数学实验第四次实验报告

计 76　张翔　2017011568

2020 年 3 月 25 日

1　实验目的

1. 掌握用 MATLAB 软件求解非线性方程和方程组的基本用法, 并对结果作初步分析;

2. 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　Ch7-P3 按揭贷款

2.1　模型建立

对于按揭贷款，设本金为 m，首付为 a，按月还款 (每月固定还款为 x)，月利率为 r，第 i 月还款后的欠款金额为 x_i，则有如下的递推关系

$$x_{i+1} = (1+r)x_i - x \quad i \geq 1, i \in \mathbb{N}$$

$$x_0 = m - a$$

解上述递推方程可知，第 n 月时还需还款

$$x_n = (1+r)^n x_0 - x\sum_{i=0}^{n-1}(1+r)^i = (1+r)^n x_0 - x\frac{(1+r)^n - 1}{r}$$

若第 n 月还清，那么令 $x_n = 0$，解该方程可以得到 r，即每个月的利率。如果按年还款，上述式子仍不变，只是相应变量按年计算，如年利率 r，每年固定还款 x。

2.2　主要算法

对于题给问题 1，将相关变量的取值代入上述式子，可以解得月利率；对于问题 2，解出两家银行的月利率和年利率后，使用题给的年利率和月利率转换关系转化后，再进行比较，即可知道哪家银行更加划算。求解非线性方程可以使用 `fzero` 函数，求解之前可以通过画图来确定解的大致范围。

实际计算时，由于 $r > 0$ 恒成立，上述方程等价变换为

$$r(1+r)^n x_0 - x(1+r)^n + x = 0 \tag{1}$$

由此移除了分母上的未知数 r，能够提高数值运算的稳定性。

2.3　Matlab 程序

```m
%% Ch7 P3
%% parameters
initial = 15;
per_money = 0.1;
time = 15 * 12;

%% draw plot
interval = 0.005;

hold on;
fplot(@(x)loan_func(x, time, initial, per_money), [0 interval]);
fplot(@(x)x.*0, [0 interval]);

%% solve eqn
[x, fv, ef, out] = fzero(@(x)loan_func(x, time, initial, per_money), 0.01)
    ;

%% function for loan
function res = loan_func(rate, time, initial, per_money)
    res = rate .* (1 + rate).^time .* initial - per_money .* ((1 + rate)
        .^ time - 1);
end
```

2.4 计算结果

对于题 1，初始欠款 $x_0 = 15$ 万元，每月还款 $x = 0.1$ 万元，共需 $n = 180$ 月还清，代入方程，画图得到

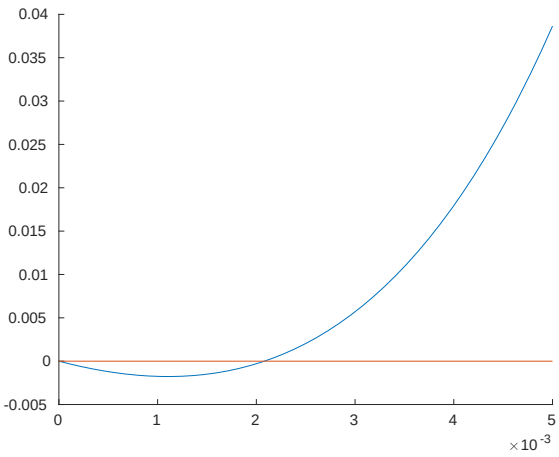
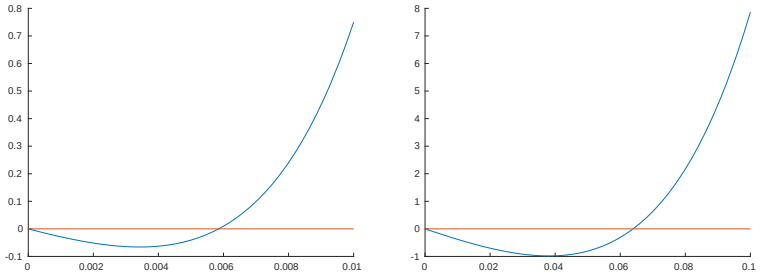


图 1: 题目 1 的 $x_n(r)$ 图线

容易得到解的区间为 $r \in [1 \times 10^{-3}, 3 \times 10^{-3}]$，取区间右端点为初值，解出 $r = 0.00208$，求解收敛。因此月利率为 0.208%

对于问题 2，第一家银行 $x_0 = 50$ 万元，每月还款 $x = 0.45$ 万元，共需 $n = 180$ 月还清；第二家银行 $x_0 = 50$ 万元，每年还款 $x = 4.5$ 万元，共需 $n = 20$ 年还清。上述参数代入方程，画图可得



(a) 银行 1　　(b) 银行 2

图 2: 两家银行对应的方程作图

解区间分别为 [0.004, 0.008]，[0.04, 0.08]，将解区间的右端点作为迭代初始值，分别得到：银行 1，$r = 0.00585$；银行 2，$r = 0.0639$。即银行 1 的月利率为 0.585%，折合年利率为 $0.585\% \times 12 = 7.02\%$；银行 2 年利率为 6.39%。显然就利率而言，银行 2 更加划算。

简单计算可知，总还款数额银行 1 为 81 万元，银行 2 为 90 万元，虽然银行 2 的利息较低，但总还款额高于银行 1。

2.5 结论

小张夫妇的贷款月利率为 0.208%。就利率而言，银行 2 的年利率为 6.39%，低于银行 1 的 7.02%，但还款总额高于银行 1，说明贷款时需要根据多种因素，综合考虑后进行选择。

3 Ch7-P6 共沸化合物

3.1 模型建立与算法设计

基本模型同课本例题，设 x_i 为第 i 种物质占比，T 为温度，P 为压强，已知参数 a_i, b_i, c_i 与交互作用矩阵 Q，则该系统满足下列方程

$$x_i \left(\frac{b_i}{T + c_i} + \ln \left(\sum_{j=1}^{n} x_j q_{ij} \right) + \sum_{j=1}^{n} \left(\frac{x_j q_{ji}}{\sum_{k=1}^{n} x_k q_{jk}} \right) - 1 - a_i + \ln P \right) = 0, \quad i = 1, 2, \cdots, n \tag{2}$$

而 $\sum_{i=1}^{n} x_i = 1$（归一化条件，组分总和为 1）可以代入上述方程，消去 x_n，从而组成 n 个未知数 $(x_1, x_2, \cdots, x_{n-1}, T)$ 的非线性方程组。该方程难以直接得到解析解，可以使用 Matlab 的 `fsolve` 函数进行计算。

实现时，上述方程左端可以使用矩阵形式表示，从而避免显式写出循环语句，从而使程序可以更好地调用向量计算的优化

$$\mathbf{x} \otimes \left(\mathbf{b} \otimes \frac{1}{T \oplus \mathbf{c}} + \ln(\mathbf{Q}\mathbf{x}) - \mathbf{a} \oplus (\ln P - 1) + \mathbf{Q}^{\mathrm{T}} \left(\mathbf{x} \otimes \frac{1}{\mathbf{Q}\mathbf{x}} \right) \right) \tag{3}$$

上述加粗的变量为矩阵/向量，带圆圈的运算符表示 element-wise 操作，否则是普通的矩阵与向量乘法或是标量运算，使用 Matlab 的点运算符可以方便地实现这些操作。

3.2 Matlab 程序

计算时 Matlab 发出警告 `Warning: Trust-region-dogleg algorithm of FSOLVE cannot handle non-square systems; using Levenberg-Marquardt algorithm instead.`，于是使用非线性优化中常用的 LM 方法进行求解，程序如下

```
%% Ch7 - P6
%% data
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];
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];
P = 760;
%% initial condition
XT0 = [0.; 0.; 0.; 50];

opt = optimoptions('fsolve', 'Algorithm', 'levenberg-marquardt');
[XT, fval, flag, out] = fsolve(@(x)azeofun(x, n, P, a, b, c, Q), XT0, opt)
    ;

res = [XT(1:n-1); 1 - sum(XT(1:n-1)); XT(n)];

function f = azeofun(XT, n, P, a, b, c, Q)
    x = [XT(1:n-1); 1 - sum(XT(1:n-1))];
    qx = Q * x;
    f = x .* ( b ./ (XT(n) + c) + log(qx) - 1 - a + log(P) + Q' * (x ./ qx
        ) );
end
```

3.3 计算结果

初值为 (0, 0, 0, 50) 时，解得

$$x_1 = 62.47\%,\ x_2 = 37.53\%,\ x_3 = 0\%,\ x_4 = 0\%,\ T = 58.14$$

初值为 (0, 0.33, 0.33, 65) 时，解得

$$x_1 = 0\%,\ x_2 = 58.58\%,\ x_3 = 41.42\%,\ x_4 = 0\%,\ T = 71.97$$

初值为 (0.25, 0.25, 0.25, 100) 时，解得

$$x_1 = 0\%,\ x_2 = 0\%,\ x_3 = 0\%,\ x_4 = 100\%,\ T = 97.77$$

初值为 (1, 0, 0, 80) 时，解得

$$x_1 = 0\%,\ x_2 = 0\%,\ x_3 = 100\%,\ x_4 = 0\%,\ T = 82.56$$

初值为 (0, 1, 0, 100) 时，解得

$$x_1 = 0\%,\ x_2 = 100\%,\ x_3 = 0\%,\ x_4 = 0\%,\ T = 80.12$$

初值为 (1, 0, 0, 70) 时，解得

$$x_1 = 100\%,\ x_2 = 0\%,\ x_3 = 0\%,\ x_4 = 0\%,\ T = 64.55$$

初值为 (0, 0.5, 0, 80) 时，解得

$$x_1 = 0\%,\ x_2 = 78.03\%,\ x_3 = 0\%,\ x_4 = 21.97\%,\ T = 76.96$$

由上述解可知，没有找到四种物质均共存的情况，说明给定压强下四种物质可能无法共同稳定存在。而共沸物至少应有 2 种物质混合，因此最后合理解应该只有三组：

- $x_1 = 62.47\%,\ x_2 = 37.53\%,\ x_3 = 0\%,\ x_4 = 0\%,\ T = 58.14$
- $x_1 = 0\%,\ x_2 = 58.58\%,\ x_3 = 41.42\%,\ x_4 = 0\%,\ T = 71.97$
- $x_1 = 0\%,\ x_2 = 78.03\%,\ x_3 = 0\%,\ x_4 = 21.97\%,\ T = 76.96$

3.3.1 结论

题给四种物质在给定压强下可能不能共同稳定存在，而可以形成共沸物的情况共有 3 种（在上面已经给出）

4 Ch7-P8 价格方程的混沌现象

4.1 模型建立与算法设计

根据题给模型，期望价格 $q(t)$ 的递推方程如下：

$$q(t+1) = \frac{r}{d}(c - \arctan(\mu q(t))) + (1-r)q(t) \tag{4}$$

使用课本中提供的 `chaos` 函数可以画图观察分岔点的大致区间，之后可以通过枚举的方式计算分岔点的精确位置。注意当 n 较大时，序列的子列（若存在）将各自收敛，若为 4 分岔，则连续取 8 个点，可以得到具有“周期”的序列 $(a_1, a_2, a_3, a_4, a_1', a_2', a_3', a_4')$，其中 $|a_i - a_i'| < \varepsilon$ 且 $|a_i - a_j| > \varepsilon$，这个周期就是分岔的数目。因此，通过列举参数 c，并在 n 较大时连续取样 $2 \times$ 分岔数 个点，通过上述判据可以检测到是否发生分岔，从而确定分岔点的位置。确定分岔点时，可以采用二**分法**，因为分岔情况一定是从 $2^n \to 2^{n+1}$。

4.2 Matlab 程序

主程序如下：

```
%% Ch7 - P8
%% parameters
mu = 4.8;
d = 0.25;
r = 0.3;
init_val = 0.5;

```

```
figure(1),
chaos(@(x, c)iter_func(x, r, d, c, mu), init_val, [0, 2, 0.001], [1000,
    1200]),
xlabel('c'),
ylabel('a_n(n\rightarrow \infty)');

%% calc fork point
n_iters = 1000;
level = 1;
fp = find_fork(@(x, c)iter_func(x, r, d, c, mu), init_val, level, n_iters,
     [1.07, 1.08], 1e-6, 1e-7);

%% draw graph
c = 1;
iters = zeros(n_iters, 1);
iters(1) = init_val;
for i = 2:n_iters
    iters(i) = feval(@(x, c)iter_func(x, r, d, c, mu), iters(i-1), c);
end
figure(),
plot(iters(n_iters - 20 : n_iters));

function y=iter_func(x, r, d, c, mu)
    y = r ./ d .* (c - atan(mu .* x)) + (1 - r) .* x;
end
```

chaos 函数与课本一致：

```
function chaos(iter_fun,x0,r,n)
kr=0;
for rr=r(1):r(3):r(2)
    kr=kr+1;
    y(kr,1)=feval(iter_fun,x0,rr);
    for i=2:n(2)
        y(kr,i)=feval(iter_fun,y(kr,i-1),rr);
    end
end
plot(r(1):r(3):r(2),y(:,n(1)+1:n(2)),'k.');
```

find_fork 函数是自动寻找分岔点的函数，它通过 test_fork 判断是否分岔，并通过二分的方法求解，如下：

```
function fp = find_fork(iter_fun, init_val, level, converge_iter, r_range,
    r_tol, tol)
    r_lo = min(r_range);
    r_hi = max(r_range);
    hi_fork = test_fork(iter_fun, init_val, level, converge_iter, r_hi,
```

```
        tol);
    lo_fork = test_fork(iter_fun, init_val, level, converge_iter, r_lo,
        tol);
    if(~xor(hi_fork, lo_fork))
        error("Unable to find a fork point");
    end
    while r_hi - r_lo > r_tol
        mid = (r_hi + r_lo) / 2;
        mid_fork = test_fork(iter_fun, init_val, level, converge_iter, mid
            , tol);
        if(mid_fork == hi_fork)
            r_hi = mid;
        else
            r_lo = mid;
        end
    end
    fp = r_hi;
end

function res = test_fork(iter_fun, init_val, level, converge_iter, r, tol)
    iter_conv = init_val;
    %% iterate till converge
    for i = 1 : converge_iter
        iter_conv = feval(iter_fun, iter_conv, r);
    end
    %% sample the pattern
    n = 2 ^ (level + 1);
    samples = zeros(n, 1);
    for i = 1:n
        samples(i) = feval(iter_fun, iter_conv, r);
        iter_conv = samples(i);
    end

    res = reshape(samples, [], 2);
    split = abs(res(:, 1) - res(:, 2));
    discrep = abs(res(1:2^(level-1),1) - res(2^(level-1)+1:2^level,1));
    res = (all(split < tol) && all(discrep > tol));
end
```

4.3 计算结果

使用 chaos 函数得到的分岔图如下

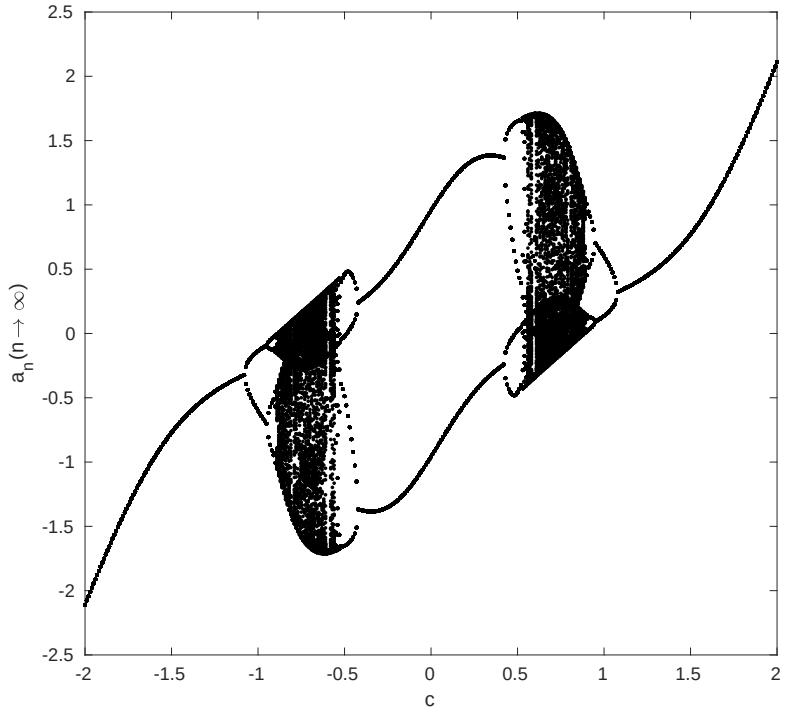
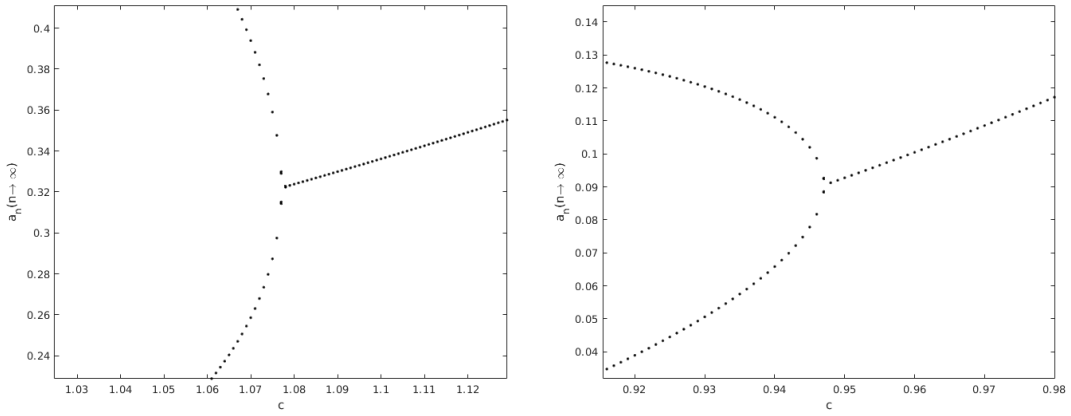


图 3: 期望价格方程关于参数 c 的分岔图

可以看出，该方程的分岔图关于 $(0, 0)$ 中心对称，但 $c < 0$ 时 $q < 0$，不符合价格的物理意义，因此实际计算时只需要考虑 $c > 0$ 的情况。

这里选取较为明显的前三个分岔点作图，如下



(a) 分岔点 1　　(b) 分岔点 2

图 4: 分岔点邻域图

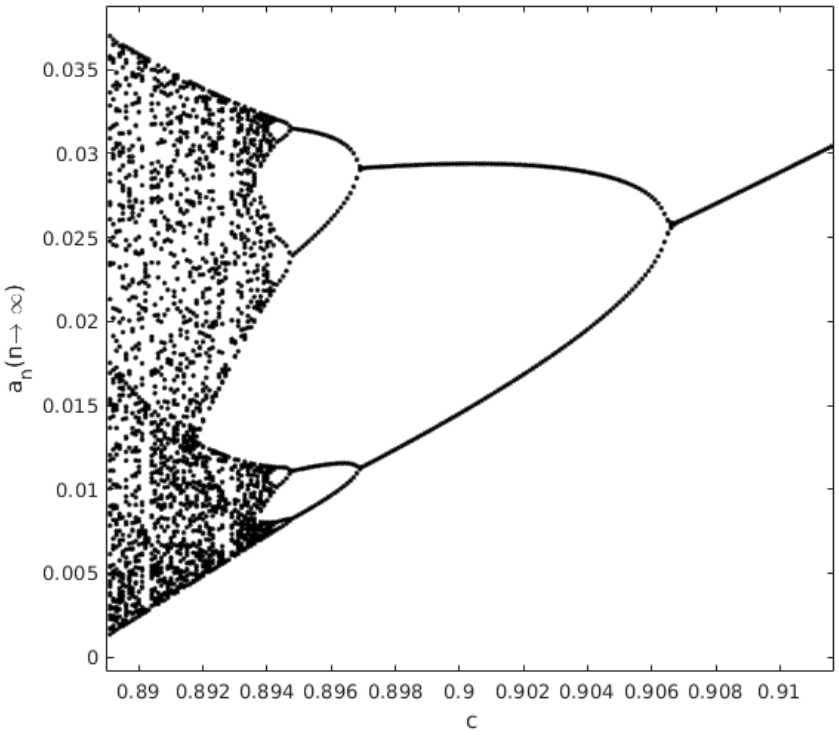


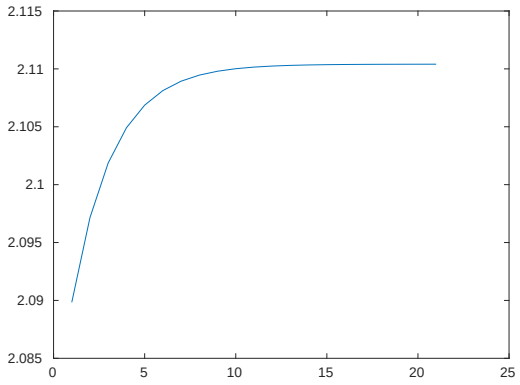
图 5: 分岔点 3, 4, 5 邻域图

由上图可以看出

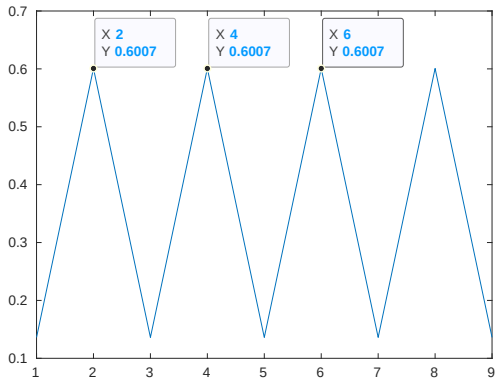
- 分岔点 1 在区间 [1.07, 1.08] 上，精确计算得到 $b_1 = 1.0761$；
- 分岔点 2 在区间 [0.94, 0.95] 上，精确计算得到 $b_2 = 0.94631$；
- 分岔点 3 在区间 [0.9, 0.91] 上，精确计算得到 $b_3 = 0.90607$;
- 分岔点 4 在区间 [0.896, 0.9] 上，精确计算得到 $b_4 = 0.89670$；
- 分岔点 5 在区间 [0.8945, 0.8955] 上，精确计算得到 $b_5 = 0.89470$。

当 c 再减小时 ($c < 0.89$)，已经难以通过观察确定下一个分岔点所在区间了，出现了明显的混沌现象。

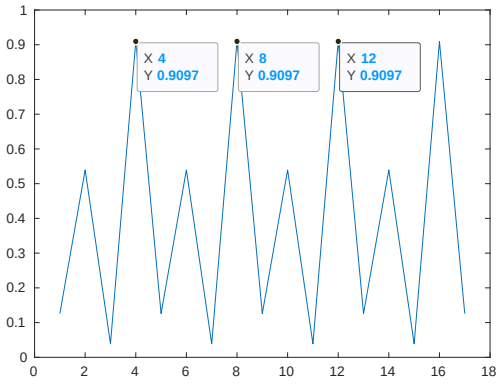
可以通过取定参数 c 画图的方式验证上述分岔点的正确性，如下



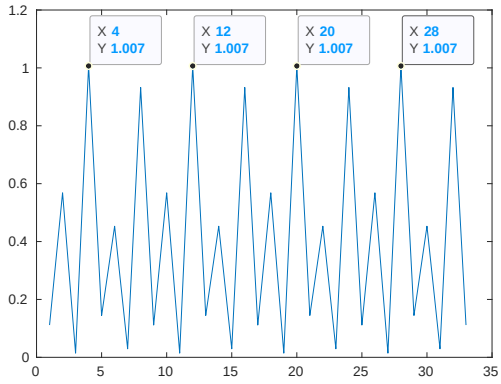
(a) $c = 2.0$, $c > b_1$，序列收敛



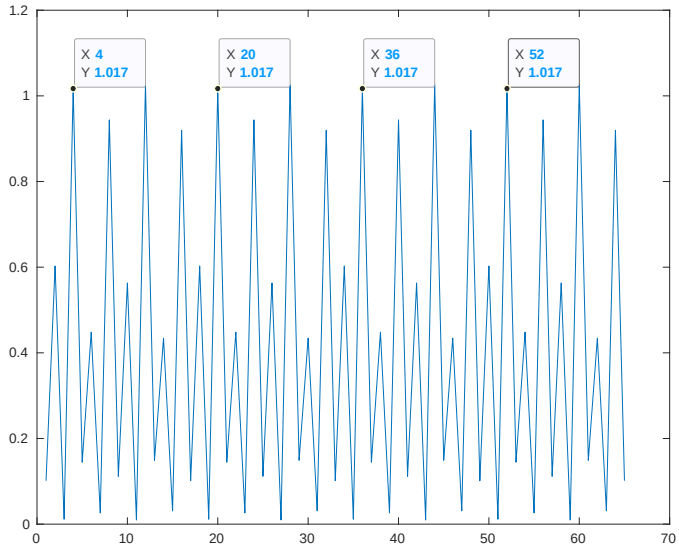
(b) $c = 1.0$, $b_2 < c < b_1$，有 2 个收敛子列



(c) $c = 0.92$, $b_3 < c < b_2$，有 4 个收敛子列



(d) $c = 0.90$, $b_4 < c < b_3$，有 8 个收敛子列



(e) $c = 0.896$, $b_5 < c < b_4$，有 16 个收敛子列

图 6: 不同参数 c 下，序列的收敛情况

上述结果说明分岔点的选取是正确的。可以看出，随着 c 参数的减小，期望价格的波动的子列越来越多。最后，计算

$$\frac{b_2-b_1}{b_3-b_2}=3.225,\ \frac{b_3-b_2}{b_4-b_3}=4.295,\ \frac{b_4-b_3}{b_5-b_4}=4.685$$

可以看出，$\frac{b_n-b_{n-1}}{b_{n+1}-b_n}$ 确实趋近于 4.669，符合 Feigenbaum 常数定律。

4.4 结论

题给的期望价格方程随着参数 c 的减小，价格波动的子列增多（出现更多分岔），并在 $c<0.89$ 时出现明显的混沌现象，且分岔点的极限趋势符合 Feigenbaum 常数。

5 收获与建议

通过本次实验，我对非线性方程组的求解有了更深刻的认识，通过 Matlab 编程、画图的方式观察了方程的求解结果以及混沌现象，并成功将二分法的思想迁移到混沌方程的分岔点求解问题上。建议：课本上求解共沸化合物的方程时最好改成用矩阵操作的方式，避免使用循环，否则在共沸物种类 n 较大时将会显著影响性能。