通常是给定本金总额，还款期数，以及贷款基准利率，求每期还款金额，这往往是借款人和贷款机构最关心的事情。

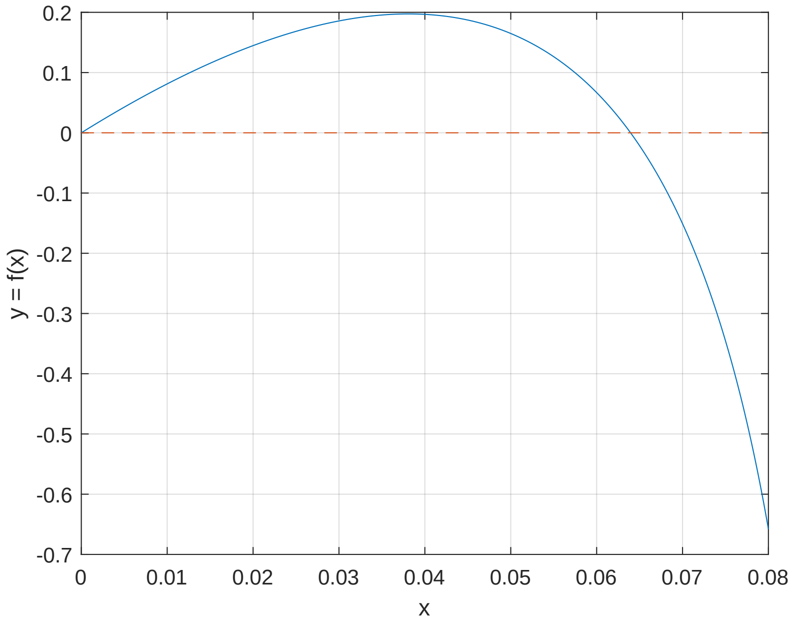


图 3: $f(x)$ 在零点附近的图像

2.1.9 结论

小张夫妇的贷款月利率为 0.2081%。从利率角度看，第二家银行更优惠。

2.2 Chap6-Ex6 均相共沸混合物（应用题）

2.2.1 问题分析

题目给定四种物质对应的参数和相互作用矩阵，需要求出给定压强下的共沸混合物的温度和组分，这是一个非线性方程组求解问题。

2.2.2 模型假设

考虑到实际情况，模型的建立基于以下假设，

1. 给定的物质能混合共存，不发生相互化学作用。
2. 均相共沸混合物满足稳定条件。

2.2.3 模型建立

设该混合物由 n 个组分组成，组分 i 所占比例为 x_i $(i = 1, 2, \cdots, n)$，则有，

$$\sum_{i=1}^{n} x_i = 1, \quad x_i \geq 0 \tag{10}$$

由假设 2，均相共沸混合物应该满足稳定条件，在压强 P 不大时，该条件可以表示为，

$$P=\gamma_i P_i, \quad i=1,2,\cdots,n \tag{11}$$

其中 P_i 为组分 i 的饱和汽相压强，γ_i 为组分 i 的液相活度系数。P_i 可根据下式确定，

$$\ln P_i=a_i-\frac{b_i}{T+c_i}, \quad i=1,2,\cdots,n \tag{12}$$

其中 a_i,b_i,c_i 为常数，T 为温度。γ_i 可根据下式确定，

$$\ln\gamma_i=1-\ln\left(\sum_{j=1}^{n}x_j q_{ij}\right)-\sum_{j=1}^{n}\left(\frac{x_j q_{ji}}{\sum_{k=1}^{n}x_k q_{jk}}\right), \quad i=1,2,\cdots,n \tag{13}$$

其中 q_{ij} 表示组分 i 与组分 j 的交互作用参数，q_{ij} 组成交互作用矩阵 Q。将方程 (11) 两边取对数后，将方程 (12)，方程 (13) 带入其中，考虑到只有组分 i 参与到共沸混合物时才满足上述条件，因此需要添加一个组分因子 x_i，得到，

$$x_i\left[1-\ln\left(\sum_{j=1}^{n}x_j q_{ij}\right)-\sum_{j=1}^{n}\left(\frac{x_j q_{ji}}{\sum_{k=1}^{n}x_k q_{jk}}\right)+a_i-\frac{b_i}{T+c_i}-\ln P\right]=0 \tag{14}$$

其中 $i=1,2,\cdots,n$。

方程 (10) 和方程 (14) 共有 $n+1$ 个方程，其中含 $n+1$ 个变量，组成了一个非线性方程组，构成了此题的模型。

2.2.4 算法设计

在实际求解过程中，首先利用方程 (10) 消去其中一个变量，即，

$$x_n=1-\sum_{i=1}^{n-1}x_i \tag{15}$$

并将其带入方程 (14) 中，得到含有 n 个变量的 n 个非线性方程，利用 Matlab 优化工具箱的 `fsolve` 命令即可求出该非线性方程组的解，需要注意的是，在不同初值条件下方程可能有不同的解，因此需要对初值进行搜索和试探。

对于误差分析，记所求的方程组为 $\boldsymbol{f}(\boldsymbol{x})=\boldsymbol{0}$，其中 $\boldsymbol{f}=(f_1,f_2,\cdots f_n)$ 代表 n 个方程对应的函数，$\boldsymbol{x}=(x_1,\cdots,x_{n-1},T)$ 为 n 个未知数，设求出的一组解为 $\hat{\boldsymbol{x}}$，则定义方程的绝对误差为 $\boldsymbol{e}=\boldsymbol{f}(\hat{\boldsymbol{x}})$，通过计算误差的 p-范数，可以大致了解方程解的精确程度。

2.2.5 Matlab 程序

请参见附录4.2。

表 1: 不同初值条件下求出的混合物组分和温度，以及方程的绝对误差（$\times 10^{-5}$）

No.	x_1	x_2	x_3	x_4	T	$\|\|\boldsymbol{e}\|\|_1$	$\|\|\boldsymbol{e}\|\|_2$	$\|\|\boldsymbol{e}\|\|_\infty$
1	0.6247	0.3753	0.0000	0.0000	58.1358	0.0000	0.0000	0.0000
2	0.0000	0.5858	0.4142	0.0000	71.9657	0.0265	0.0160	0.0111
3	0.0000	0.0000	1.0000	0.0000	82.5567	0.0005	0.0003	0.0002
4	0.0000	0.0000	0.0000	1.0000	97.7712	0.0000	0.0000	0.0000
5	0.0000	0.7803	0.0000	0.2197	76.9613	0.0001	0.0001	0.0000
6	0.0000	1.0000	0.0000	0.0000	80.1162	0.0000	0.0000	0.0000
7	0.0000	0.0000	0.0000	1.0000	97.7711	0.4756	0.4365	0.4356
8	1.0000	0.0000	0.0000	0.0000	64.5465	0.0001	0.0001	0.0001
9	0.0000	0.0000	0.0000	1.0000	97.7709	1.1153	0.9356	0.9283

2.2.6 计算结果

设置四重循环搜索初值条件，第一重遍历 $x_1 \in \{0.0, 0.1, \cdots, 1.0\}$，第二重遍历 $x_2 \in \{0.0, 0.1, \cdots, 1 - x_1\}$，第三重遍历 $x_3 \in \{0.0, 0.1, \cdots, 1 - x_1 - x_2\}$，第四重遍历 $T \in \{0, 10, \cdots, 100\}$，对于给定的一组初值 (x_1, x_2, x_3, T)，求解非线性方程组。

经过初值搜索，一共得到 3025 组解，其中大部分是重复解，经过去重并过滤掉不合法的解后，合法的方程解如表 1所示。

根据题意，需要形成共沸混合物，其中至少有两种物质的组分，因此需要进一步过滤掉仅含有一种组分的方程解，得到最终可能的共沸混合物组分及温度，共有三种情况，列举如下，

1. $x_1 = 62.47\%, \quad x_2 = 37.53\%, \quad x_3 = 0.00\%, \quad x_4 = 0.00\%, \quad T = 58.1358$
2. $x_1 = 0.00\%, \quad x_2 = 58.58\%, \quad x_3 = 41.42\%, \quad x_4 = 0.00\%, \quad T = 71.9657$
3. $x_1 = 0.00\%, \quad x_2 = 78.03\%, \quad x_3 = 0.00\%, \quad x_4 = 21.97\%, \quad T = 76.9613$

2.2.7 结果的数学分析

由表 1可以看出，方程解的误差的 1-范数，2-范数，以及 ∞-范数均控制在 10^{-4} 以内，最终选取的三组解的误差控制在 10^{-6} 以内，因此方程的解较为精确，最终结果可作为实际应用的参考。

2.2.8 结果的实际意义

在压强为 760 mmHg 下，为了形成均相共沸混合物，计算机给出了三组可能的组分和温度结果，三组结果都十分精确地满足了稳定性条件，具有实际参考

价值。在此基础上，需要进一步通过实验验证其可行性，该理论计算的结果对实际实验具有重要的指导意义。

2.2.9 结论

在压强为 760 mmHg 下，为了形成均相共沸混合物，温度和组分有三种可能的情况，分别为，

1. 温度为 58.1358 度，组分 1 占 62.47%，组分 2 占 37.53%，无其余组分。
2. 温度为 71.9657 度，组分 2 占 58.58%，组分 3 占 41.42%，无其余组分。
3. 温度为 76.9613 度，组分 2 占 78.03%，组分 4 占 21.97%，无其余组分。

2.3 Chap6-Ex8 价格混沌（计算题）

2.3.1 算法设计

由题意，商品在 t 时期的市场价格为 p_t，需求函数为 $D(p_t)=c-dp_t,\ (c,d>0)$，生产方期望价格为 q_t，供应函数为 $S(q_t)$，供销平衡时 $S(q_t)=D(p_t)$。生产方在 $t+1$ 时期会将价格调整为 q_{t+1}，其中 $q_{t+1}-q_t=r[p(t)-q(t)],\ (0<r<1)$。设 $S(x)=\arctan(\mu x),\mu=0.8,d=0.25,r=0.3$，以 c 为可变参数。

由上述条件，可得出 q_t 的差分方程为，

$$q_{t+1}=\frac{r}{d}\left[c-\arctan(\mu q_t)\right]+(1-r)q_t \tag{16}$$

为了观察其分岔与混沌现象，设置参数集 $\mathcal{C}=\{c_i\}_{i=1}^m$，其中 c_i 按大小排序，在每个固定的参数 $c_i\in\mathcal{C}$ 下，迭代 n 次计算序列 q_t，并画出序列的第 n_s 到第 n 个点。

为了找到具体的分岔点数值，观察到当 t 充分大时，分岔序列 q_t 是周期性的，且周期恰为分岔数，因此，首先在每个参数 c_i 下，求出当 t 充分大时 q_t 的周期 T_i，其中 T 必然为 2 的某个幂，然后遍历每个参数 c_i，若其对应 q_t 的周期 $T_i=2T_{i-1}$，则 c_i 即为一个分岔点。

2.3.2 Matlab 程序

请参见附录4.3。

2.3.3 计算结果

当参数 c 在 $[-2,2]$ 区间内变化时，生产方期望价格会出现分岔与混沌现象，如图 4所示。由于题目说明 $c>0$，因此这里只关心 $c>0$ 的情况，即从右向左寻找分岔点。

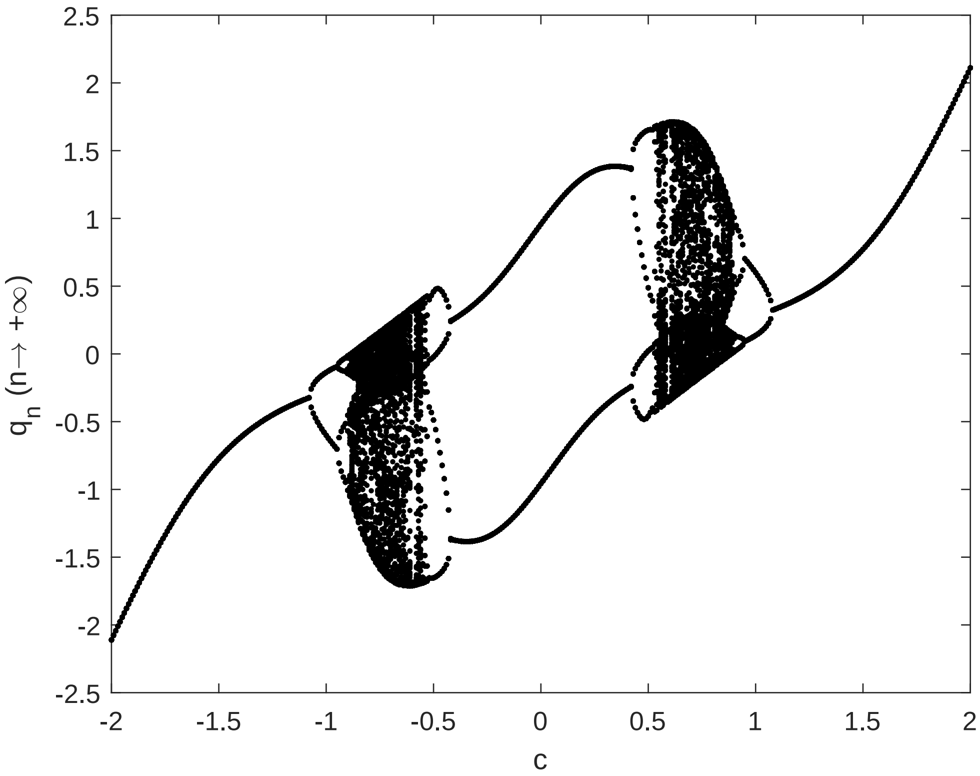


图 4: 期望价格 q_t 的分岔与混沌现象

表 2: 前六个分岔点位置，其中 c_n 表示第 n 个分岔点。

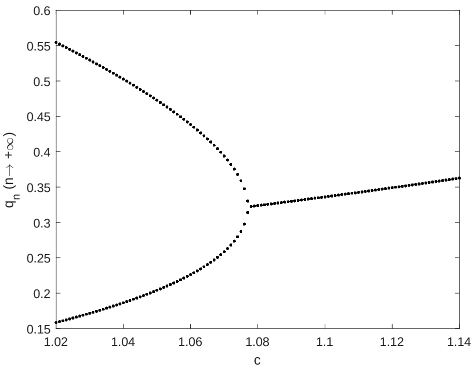
n	1	2	3	4	5	6
c_n	1.0795	0.9486	0.9071	0.8971	0.8948	0.8943
$\frac{c_n-c_{n-1}}{c_{n+1}-c_n}$	/	3.1542	4.1500	4.3478	4.6000	/

前四个分岔点位置附近的图像如图 5所示，后续的分岔点不一一画出，具体数值如表 2所示，表格中列出了前六个分岔点，其中 c_n 为第 n 个分岔点对应的参数 c 的值，根据这些分岔点 c_n，可计算出其差值比 $\dfrac{c_n - c_{n-1}}{c_{n+1} - c_n}$，同样列在了表格中。

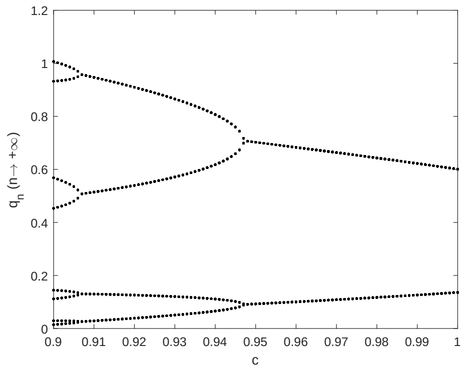
为了讨论价格 q_t 的变化规律，这里根据分岔点的位置选取了 6 个参数 c 值，分别对应着 1 分岔，2 分岔，4 分岔，8 分岔，16 分岔，以及 32 分岔，在每个 c 值下对应的 q_t 随 t 变化的图像如图 6所示。

2.3.4 结果分析

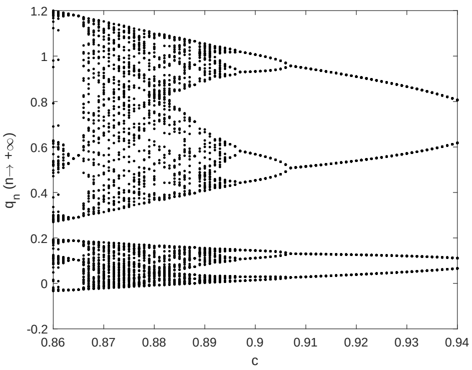
从图 4可以看出，随着参数 c 的变化，方程 (16) 的序列极限出现了分岔现象，每次分岔时分岔数量增加到原来的两倍，因此分岔数呈指数增长，这样就导致了混沌现象。



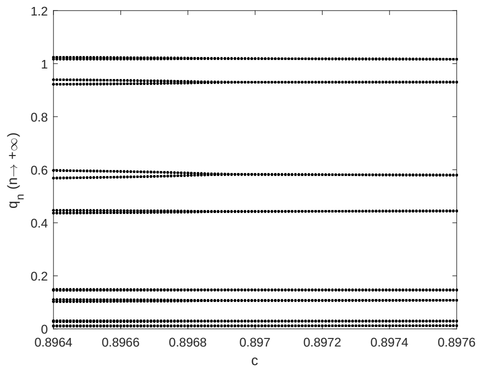
(a) 第一个分岔点



(b) 第二个分岔点



(c) 第三个分岔点



(d) 第四个分岔点

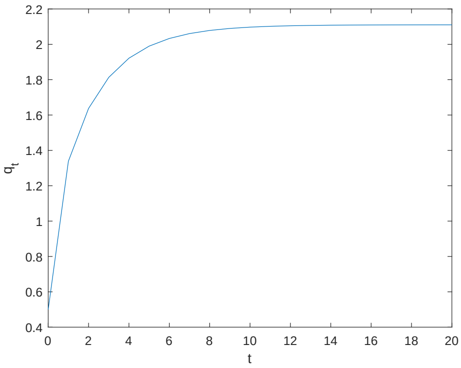
图 5: 前四个分岔点附近的图像

从表 2可以看出，随着 n 的增长，差值比 $\dfrac{c_n - c_{n-1}}{c_{n+1} - c_n}$ 逐渐趋近于 Feigenbaum 常数 4.6692，大致验证了分岔点的极限规律。从这个规律中可以看出，每经过一个分岔点，相邻两个分岔点的间隔就缩小一次，也就是说，随着参数的变化，当分岔数越多，分岔增加的速度就越快，这样类似于正反馈的分岔机制也是导致混沌的一个重要原因。

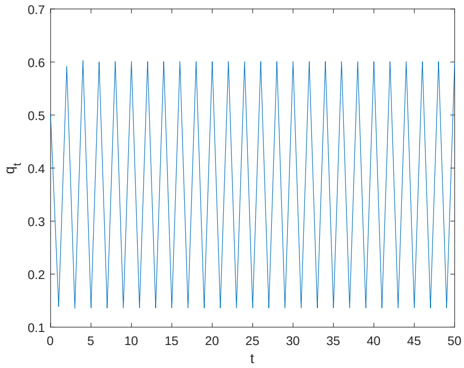
从图 6可以看出，当 $c_1 < c$ 时，价格 q_t 收敛到一个固定的值，当 $c_2 < c < c_1$ 时，价格 q_t 在两个定值之间来回振荡，有两个收敛的子序列，当 $c_3 < c < c_2$ 时，价格 q_t 在四个定值之间来回振荡，有四个收敛的子序列，以此类推，随着分岔数量的指数增长，q_t 的发散程度越来越严重，出现了混沌现象。在一定程度上，q_t 的分岔与混沌现象也反映了市场经济的周期性现象。

2.3.5 结论

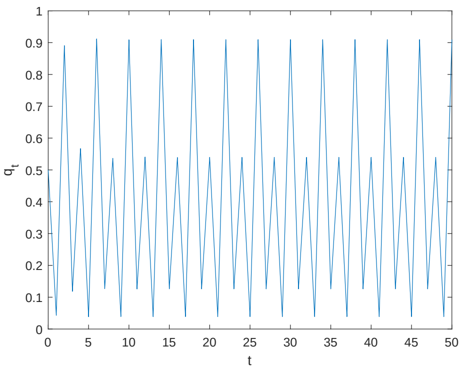
当参数 c 在区间 $[0, 2]$ 变化时，生产方期望价格会出现分岔和混沌现象，反映了市场经济的周期性现象。



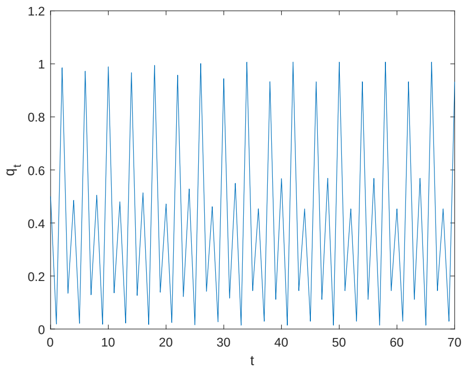
(a) 当 $c = 2.0000$ 时的序列 q_t 图像



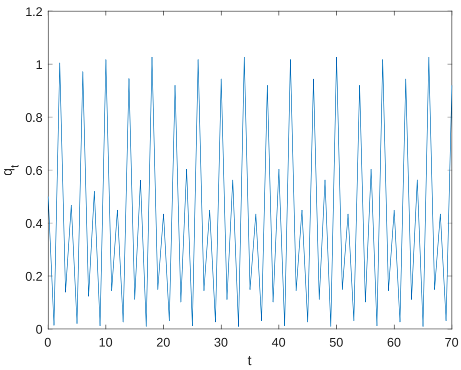
(b) 当 $c = 1.0000$ 时的序列 q_t 图像



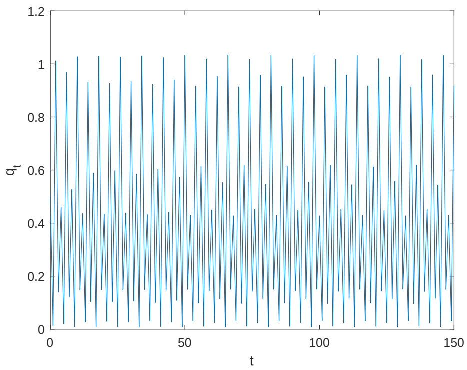
(c) 当 $c = 0.9200$ 时的序列 q_t 图像



(d) 当 $c = 0.9000$ 时的序列 q_t 图像



(e) 当 $c = 0.8960$ 时的序列 q_t 图像



(f) 当 $c = 0.8945$ 时的序列 q_t 图像

图 6: 参数 c 取不同值下的序列 q_t 图像

3 收获与建议

在本次实验中，我掌握了用 MATLAB 软件求解非线性方程和方程组的基本方法，并对结果进行了初步分析，用非线性方程和方程组建立了实际问题的模型，在解决实际问题的过程中，我对数学方法的原理和应用有了更深刻的理解。

希望助教能对每次的实验进行详细的解答，希望老师在未来的课堂上介绍更多数学应用的前沿知识。

4 附录：Matlab 程序代码

4.1 Chap6-Ex3

```matlab
N = 20;
Q = 50;
q = 4.5;

fun = @(x) N * log(1 + x) + log(1 - Q / q * x);
x = 0:0.0001:0.08;
y = fun(x);
figure; plot(x, y, x, zeros(size(x)), '--');
xlabel('x'); ylabel('y = f(x)'); grid on;

[x, fval] = fzero(fun, 0.065);
x
fval
```

4.2 Chap6-Ex6

```matlab
Q = [1.0 0.192 2.169 1.611;
    0.316 1.0 0.477 0.524;
    0.377 0.360 1.0 0.296;
    0.524 0.282 2.065 1.0];
n = 4;
a = [18.607 15.841 20.443 19.293]';
b = [3643.31 2755.64 4628.96 4117.07]';
c = [239.73 219.16 252.64 227.44]';
P = 760;

% search and solve
results = zeros(4, 2000);
results_cnt = 0;
for x1 = 0:0.1:1
```

```
    for x2 = 0:0.1:1-x1
        for x3 = 0:0.1:1-x1-x2
            for T = 0:10:100
                XT0 = [x1 x2 x3 T]';
                [XT, val] = fsolve(@eqn, XT0, [], n, P, a, b, c, Q);
                x = [XT(1:n-1); 1 - sum(XT(1:n-1))];
                results_cnt = results_cnt + 1;
                results(:, results_cnt) = XT';
            end
        end
    end
end

% filter results
results = real(results);
valid_results = results(:, 1);
valid_cnt = 1;

for result = results
    if min(sum(abs(result - valid_results), 1)) > 1e-4
        if all(result(1:3) > -1e-5) && sum(result(1:3)) < 1 + 1e-5
            valid_cnt = valid_cnt + 1;
            valid_results(:, valid_cnt) = result;
        end
    end
end

% error analysis
err = zeros(4, valid_cnt);
err_1 = zeros(1, valid_cnt);
err_2 = zeros(1, valid_cnt);
err_inf = zeros(1, valid_cnt);
for i = 1:valid_cnt
    err(:, i) = eqn(valid_results(:, i), n, P, a, b, c, Q);
    err_1(i) = norm(err(:, i), 1);
    err_2(i) = norm(err(:, i), 2);
    err_inf(i) = norm(err(:, i), inf);
end

% equation to solve
function y = eqn(XT, n, P, a, b, c, Q)
    x = [XT(1:n-1); 1 - sum(XT(1:n-1))];
    T = XT(n);
    Qx = Q * x;
    y = x .* (1 - log(Qx) - Q' * (x ./ Qx) + a - b ./ (T + c) - log(P
```

```
        ));
end
```

4.3 Chap6-Ex8

```
mu = 4.8;
d = 0.25;
r = 0.3;
fun = @(q, c) r/d*(c - atan(mu * q)) + (1-r) * q;

ps_fork1 = 1.02:0.001:1.14;
ps_fork2 = 0.9:0.001:1;
ps_fork3 = 0.86:0.001:0.94;
ps_fork4 = 0.8964:0.00001:0.8976;
ps_forks = 0.86:0.0001:1.09;
ps_global = -2:0.01:2;

ps = ps_forks;
max_iter = 1000;
x1 = 0.5;

q = chaos(fun, x1, ps, max_iter);
% figure; plot(0:150, q(10000, 1:151)); xlabel('t'); ylabel('q_t');
forks_index = get_forks(q);
forks = ps(forks_index);
forks
e = length(forks);
fegenbaum = (forks(2:e-1) - forks(3:e)) ./ (forks(1:e-2) - forks(2:e-1));
fegenbaum

figure; plot(ps, q(:, max_iter - 98:max_iter + 1), 'k.');
xlabel('c'); ylabel('q_n (n\rightarrow +\infty)');

function x = chaos(fun, x1, ps, max_iter)
% observe chaos while changing the param of the difference equation
% fun: iterated function of difference quation x_{n+1} = fun(x_n, p),
%       where the changeable param p might cause chaos
% x1: initial value of the sequence x, i.e., x_1
% ps: values of the changing param p to investigate
% max_iter: max iterations of the difference equation
% return: the sequence x under each param value specified in ps,
%       where x(i, j) is the value of x(j) under i-th param value
    x = zeros(length(ps), max_iter + 1);
```

```
    for p_index = 1:length(ps)
        p = ps(p_index);
        x(p_index, 1) = x1;
        for iter = 1:max_iter
            x(p_index, iter + 1) = fun(x(p_index, iter), p);
        end
    end
end

function forks_index = get_forks(xs)
    [num_p, max_iter] = size(xs);
    ways = ones(num_p, 1);
    for index = 1:num_p
        x = xs(index, :);
        while 1
            if max_iter < 2 * ways(index)
                ways(index) = -1;
                break;
            end
            last = x(max_iter - ways(index) + 1:max_iter);
            prev = x(max_iter - 2 * ways(index) + 1:max_iter - ways(
                index));
            if mean(abs(last - prev)) < 1e-6 * ways(index)
                break
            end
            ways(index) = ways(index) * 2;
        end
    end

    edge_cnt = 0;
    for i = 2:length(ways)
        if ways(i) ~= ways(i-1) && ways(i) ~= -1 && ways(i-1) ~= -1
            edge_cnt = edge_cnt + 1;
            forks_index(edge_cnt) = i;
        end
    end
end
```