

大作业 4 FPU 模型

张钰坤

2000011314

(C 语言实现)

2022 年 4 月 8 日

目录

| | |
|-----------------------------|-----------|
| 1 题目解答 | 2 |
| 1.1 第一问 | 2 |
| 1.2 第二问 | 4 |
| 1.2.1 思路分析 | 4 |
| 1.3 第三问 | 5 |
| 1.4 第四问 | 5 |
| 1.5 第五问 | 6 |
| 1.6 第六问 | 14 |
| 1.6.1 类似第二问的结果 | 15 |
| 1.6.2 类似第四问的结果 | 15 |
| 1.6.3 类似第五问结果 | 16 |
| 1.7 第七问 | 24 |
| 1.8 第八问 | 25 |
| 2 附录 (源代码) | 25 |
| 2.1 "q2.c" | 25 |
| 2.2 "q4.c" | 28 |
| 2.3 "q5.c" | 30 |
| 2.4 "plotTool.py" | 32 |
| 2.5 "q6_to_q2.c" | 33 |
| 2.6 "q6_to_q4.c" | 35 |
| 2.7 "q6_to_q5.c" | 38 |
| 2.8 "q7.c" | 40 |
| 2.9 "q8.c" | 43 |
| 2.10 "q8.py" | 45 |

1 题目解答

1.1 第一问

下面解析地证明系统本征模式具有题目所给形式。

对于没有高次项的情形，体系哈密顿量具有形式

$$H = \frac{1}{2} \sum_{j=1}^n p_j^2 + \frac{1}{2} \sum_{j=0}^n (q_j - q_{j+1})^2 \quad (1)$$

$$= \frac{1}{2} \mathbf{p}^\top \mathbf{p} + \frac{1}{2} \mathbf{q}^\top \mathbf{K} \mathbf{q} \quad (2)$$

其中，

$$\mathbf{K} = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & \cdots & 0 & 0 \\ 0 & -1 & 2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2 & -1 \\ 0 & 0 & 0 & \cdots & -1 & 2 \end{pmatrix}$$

欲求系统本征模式，需要将 \mathbf{K} 对角化。

下面将会证明：

$$\mathbf{K} = \mathbf{P}^\top \mathbf{D} \mathbf{P} \quad (3)$$

$$P_{kj} = \sqrt{\frac{2}{n+1}} \sin \frac{\pi k j}{n+1}, \mathbf{D} = \text{diag}\{\omega_1^2, \dots, \omega_n^2\}, \omega_k = 2 \sin \frac{\pi k}{2(n+1)} \quad (4)$$

注意到可以将 \mathbf{K} 写为

$$\mathbf{K} = 2\mathbf{I} - \mathbf{T}$$

其中，

$$\mathbf{T} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

可以知道， \mathbf{K} 和 \mathbf{T} 的本征向量相同， \mathbf{K} 的本征值 μ 和 \mathbf{T} 的本征值 λ 满足关系 $\mu = 2 - \lambda$ 。因此，我们只要求得 \mathbf{T} 的本征值、本征向量，就可以得到 \mathbf{K} 的本征值、本征向量。

设 \mathbf{T} 的本征值 $\lambda = 2c$ （实对称矩阵 \mathbf{T} 一定有 $c \in \mathbb{R}$ ），本征向量 $\mathbf{v} = (v_1, v_2, \dots, v_n)^\top$ 。我们稍后将会看到，把 λ 设为 $2c$ 只是为了数学上的方便。

于是有，

$$0 = (\mathbf{T} - \lambda \mathbf{I})\mathbf{v} = \begin{pmatrix} -2c & 1 & 0 & \cdots & 0 & 0 \\ 1 & -2c & 1 & \cdots & 0 & 0 \\ 0 & 1 & -2c & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -2c & 1 \\ 0 & 0 & 0 & \cdots & 1 & -2c \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{n-1} \\ v_n \end{pmatrix}$$

$$= \begin{pmatrix} -2cv_1 + v_2 \\ v_1 - 2cv_2 + v_3 \\ \vdots \\ v_{j-1} - 2cv_j + v_{j+1} \\ \vdots \\ v_{n-2} - 2cv_{n-1} + v_n \\ v_{n-1} - 2cv_n \end{pmatrix}$$

除了第一个和最后一个方程，其他都有 $v_{j-1} - 2cv_j + v_{j+1} = 0$ 的形式，为了使第一个和最后一个方程也具有这样的形式，我们引入 $v_0 = v_{n+1} = 0$ ，这样所有 v_i 都满足如下关系

$$v_{j-1} - 2cv_j + v_{j+1} = 0, j = 1, 2, \dots, n \quad (5)$$

如果我们把 $\mathbf{v} = (v_1, v_2, \dots, v_n)^\top$ 中每个元素看成一个数列 $\{v_j\}$ ，那么上式给出了这个数列的二阶递推关系。根据二阶常系数递推数列的一般解法，我们设 $v_j = r^j$ ，方程 1.1 给出 $1 - 2cr + r^2 = 0$ ，得到两个根。

$$r_\pm = c \pm \sqrt{c^2 - 1}$$

我们将对 $c^2 = 1$ 和 $c^2 \neq 1$ 分别讨论。

当 $c^2 = 1$ 时， $r=c$ 并且方程 1.1 的通解为

$$v_j = (A + Bj)c^j$$

根据边界条件 $v_0 = v_{n+1} = 0$ ，得到 $A = 0, (A + B(n+1))c^{n+1} = 0$ ，于是 $A = B = 0$ 。由于特征向量一定非零，所以 c^2 一定不为 1。

当 $c^2 \neq 1$ 时，设 $r := r_+ = c + \sqrt{c^2 - 1}$ ，于是 $r_- = c - \sqrt{c^2 - 1} = 1/r$ ，这样，方程 1.1 的解就可以写为

$$v_j = Ar_+^j + Br_-^j = Ar^j + Br^{-j}, j = 0, \dots, n+1$$

应用边界条件 $v_0 = 0$ 得到 $A + B = 0$ ，我们得到

$$v_j = A(r^j - r^{-j}), j = 0, \dots, n+1$$

由于特征向量非零，需要 $A \neq 0$ ，再根据边界条件 $v_{n+1} = 0$ 我们得到

$$r^{n+1} - r^{-(n+1)} = 0 \implies r^{2(n+1)} = 1$$

设 $r = e^{i\theta}$ ，则 $c = \cos \theta$ 且 $1 = e^{2i(n+1)\theta}$ 。于是 $\theta = k\pi/(n+1), 1 \leq k \leq n$ （这里排除掉 $k=0$ 和 $k=n+1$ 是因为 $c^2 \neq 1$ ）。接着，取 $A = 1/(2i)$ ，我们就得到了 \mathbf{T} 的特征向量

$$\mathbf{v}_k = \left(\sin \frac{\pi k}{n+1}, \sin \frac{2\pi k}{n+1}, \dots, \sin \frac{n\pi k}{n+1} \right)^\top$$

由于

$$\begin{aligned}
& \forall k = 1, 2, \dots, n \\
& \sin^2 \frac{\pi k}{n+1} + \sin^2 \frac{2\pi k}{n+1} + \dots + \sin^2 \frac{n\pi k}{n+1} \\
& = \frac{1 - \cos \frac{2\pi k}{n+1}}{2} + \frac{1 - \cos \frac{4\pi k}{n+1}}{2} + \dots + \frac{1 - \cos \frac{2n\pi k}{n+1}}{2} \\
& = \frac{n}{2} - \frac{1}{2} (\cos \frac{2\pi k}{n+1} + \cos \frac{4\pi k}{n+1} + \dots + \cos \frac{2n\pi k}{n+1}) \\
& = \frac{n}{2} - \frac{1}{2} (\Re(e^{i\frac{2\pi k}{n+1}} + e^{i\frac{4\pi k}{n+1}} + \dots + e^{i\frac{2n\pi k}{n+1}})) \\
& = \frac{n+1}{2}
\end{aligned}$$

所以特征向量需要引入正交化因子 $\sqrt{2/(n+1)}$ 。

根据上述讨论，我们得到 T 的特征值是 $\lambda = 2c = 2 \cos(k\pi/(n+1))$ ，特征向量

$$\mathbf{v}_k = \sqrt{\frac{2}{n+1}} (\sin \frac{\pi k}{n+1}, \sin \frac{2\pi k}{n+1}, \dots, \sin \frac{n\pi k}{n+1})^\top$$

再根据 K 和 T 的关系，得到 K 的特征值 $\mu = 2 - \lambda = 2 - 2 \cos(k\pi/(n+1)) = (2 \sin \frac{k\pi}{2(n+1)})^2 = \omega_k^2$ ，特征向量 $\mathbf{v}_k = \sqrt{\frac{2}{n+1}} (\sin \frac{\pi k}{n+1}, \sin \frac{2\pi k}{n+1}, \dots, \sin \frac{n\pi k}{n+1})^\top$
于是

$$\mathbf{K} = (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n)^\top \text{diag}\{\omega_1^2, \omega_2^2, \dots, \omega_n^2\} (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n) = \mathbf{P}^\top \mathbf{D} \mathbf{P}$$

其中 P 是正交矩阵。容易知道 $P[i; j] = \sqrt{\frac{2}{n+1}} \sin \frac{\pi i j}{n+1}$ ，i、j 对称，P 还是对称矩阵。

设

$$\mathbf{Q} = \mathbf{P} \mathbf{q} \tag{6}$$

体系哈密顿量可以改写为

$$\begin{aligned}
\mathbf{H} &= \frac{1}{2} \dot{\mathbf{q}}^\top \dot{\mathbf{q}} + \frac{1}{2} \mathbf{q}^\top \mathbf{K} \mathbf{q} = \frac{1}{2} \dot{\mathbf{q}}^\top \mathbf{P}^\top \mathbf{P} \dot{\mathbf{q}} + \frac{1}{2} \mathbf{q}^\top \mathbf{P}^\top \mathbf{D} \mathbf{P} \mathbf{q} \\
&= \frac{1}{2} \dot{\mathbf{Q}}^\top \dot{\mathbf{Q}} + \frac{1}{2} \mathbf{Q}^\top \mathbf{D} \mathbf{Q} \\
&= \frac{1}{2} (Q_1, Q_2, \dots, Q_n) \begin{pmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_n \end{pmatrix} + \frac{1}{2} (Q_1, Q_2, \dots, Q_n) \begin{pmatrix} \omega_1^2 & & & \\ & \omega_2^2 & & \\ & & \ddots & \\ & & & \omega_n^2 \end{pmatrix} \begin{pmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_n \end{pmatrix}
\end{aligned}$$

于是就得到了体系 n 个本征模式的能量 $E_k = \frac{1}{2} \dot{Q}_k^2 + \frac{1}{2} \omega_k^2 Q_k^2$ 其中 $Q_k = \sqrt{\frac{2}{n+1}} \sum_{j=1}^n \sin \frac{\pi k j}{n+1} q_j$, $\omega_k = 2 \sin \frac{\pi k}{2(n+1)}$

1.2 第二问

这一问本质上需要解决两个问题：

问题一：根据各个简正模式的初始位置确定各个质点在真实相空间的初始位置；

问题二：求解系统随时间的演化，并计算前四个简正模式的能量。

1.2.1 思路分析

问题一：由于质点的真实位形和简正模式位形通过一个正交变换联系（方程 1.1），给定简正模式位形可以求出真实位形，这只需解一个线性方程组即可。由于 P 是正交矩阵，是满秩的，于是可以采用列主元的 Doolittle 分解法求解。

问题二：这是个哈密顿系统，为保证能量守恒，应采用辛算法求解系统随时间的演化。由于蛙跳法是辛的，这里可以采用蛙跳法进行。

具体代码实现见附录”q2.c”。

这里展示 E_1, E_2, E_3, E_4 随时间的变化，时间轴单位取 $2\pi/\omega_1$ 。（原始数据见“q2.txt”）

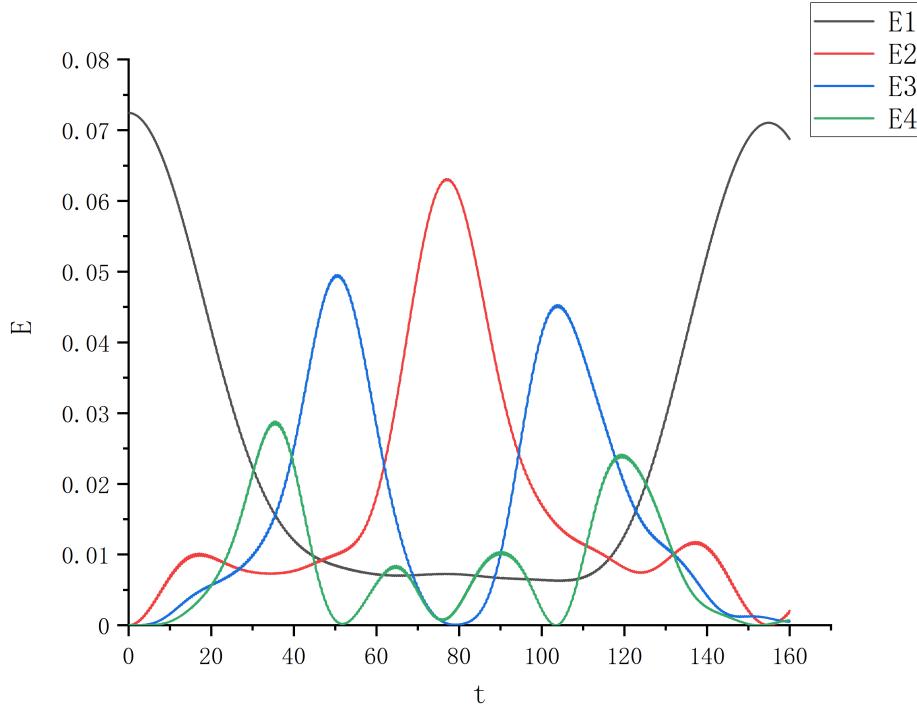


图 1: 第二问图

回归时刻 $t = 154.5$, 能量比 $\eta = \frac{0.071035}{0.072449} = 98\%$

1.3 第三问

我们期待看到模式 1 逐渐弥散到其他能量模式上，最后各个模式能量基本一致，实际上却看到存在这样一个周期，体系基本回到了初态，下面提出几种可能的解释。

可能 1： α 不够大，非线性效应不够显著，线性效应仍然占据主导，导致能量仍然倾向于回到初始本正能量 E_1 。

可能 2： n 不够大，每个弹簧非线性效应的耦合不够混乱，导致体系仍具有某种周期，就像极限情况 $n=1$ ，哪怕存在非线性回复力，振子依然做周期运动。

可能 3：由于非线性系统的演化极大地依赖初值，这个 Q_1 初始值可能比较偶然出现了能量暂时回归的现象。

可能 4：体系演化时间不够长，这里虽然有高达 98% 的能量回到初始状态，但是还是有 2% 弥散掉了，能让体系演化更长一段时间，会观察到能量均分的现象。

1.4 第四问

这一问与第二问求解过程几乎完全一样，只有一点点变化是体系要多演化一段时间，并且只用求每个时刻的 E_1 即可，具体代码实现见附录”q4.c”。

这里展示 E_1 随更长的时间变化的图像。（原始数据见“q4.txt”）

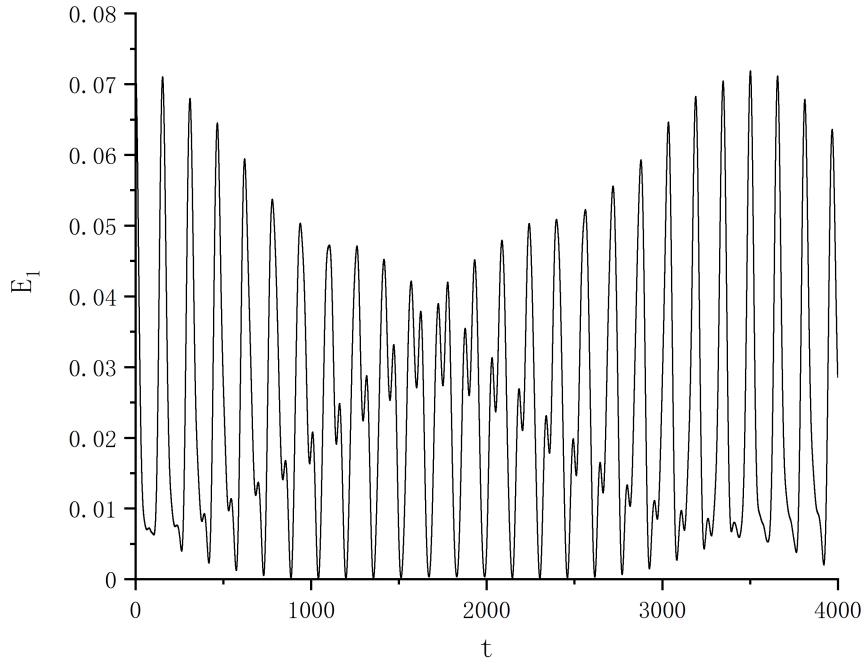


图 2: 第四问图

回归时刻 $t = 3502.18$, 能量比例 $\eta = \frac{0.071902}{0.072449} = 99\%$

1.5 第五问

本问与前问求解过程基本一致, 为探究各个模式能量随时间的变化。我们取 t 的单位仍为 $2\pi/\omega_1$, 每隔 0.01 计算一次全部 32 个模式的能量。同时, 为比较不同初始条件下体系演化形式的不同, 我们分别就 $Q_1(0) = 20$ 和 $Q_1(0) = 4$ 分别计算 t 从 0 至 4000 中所有模式能量的演化。具体代码实现见附录”q5.c”。运行结果 $Q_1(0) = 20$ 对应”q5_sparsify.txt”, $Q_1(0) = 4$ 对应”q5_cmp_sparsify.txt”。

接下来, 我们完全可以选择将数据导入 origin, 再绘制 64 幅”E-t”图, 但这样十分耗时耗力。我们选择运用 python 的 numpy 和 matplotlib 包可以实现从 txt 读取数据、画图并保存。具体代码实现见附录”plotTool.py”。

下面展示两种初始条件所有模式能量随时间的演化图。其中左侧是初态 $Q_1(0) = 4$ 的体系, 右侧是初态 $Q_1(0) = 20$ 的体系, 由上到下分别是 E_1 到 E_{32} 随时间的演化。

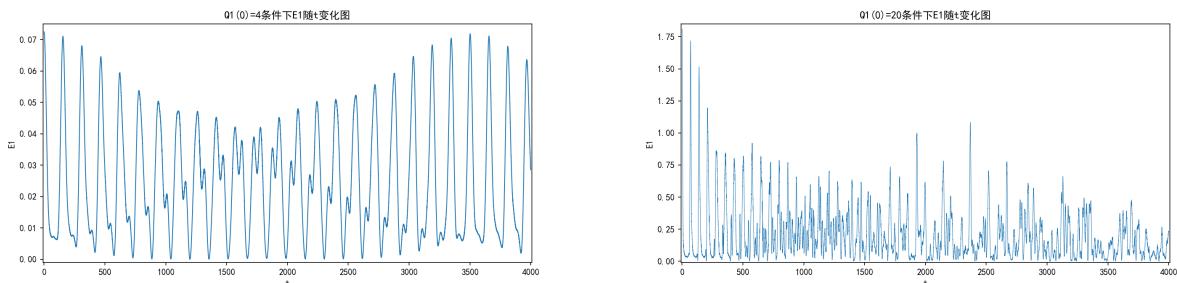


图 3: E1

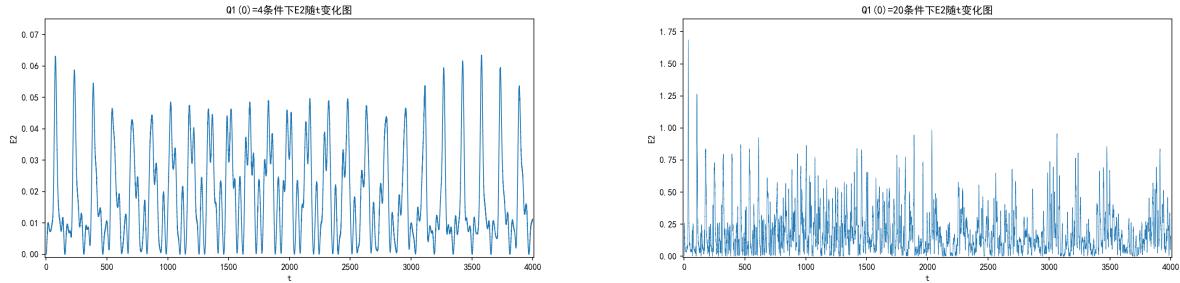


图 4: E2

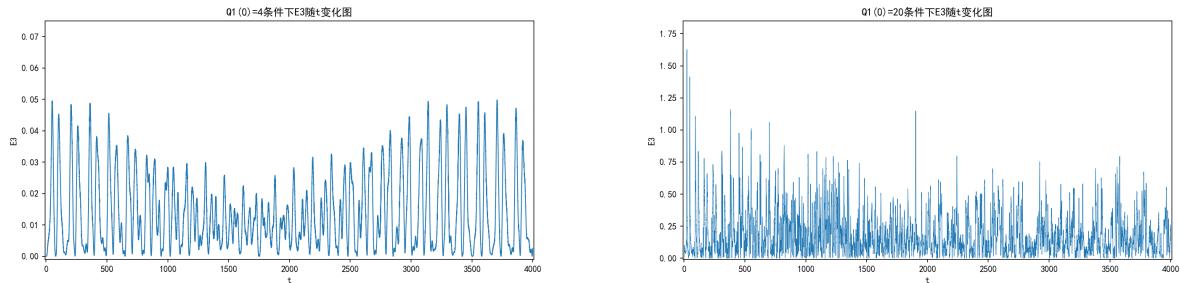


图 5: E3

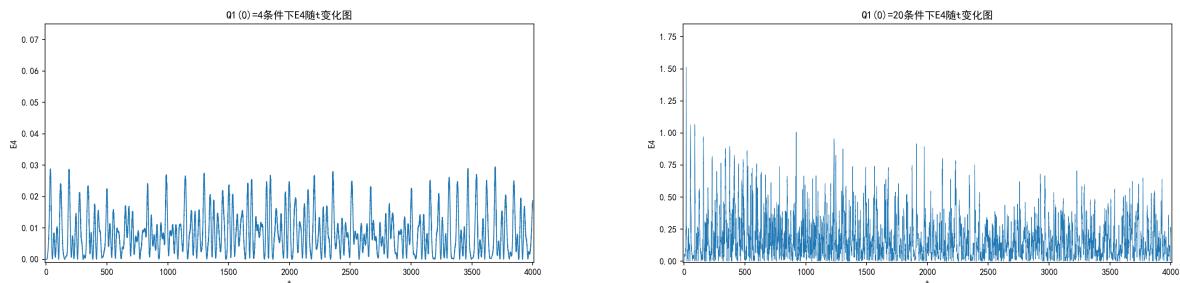


图 6: E4

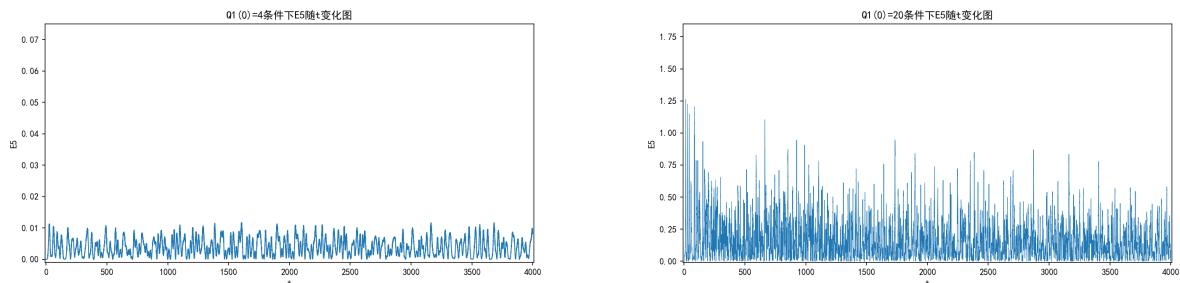


图 7: E5

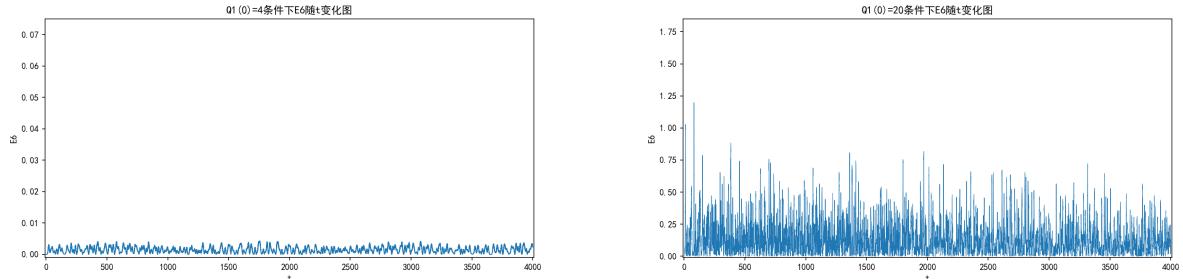


图 8: E6

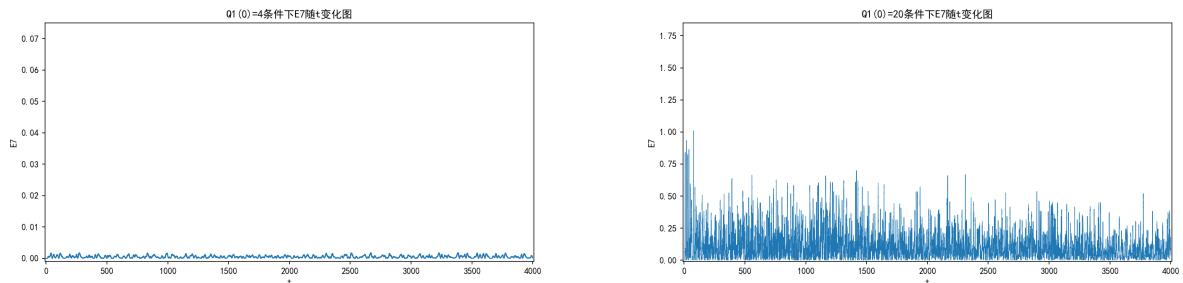


图 9: E7

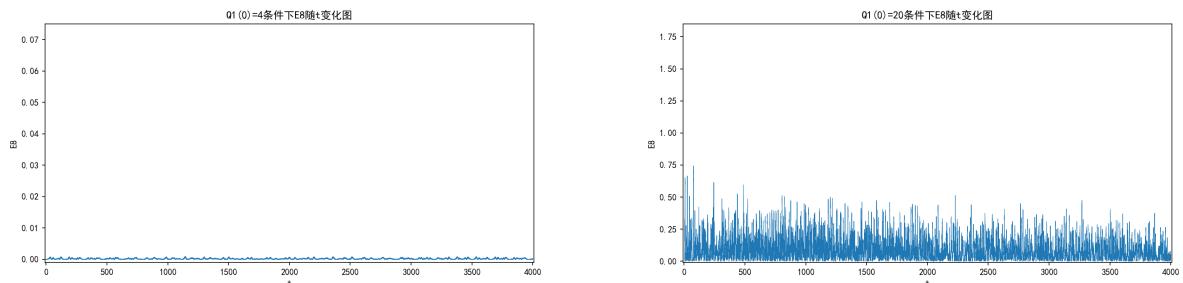


图 10: E8

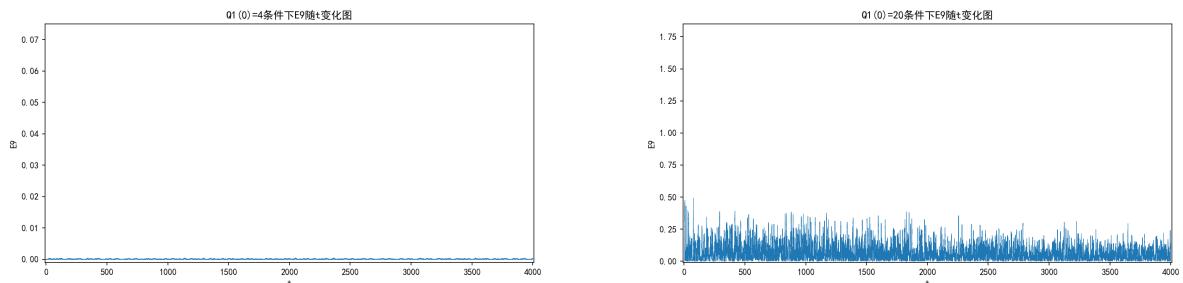


图 11: E9

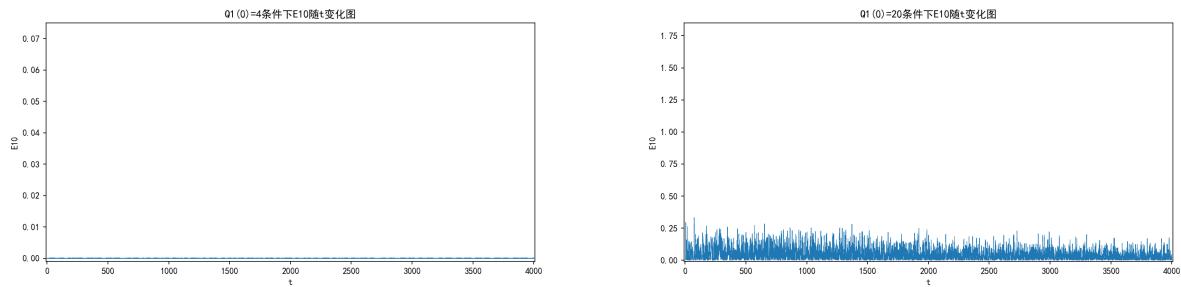


图 12: E10

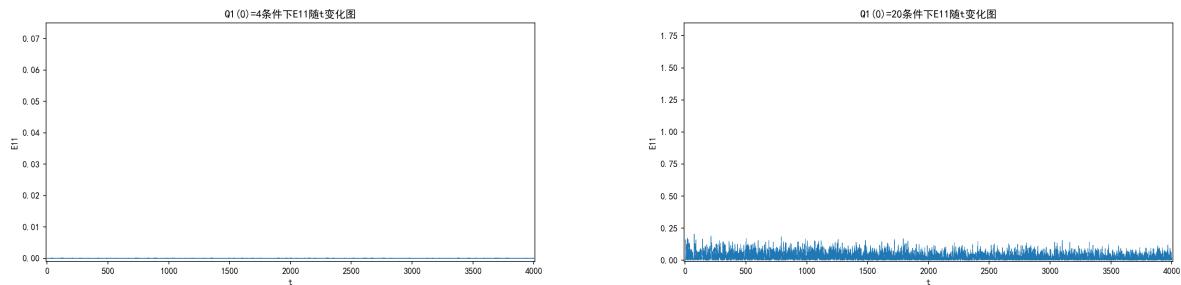


图 13: E11

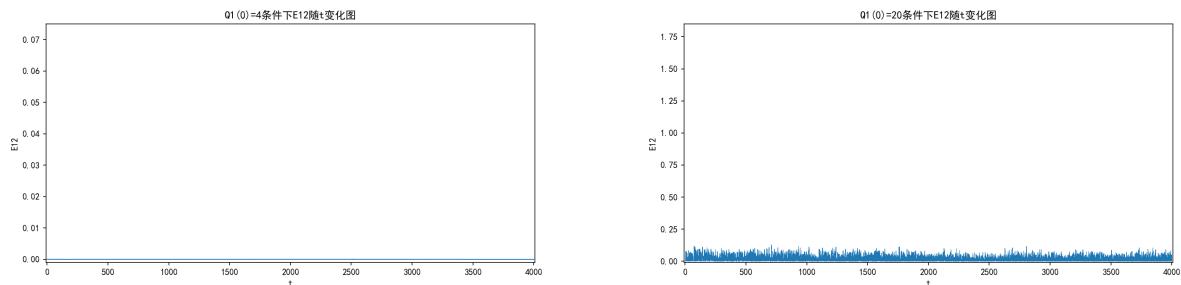


图 14: E12

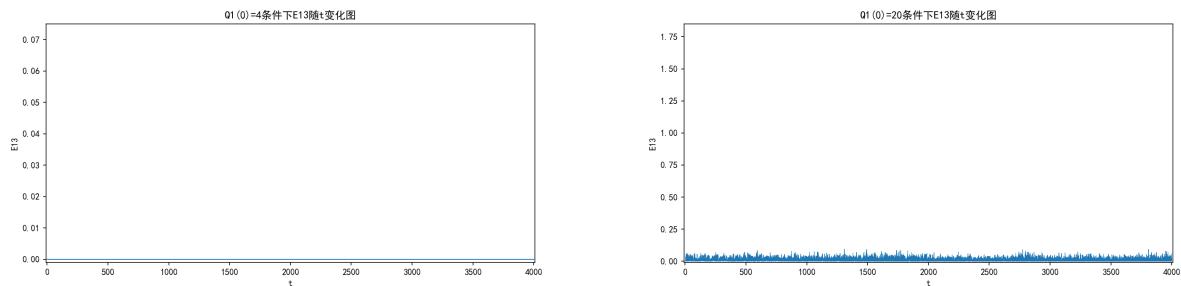


图 15: E13

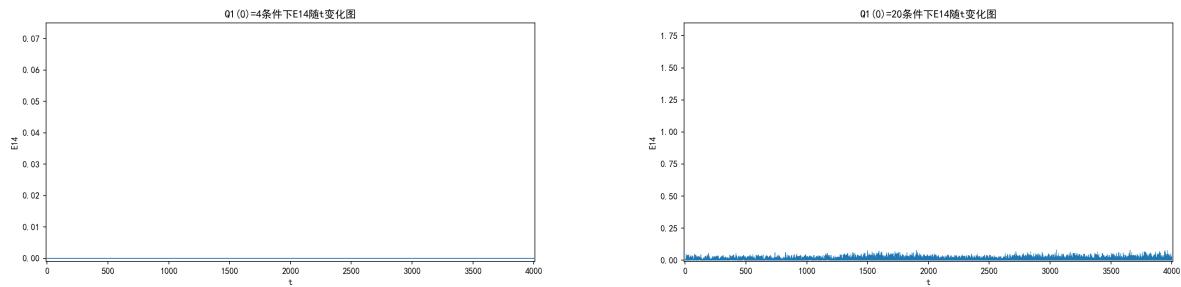


图 16: E14

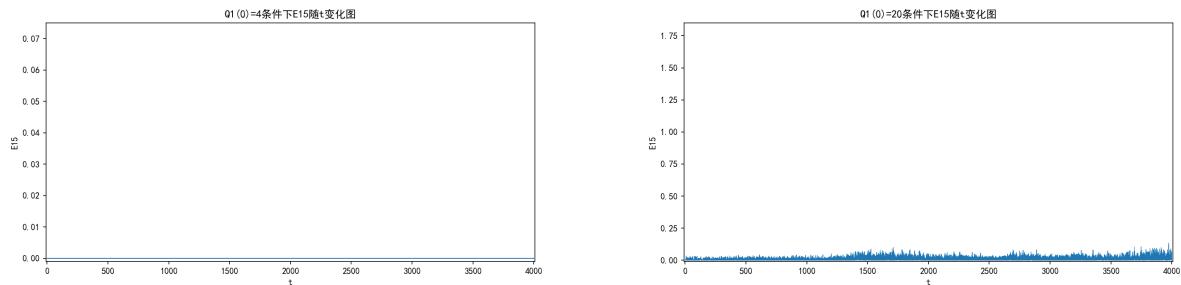


图 17: E15

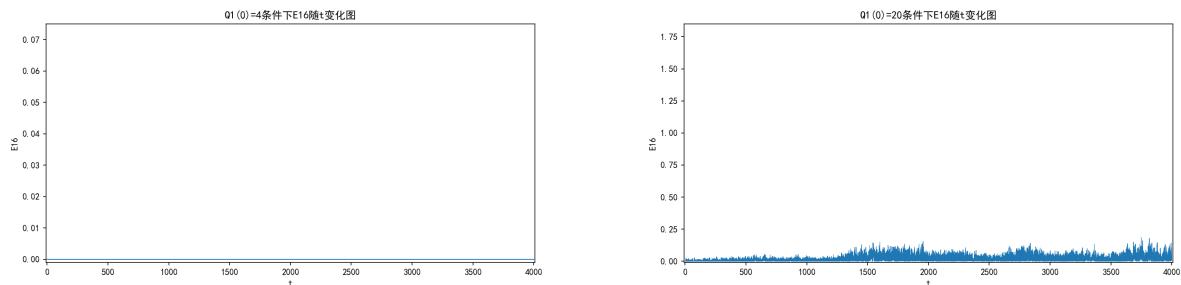


图 18: E16

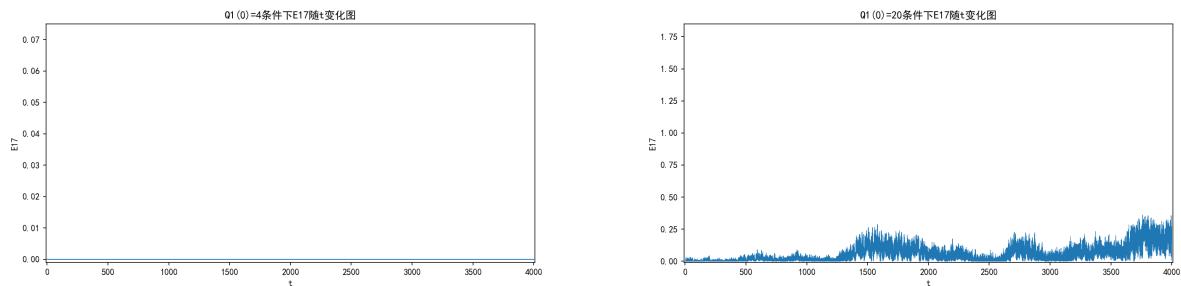


图 19: E17

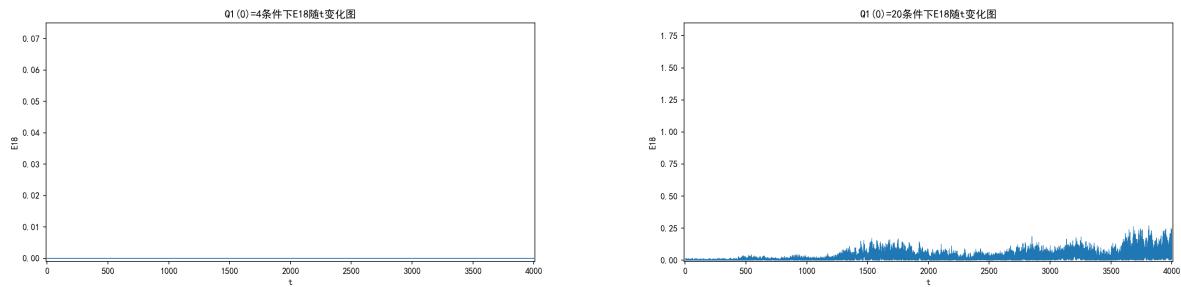


图 20: E18

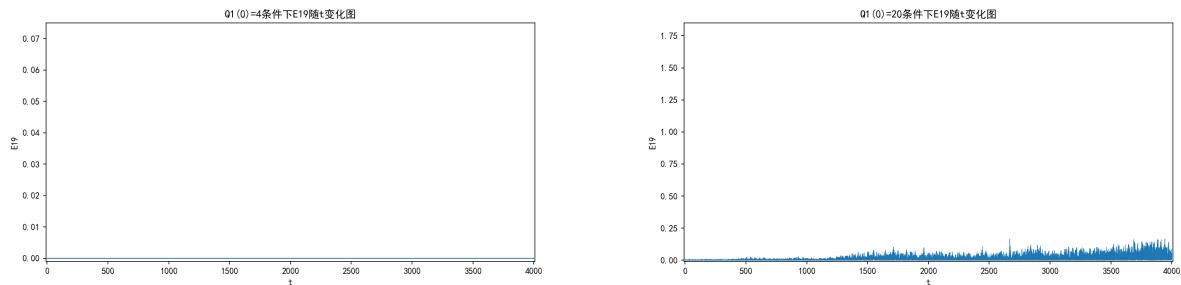


图 21: E19

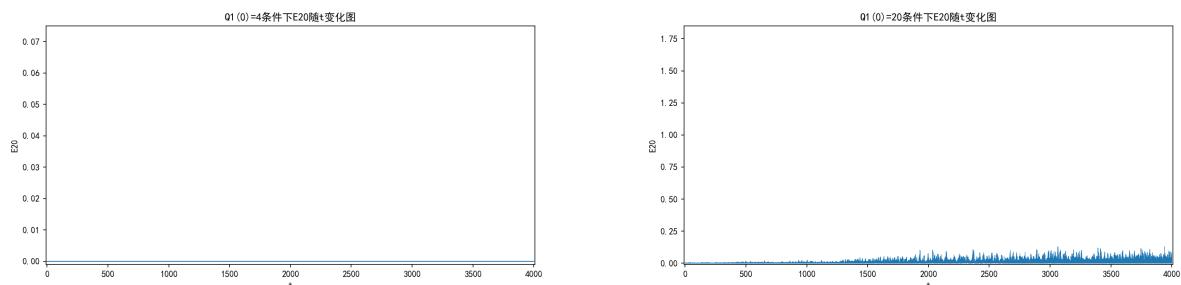


图 22: E20

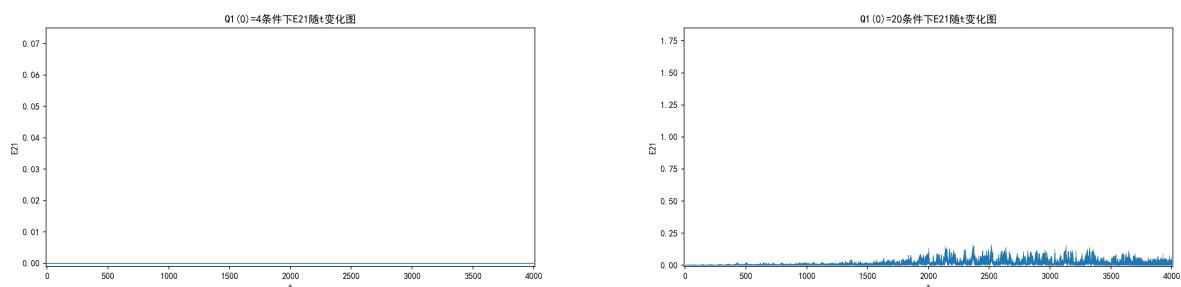


图 23: E21

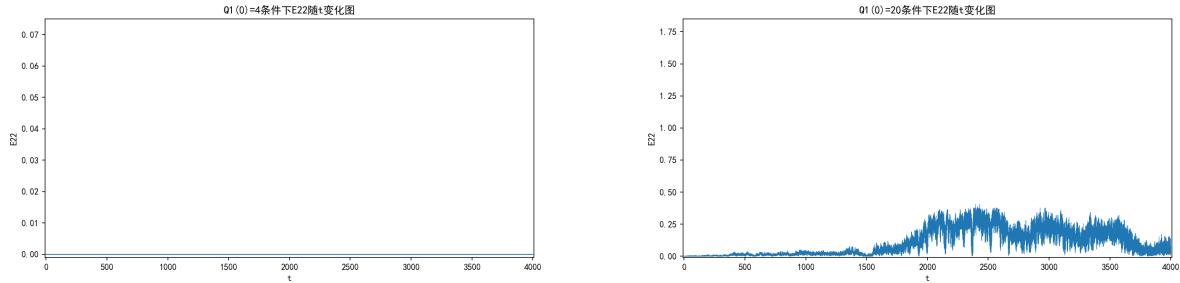


图 24: E22

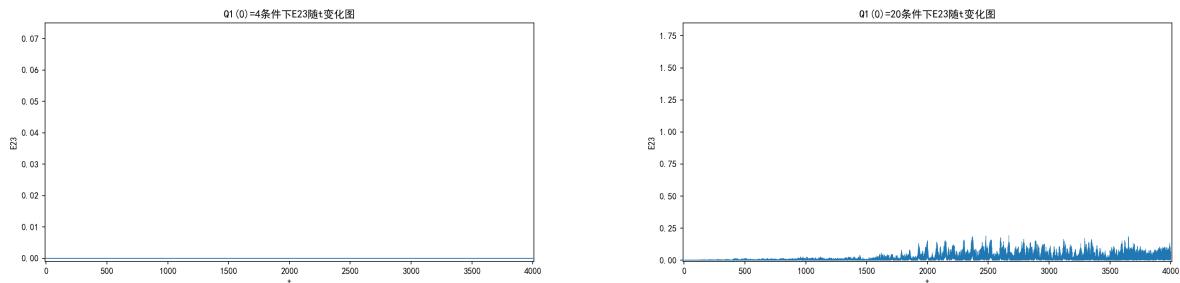


图 25: E23

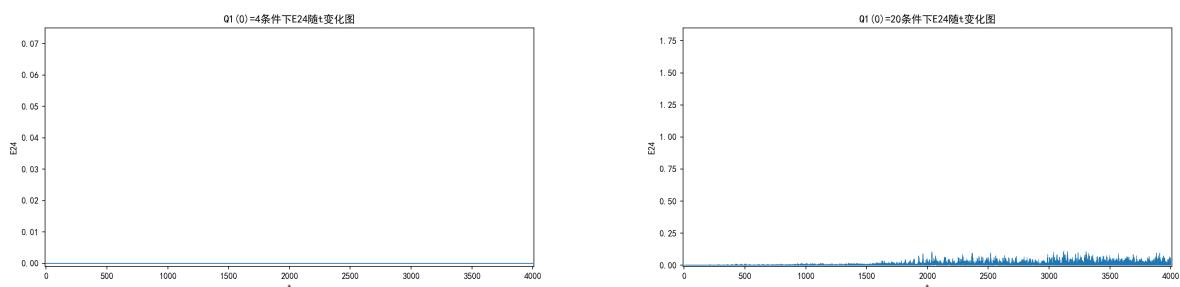


图 26: E24

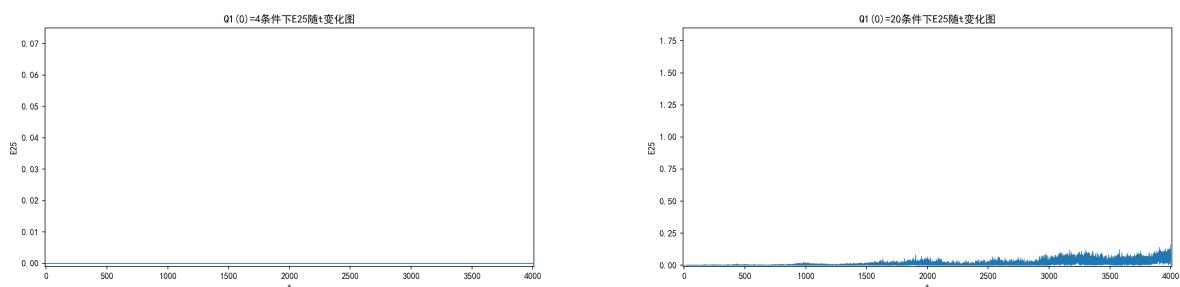


图 27: E25

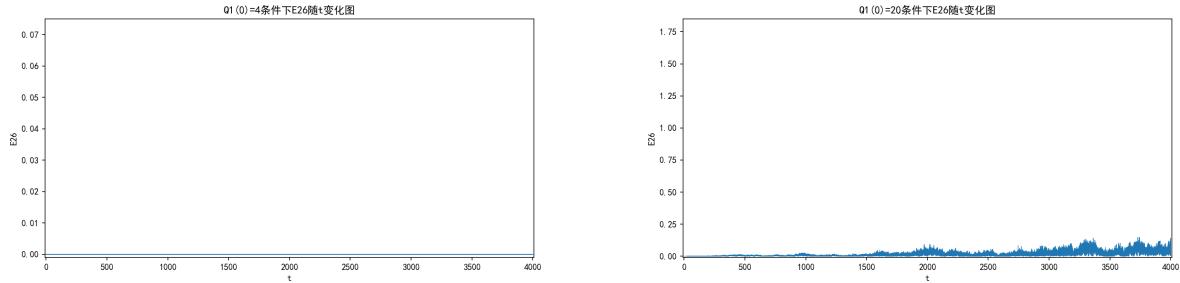


图 28: E26

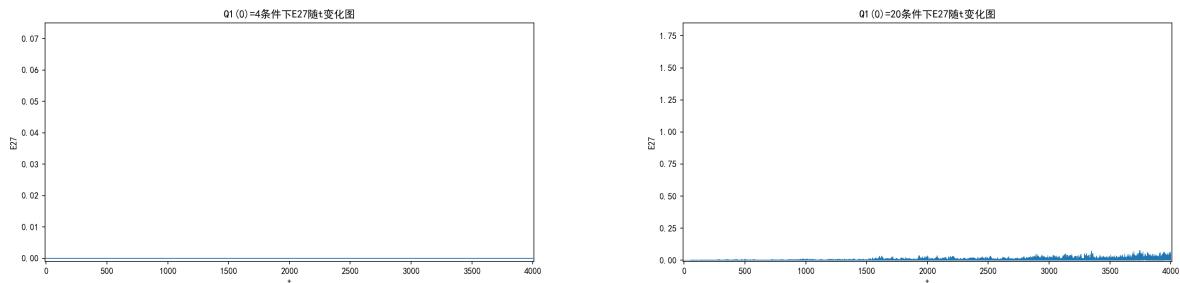


图 29: E27

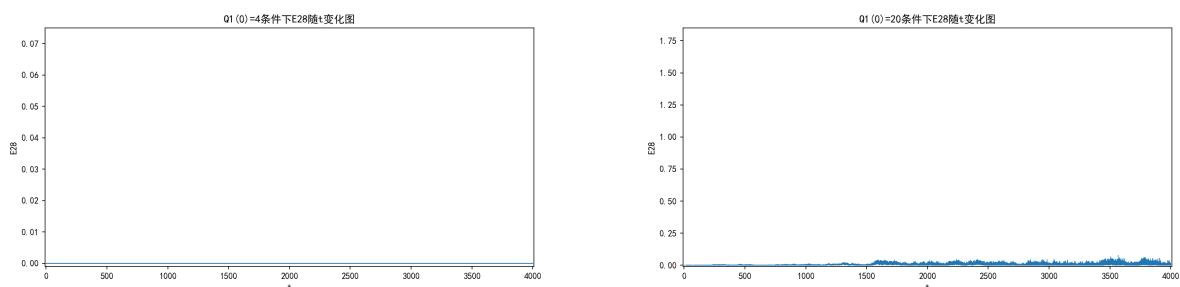


图 30: E28

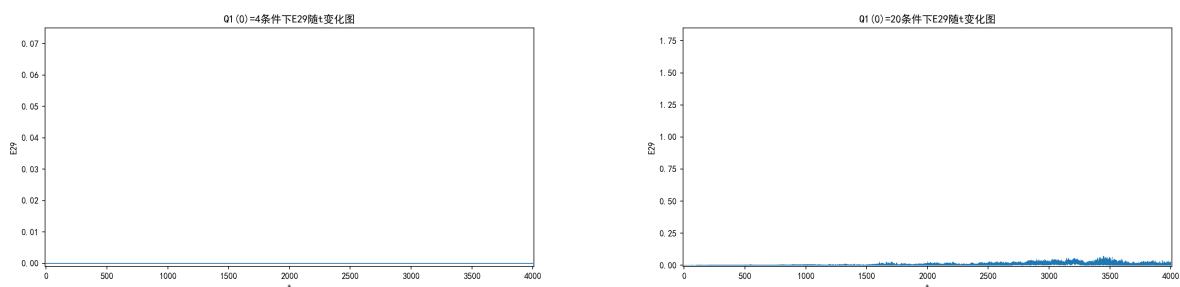


图 31: E29

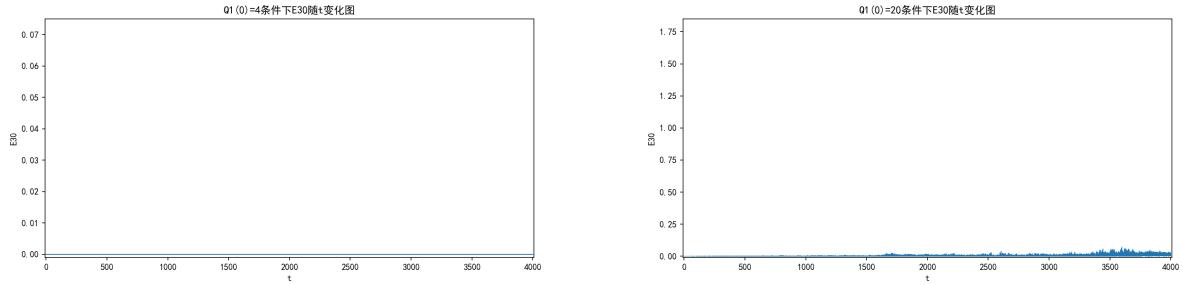


图 32: E30

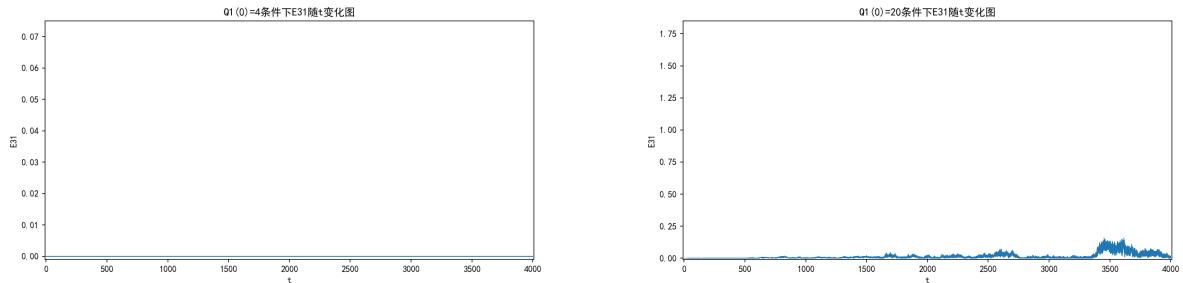


图 33: E31

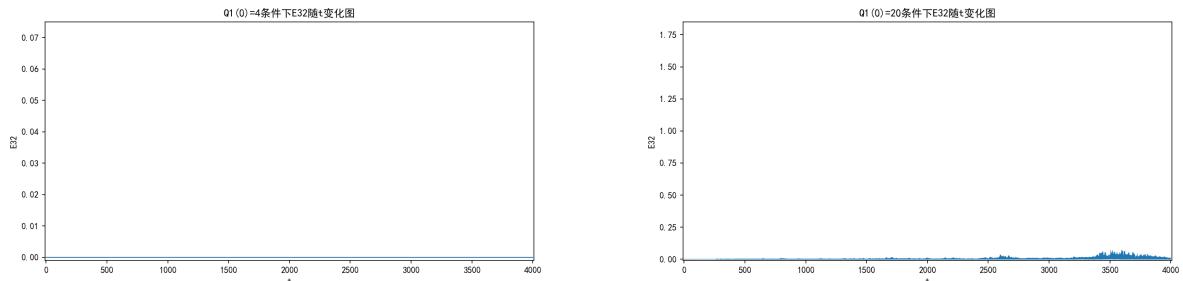


图 34: E32

这里每一类初始条件各个模式均选取相同的纵坐标显示范围，可以直观比较出各个模式的强度。观察 E_1 至 E_4 ，左侧图具有明显回归周期至 3500，但右侧图中，保守的说，自 $t=500$ 开始，系统就偏离了左侧的演化规律，呈现混乱状态。再观察其他能量右侧图， $t=500$ 之后，体系没有明显周期性规律可言。因此体系在大约 $t=500$ 之后表现出混乱特征。

同时，自上而下观察右侧图，可以看出初态 $Q_1(0) = 20$ 条件下， $\langle E_k \rangle$ 随 k 的增大而减小。除了 E_{17}, E_{18}, E_{22} 部分区间有突然的增大之外，整体上这种单调关系是成立的。

进一步的，如果对比左右两侧的图，可以发现初态 $Q_1(0) = 4$ 条件下基本没有能量转移到 E_9 之后的模式上，而初态 $Q_1(0) = 20$ 条件下能量在所有模式上都有转移，这和初态 $Q_1(0) = 20$ 条件得到的体系混乱程度更高是一致的。

1.6 第六问

这里只需要将第 2、4、5 问中的代码与非齐次项相关的地方作出修改即可。

这里取 $n=32$, $\beta = 1$, 初始条件 $Q_1(0) = 10$ 其他都为 0 为例。

1.6.1 类似第二问的结果

代码见附录”q6_to_q2.c”，所得原始数据见”q6_to_q2.txt”

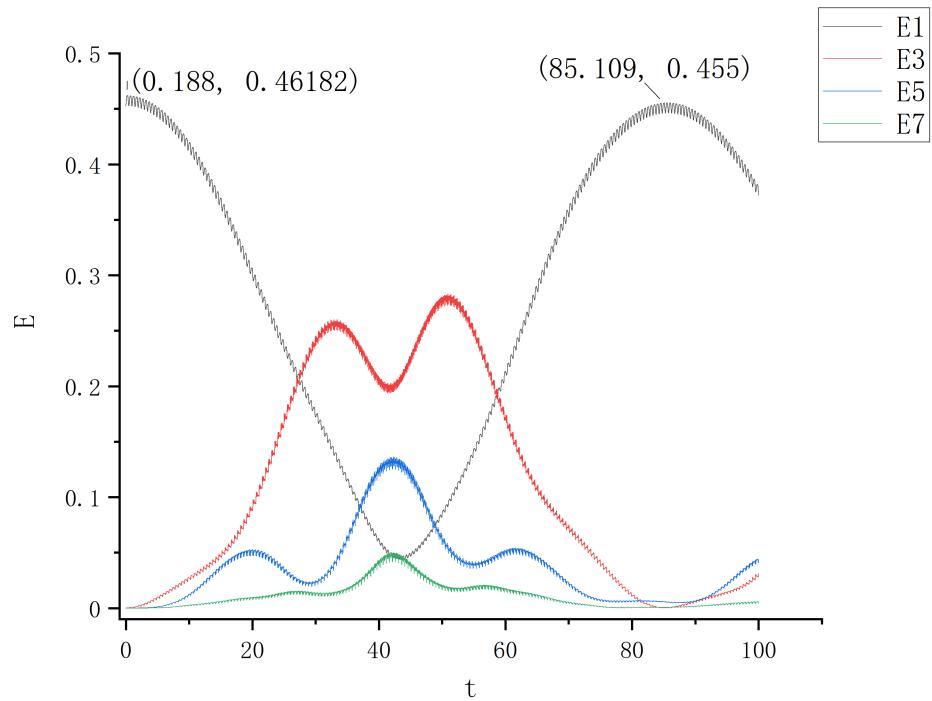


图 35: 第六问联系第二问图

可以看出一个回归周期 $t=85$, 能量比例 $\eta = 0.455/0.46182 = 98.5\%$ 。特别注意考虑到偶数模式为零, 这里画出的是 E_1, E_3, E_5, E_7 随时间的变化。

1.6.2 类似第四问的结果

代码见附录”q6_to_q4.c”，所得原始数据见”q6_to_q4.txt”

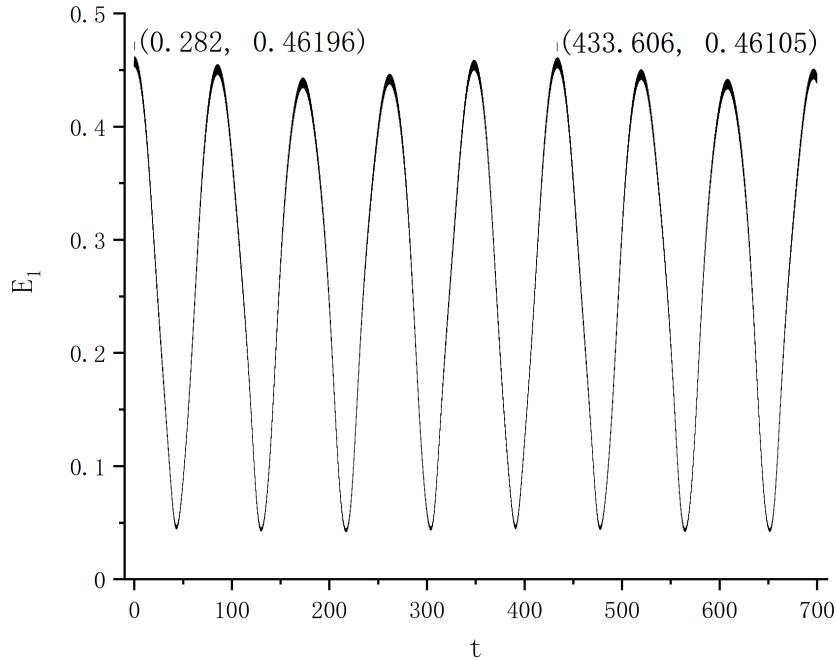


图 36: 第六问联系第四问图

可以看出一个回归周期 $t=433$, 能量比例 $\eta = 0.46105/0.46196 = 99.8\%$ 。与第四问类似, 能量回归率大于小周期处。

1.6.3 类似第五问结果

重置 $Q_1(0) = 100$, 将 $Q_1(0) = 10$ 和 $Q_1(0) = 100$ 各个模式能量在 t 从 0-700 (单位 $2\pi/\omega_1$) 的演化做对比, 得到下图。代码见”q6_tp_q5.c”, 原始数据见”q6_tp_q5.txt”, 画图方法同第五问。

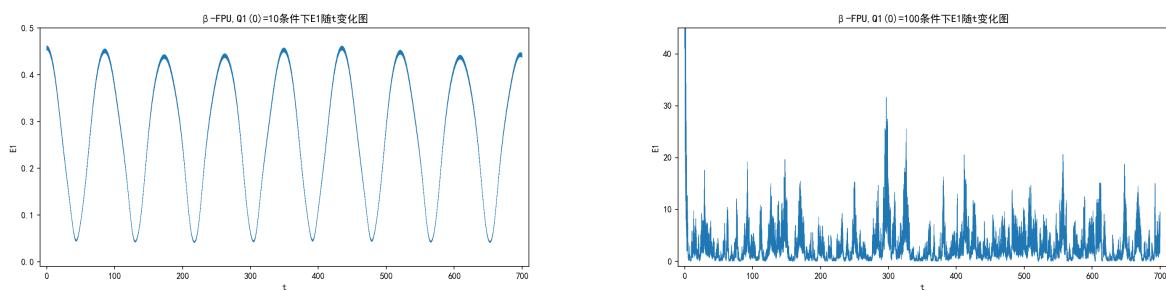


图 37: E1

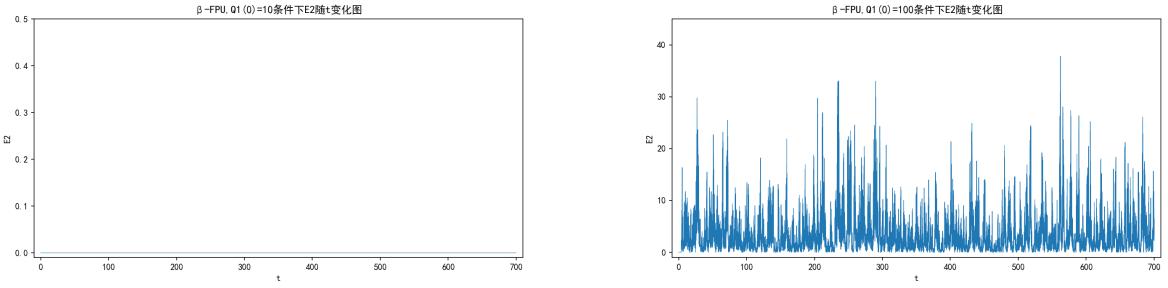


图 38: E2

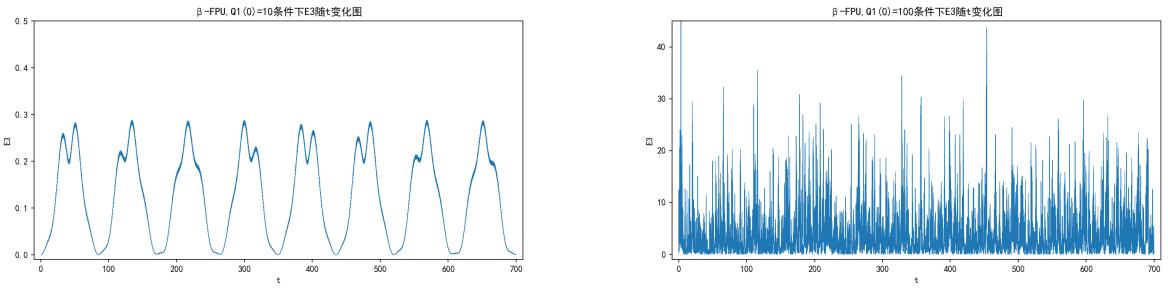


图 39: E3

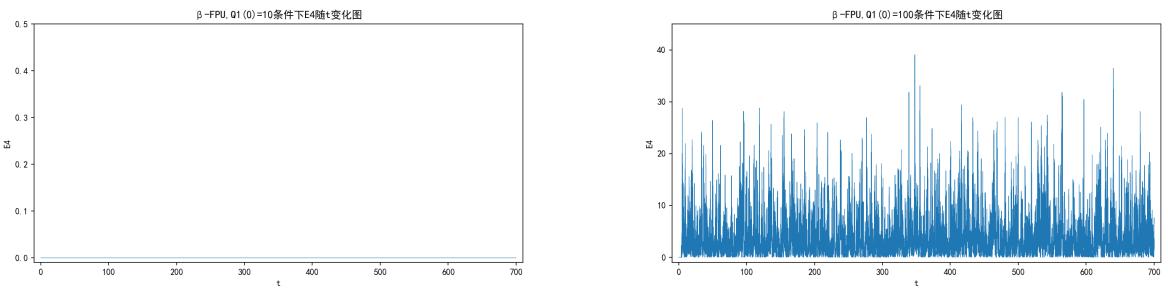


图 40: E4

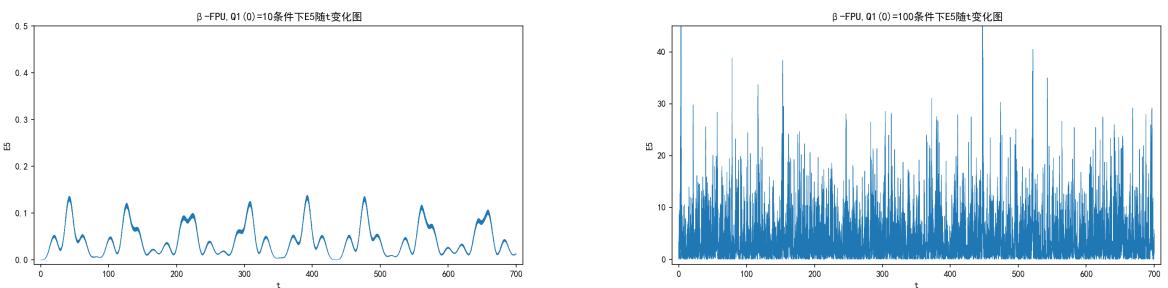


图 41: E5

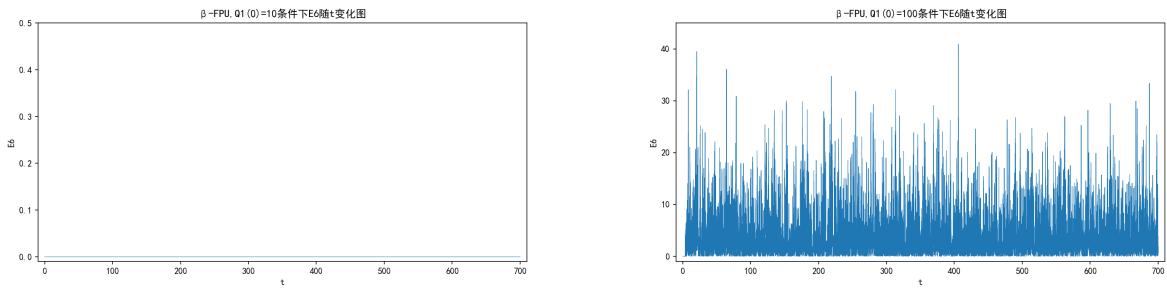


图 42: E6

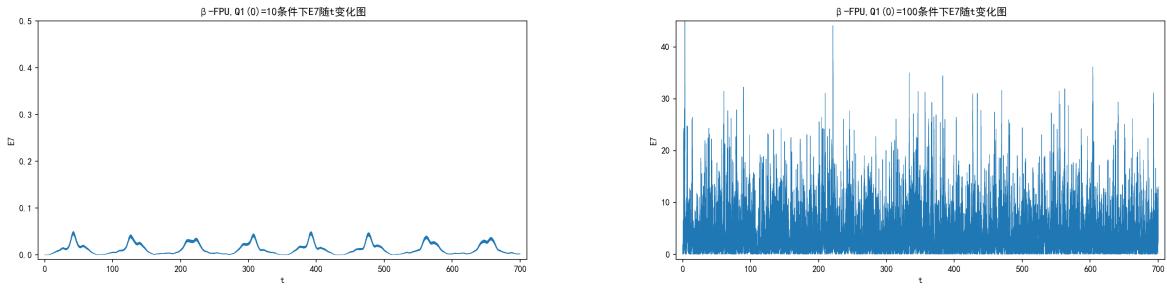


图 43: E7

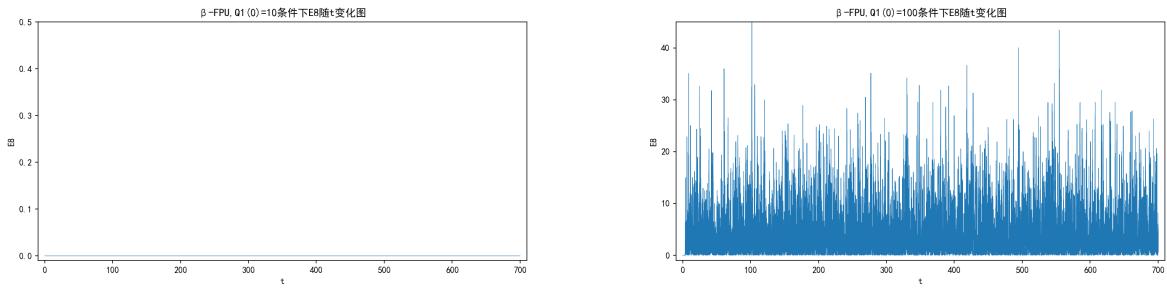


图 44: E8

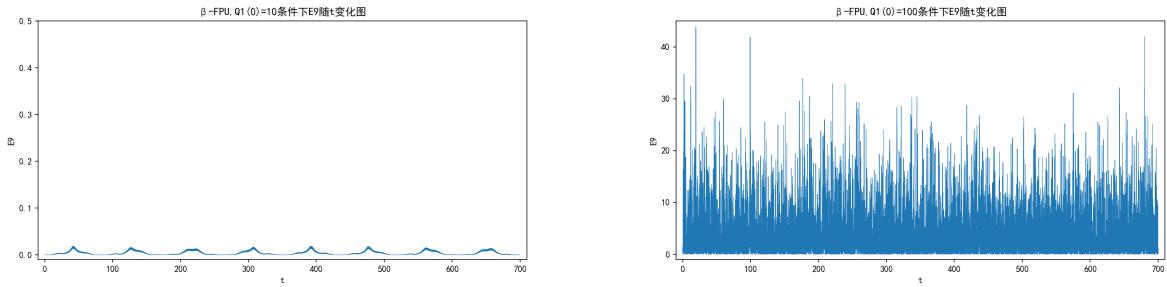


图 45: E9

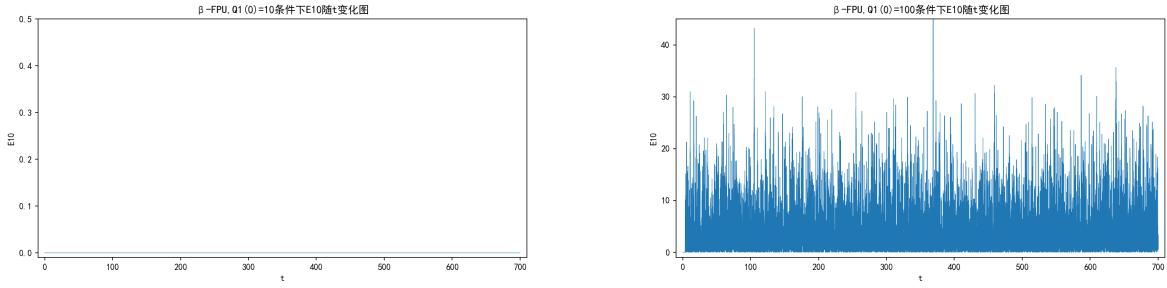


图 46: E10

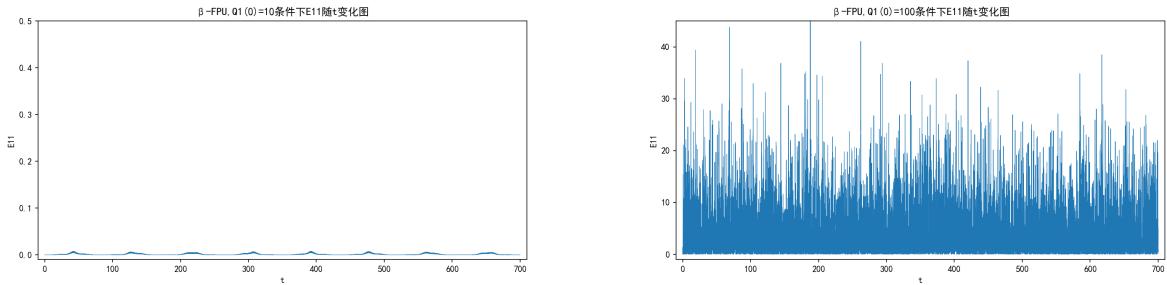


图 47: E11

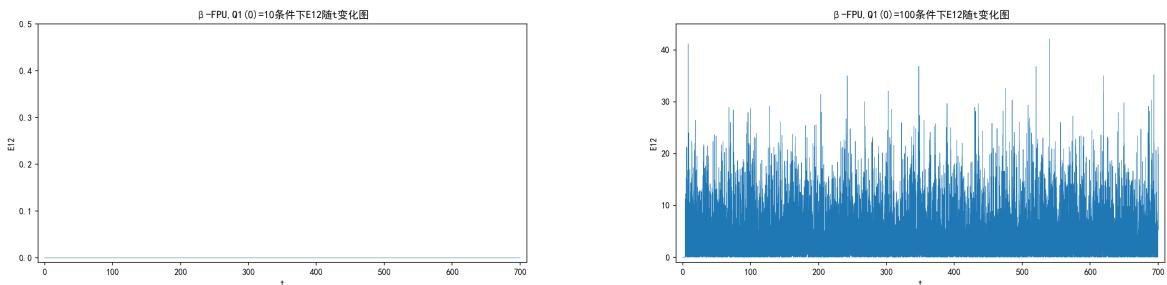


图 48: E12

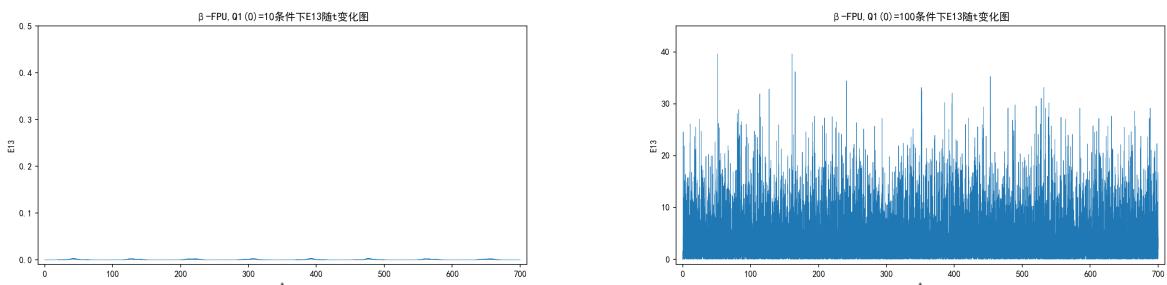


图 49: E13

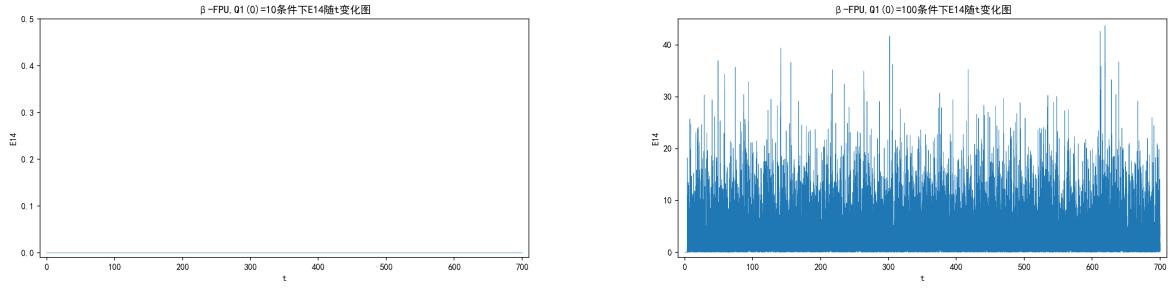


图 50: E14

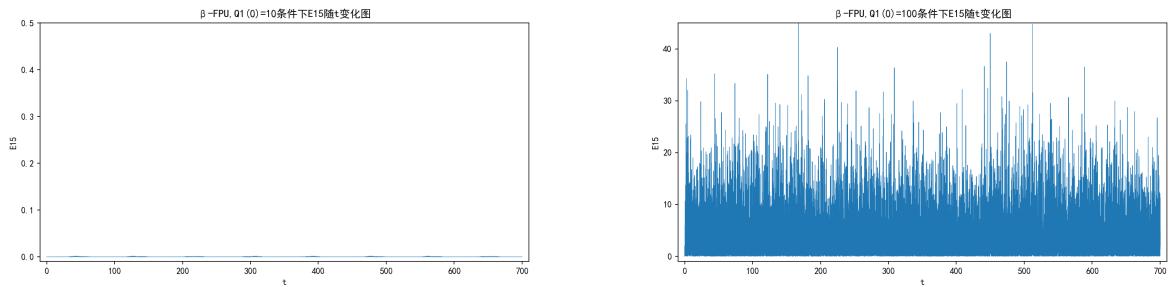


图 51: E15

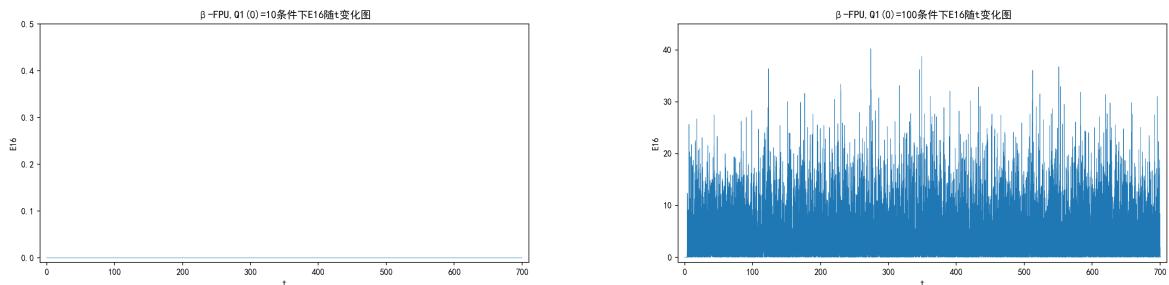


图 52: E16

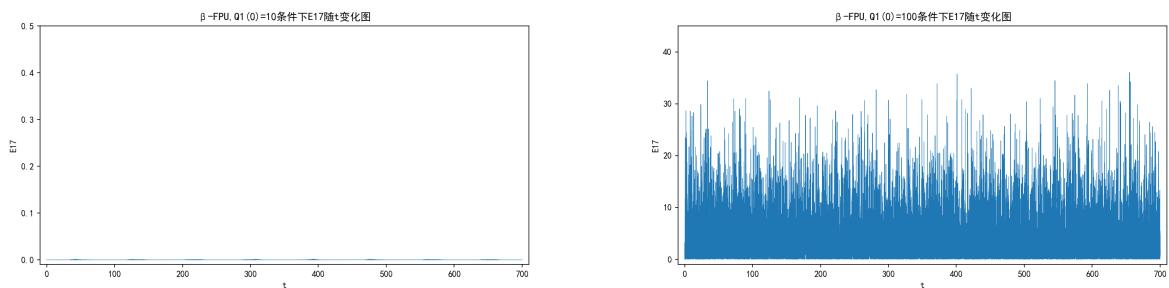


图 53: E17

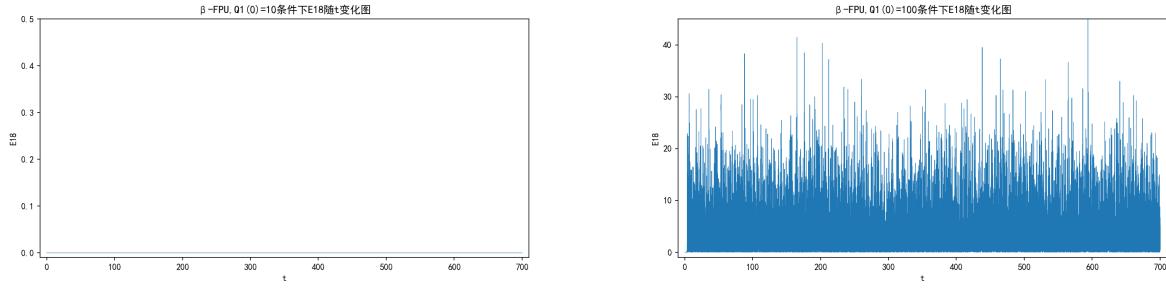


图 54: E18

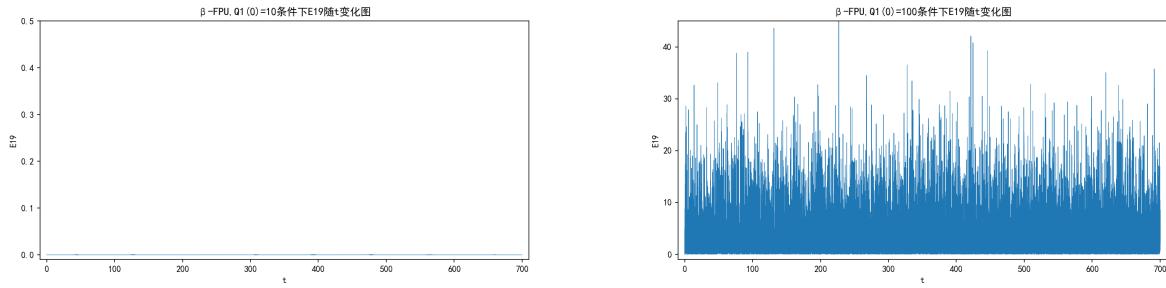


图 55: E19

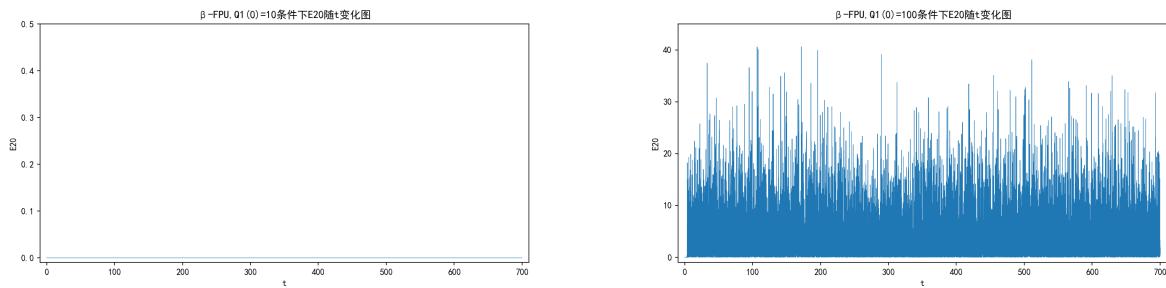


图 56: E20

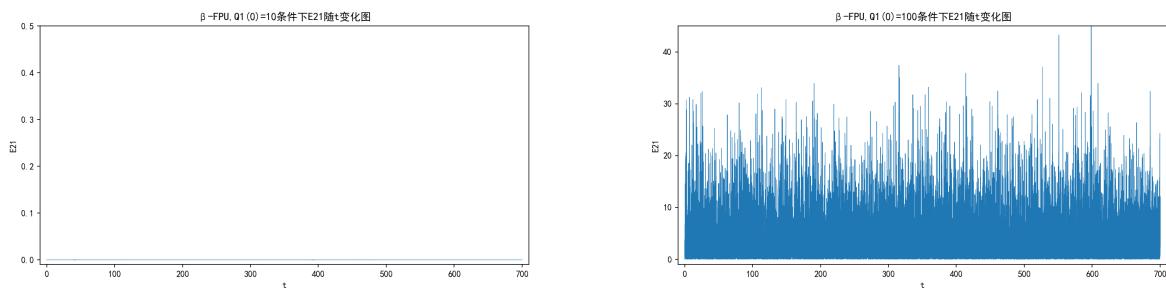


图 57: E21

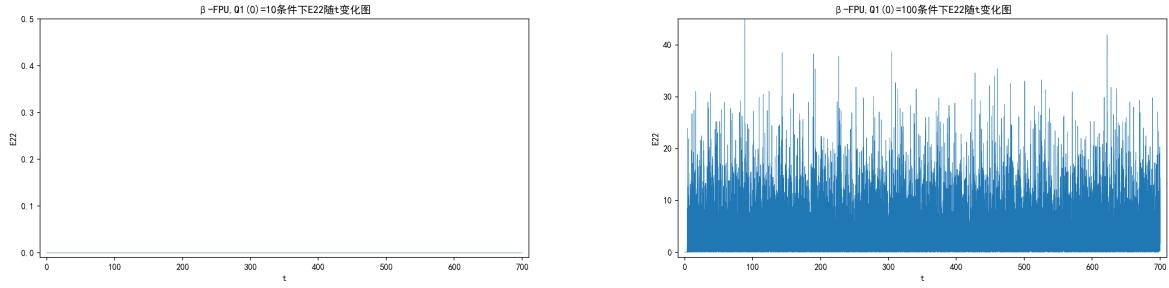


图 58: E22

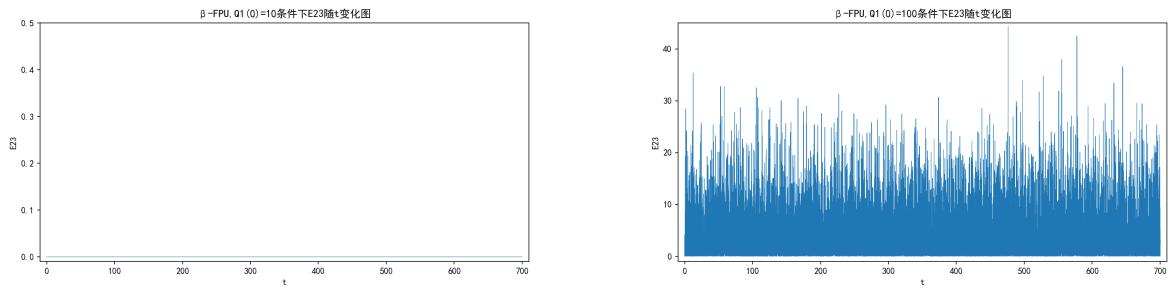


图 59: E23

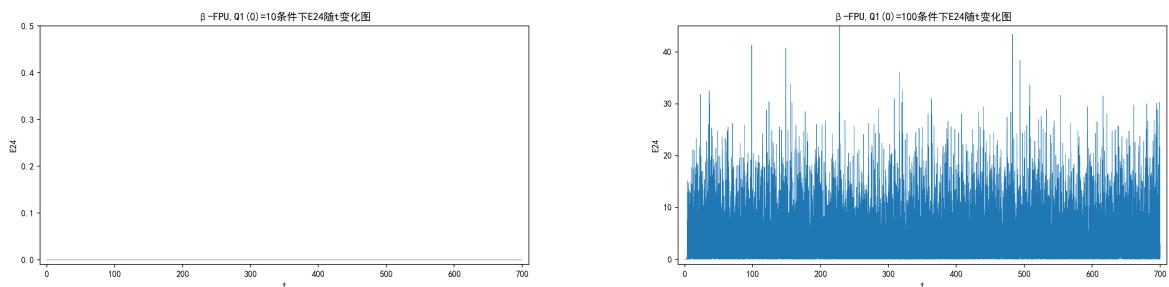


图 60: E24

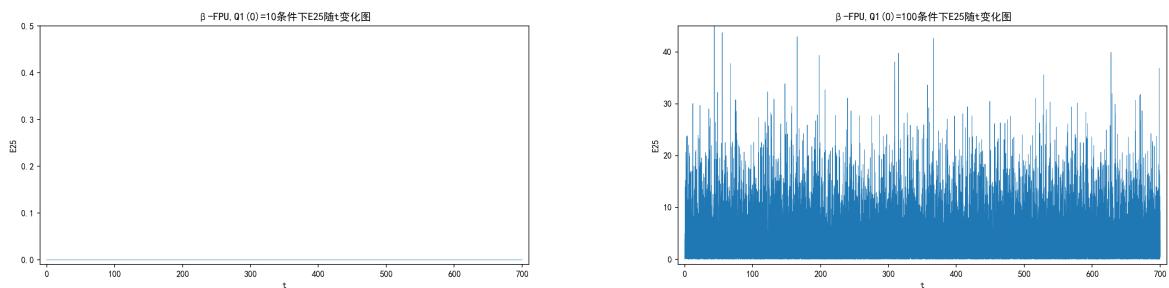


图 61: E25

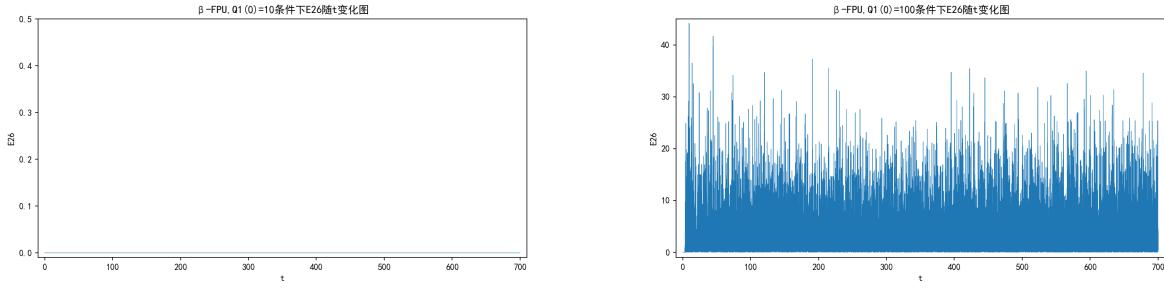


图 62: E26

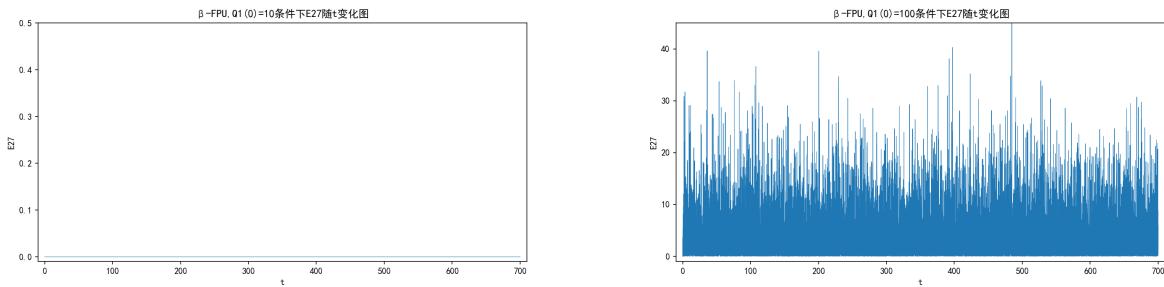


图 63: E27

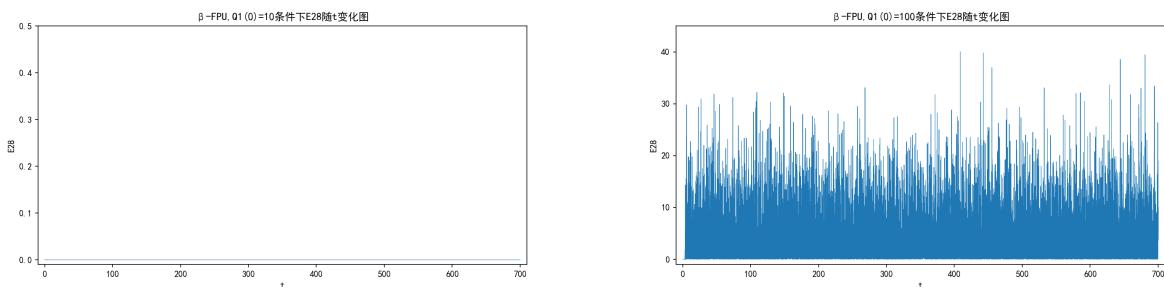


图 64: E28

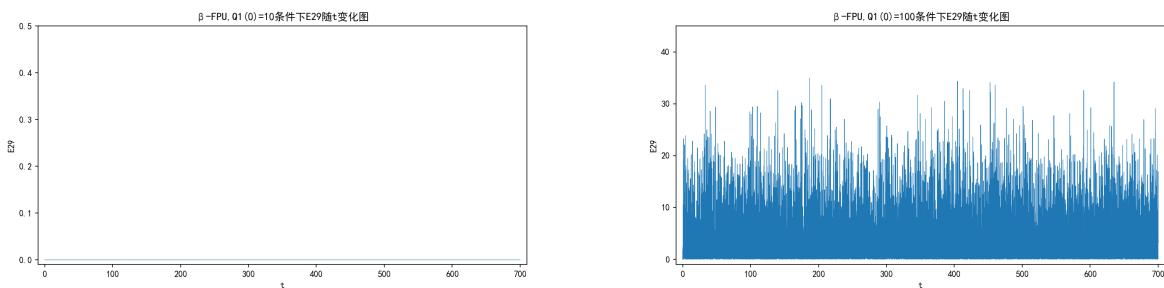


图 65: E29

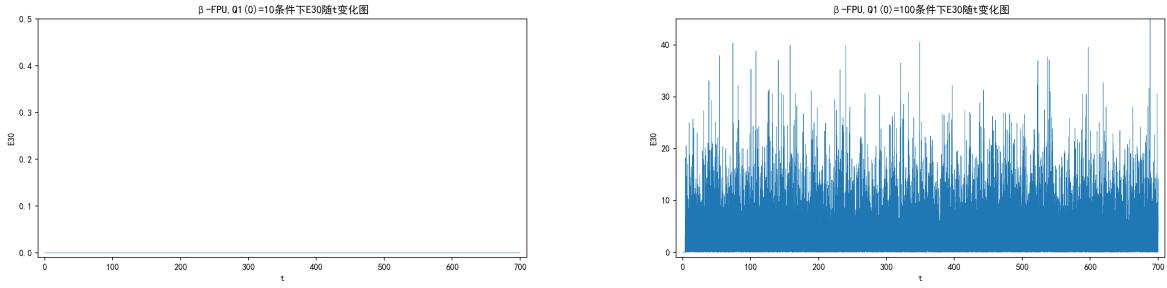


图 66: E30

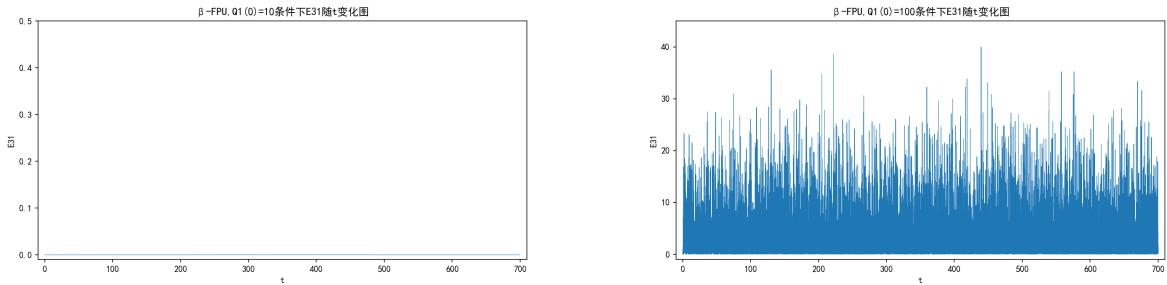


图 67: E31

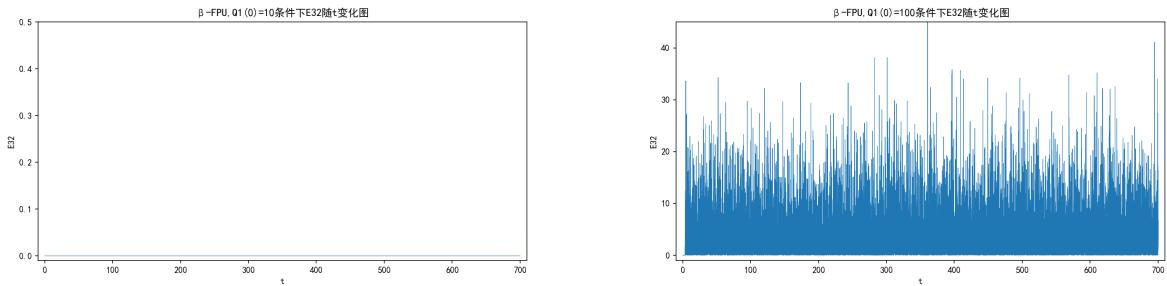


图 68: E32

可以看出改变初态后，体系周期性被显著打破，体现出混沌的特性。

1.7 第七问

由前问-类似第五问结果中的左侧图可以看出，当初始模式能量集中在模式 1 时，偶数次模式能量始终为零，但是如果改变初条件如前问右侧图所示，这种奇偶锁定就破坏了，这与下面的计算得到的结论相仿。

本问计算与前问相仿，只需改变参数 n 即可，代码见”q7.c”，运行结果见”q7.txt”（注意这里要用科学计数法输出数据）。

我们还取 t 的单位为 $2\pi/\omega_1$ ，得到下图。

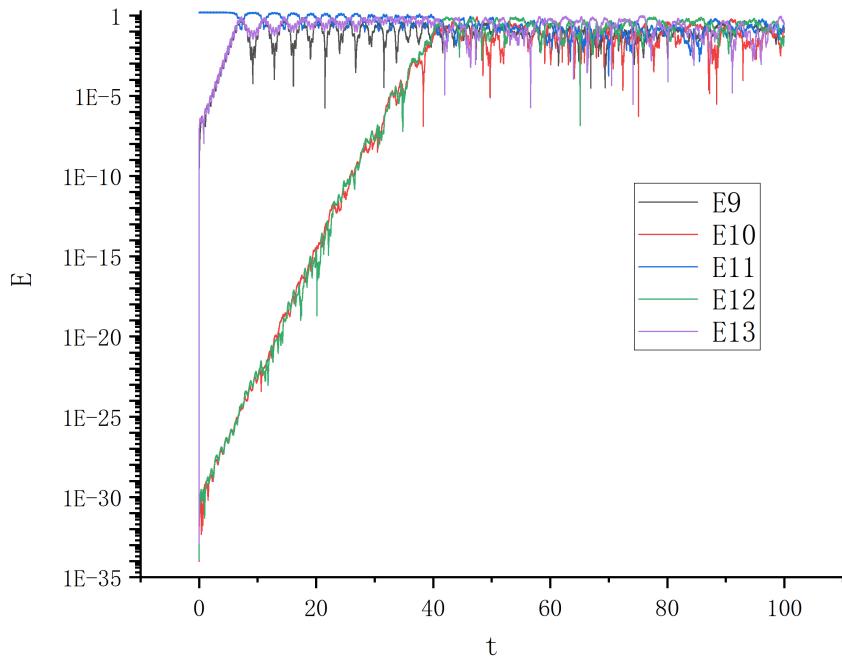


图 69: 第七问图

可以看出，随着时间演化，偶数次模式的能量的计算结果不断上升，最终与奇数次模式能量的量级相当，达到混乱状态。可能是由于舍入误差和蛙跳法的精度带来的误差影响下，偶数次模式能量不断获得一些噪声并逐渐积累变大导致。

1.8 第八问

这里时间仍取 $2\pi/\omega_1$ 为单位，可以每隔 0.1 时间打出一个所有质点的位移，总共打 200 份时间。代码见”q8.c”，原始数据见”q8_beta=0.txt”($\beta = 0$)，”q8_beta=1.txt”($\beta = 1$)。

这里可以对每个时刻，以位移为纵坐标、质点编号为横坐标画图，进而可以观察波的演化，应用”q8_beta=0.txt””q8_beta=1.txt” 我们已经可以得到所有的波随时间演化的信息了。为方便绘图，我们同样借助 python，源代码见附录”q8.py”。

这样可以得到每个时刻波的位形图，应用 ScreenToGif 软件，将所有帧连成视频，得到 $\beta = 0$ 和 $\beta = 1$ 波的演化视频，见”q8_beta=0.mp4””q8_beta=1.mp4”

可以看出，当 $\beta = 0$ 时，没有非线性效应，波在传播过程中会逐渐弥散；而当 $\beta = 1$ 时，波能够维持原来的形状随时间演化！

2 附录（源代码）

以下所有 C 程序均引用了”matrix.h”包，由于这个包较长，这里不再列出，而把源文件附在文件夹中，详见”matrix.h”

2.1 ”q2.c”

```

1 #include "matrix.h"
2 #define NUM 32
3 #define alpha 0.25

```

```

4 #define SCALE 1e+3
5
6 double dt;
7
8 double dqH(int j ,vector* q){
9     if(j<0||j>NUM-1){printf( "wrong index: the index of the vector entry should
10      be in range 0 to NUM-1\n");return 0;}
11     switch (j)
12     {
13     case 0:
14         return -(q->el[j]-q->el[j+1])-q->el[j]-alpha*pow((q->el[j]-q->el[j+1])
15             ,2)+alpha*pow((q->el[j]) ,2);
16         break;
17
18     case NUM-1:
19         return -(q->el[j])+(q->el[j-1]-q->el[j])-alpha*pow((q->el[j]) ,2)+alpha*
20             pow((q->el[j-1]-q->el[j]) ,2);
21         break;
22
23     default :
24         return -(q->el[j]-q->el[j+1])+(q->el[j-1]-q->el[j])-alpha*pow((q->el[j]-
25             q->el[j+1]) ,2)+alpha*pow((q->el[j-1]-q->el[j]) ,2);
26         break;
27     }
28 }
29
30 double eigen_transform(SqMatrix* A,vector* b,int k){
31     if(A->n!=b->n) return 0;
32     int i ;
33     double s=0;
34     for ( i = 0; i < b->n ; i++)
35     {
36         s+=A->el[k][ i ]*b->el[ i ];
37     }
38     return s;
39 }
40
41 int main(){
42     dt=PI/sin(PI/(2*(NUM+1)))/SCALE;
43     FILE* fdata=fopen( "C:/C_files/HW4/q5_cmp_sparsify.txt" , "w" );
44     SqMatrix* A=createZeroSqMatrix(NUM);
45     int i,j,k;
46     for ( k = 0; k < NUM; k++)
47     {
48         for ( j = 0; j < NUM; j++)
49         {
50             A->el[k][ j]=sqrt(2./ (NUM+1))*sin(PI*(k+1)*(j+1)/(NUM+1));
51         }
52     }
53 }
```

```

47     }
48 }
49 SqMatrix* Acopy=createZeroSqMatrix(NUM);
50 copy_SqMatrix(A,Acopy);
51
52 vector* q=createNewVector(NUM);
53 q->el[0]=4;
54 if(Solve_RankN_LinearEq(A,q)){
55     print_vector(q);
56 }
57 vector* p=createNewVector(NUM);
58
59 fprintf(fdata,"0\t");
60
61 for ( i=0;i<4;i++)
62 {
63     double Qk=eigen_transform(Acopy,q,i);
64     double Pk=eigen_transform(Acopy,p,i);
65     double wk=2*sin((i+1)*PI/2/(NUM+1));
66     double Ek=0.5*Pk*Pk+0.5*pow(wk*Qk,2);
67     fprintf(fdata,"%f\t",Ek);
68
69 }
70 fprintf(fdata,"\n");
71
72
73 vector* pf, *pr;
74 pr=createNewVector(NUM);
75 pf=createNewVector(NUM);
76 for ( j = 0; j < NUM; j++)
77 {
78     pf->el[j]=0.5*dt*dqH(j,q);
79 }
80
81 for ( i=1;i<=160*SCALE; i++){
82     copy_vector(pf,pr);
83     for ( j = 0; j < NUM; j++)
84     {
85         q->el[j]+=(pr->el[j]*dt);
86     }
87     for ( j = 0; j < NUM; j++){
88         pf->el[j]=pr->el[j]+dt*dqH(j,q);
89     }
90     average_vector(pr,pf,p);
91     fprintf(fdata,"%f\t", (double)i/SCALE);
92     for ( k=0;k<4;k++)
93     {

```

```

94     double Qk=eigen_transform(Acopy,q,k);
95     double Pk=eigen_transform(Acopy,p,k);
96     double wk=2*sin((k+1)*PI/2/(NUM+1));
97     double Ek=0.5*Pk*Pk+0.5*pow(wk*Qk,2);
98     fprintf(fdata,"%lf\n",Ek);
99 }
100 fprintf(fdata,"\\n");
101 }
102 return 0;
103 }
```

2.2 "q4.c"

```

1 #include "matrix.h"
2 #define NUM 32
3 #define alpha 0.25
4 #define SCALE 1e+3
5
6 double dt;
7
8 double dqH(int j,vector* q){
9     if(j<0||j>NUM-1){printf("wrong index: the index of the vector entry should
10      be in range 0 to NUM-1\\n");return 0;}
11     switch (j)
12     {
13     case 0:
14         return -(q->el[j]-q->el[j+1])-q->el[j]-alpha*pow((q->el[j]-q->el[j+1])
15             ,2)+alpha*pow((q->el[j]),2);
16         break;
17
18     case NUM-1:
19         return -(q->el[j])+(q->el[j-1]-q->el[j])-alpha*pow((q->el[j]),2)+alpha*
20             pow((q->el[j-1]-q->el[j]),2);
21         break;
22
23     default :
24         return -(q->el[j]-q->el[j+1])+(q->el[j-1]-q->el[j])-alpha*pow((q->el[j]-
25             q->el[j+1]),2)+alpha*pow((q->el[j-1]-q->el[j]),2);
26         break;
27     }
28 }
29
30 double eigen_transform(SqMatrix* A,vector* b,int k){
31     if(A->n!=b->n) return 0;
32     int i;
```

```

30     double s=0;
31     for ( i = 0; i < b->n ; i++)
32     {
33         s+=A->el [k] [ i ]*b->el [ i ];
34     }
35     return s;
36 }
37
38 int main()
39 {
40     dt=PI/sin (PI/(2*(NUM+1)))/SCALE;
41     FILE* fdata=fopen( "C:/ C_files/ HW4/q4.txt" , "w");
42     SqMatrix* A=createZeroSqMatrix (NUM);
43     int i,j,k;
44     for ( k = 0; k < NUM; k++)
45     {
46         for ( j = 0; j < NUM; j++)
47         {
48             A->el [k] [ j]=sqrt (2./ (NUM+1))*sin (PI*(k+1)*(j+1)/ (NUM+1));
49         }
50     }
51     SqMatrix* Acopy=createZeroSqMatrix (NUM);
52     copy_SqMatrix(A,Acopy);
53
54     vector* q=createNewVector (NUM);
55     q->el [0]=4;
56     if (Solve_RankN_LinearEq(A,q))
57     {
58         print_vector(q);
59     }
60     vector* p=createNewVector (NUM);
61
62     fprintf(fdata , "%f \n");
63
64     double Qk=eigen_transform (Acopy,q,0);
65     double Pk=eigen_transform (Acopy,p,0);
66     double wk=2*sin (PI/2/ (NUM+1));
67     double Ek=0.5*Pk*Pk+0.5*pow(wk*Qk,2);
68     fprintf(fdata , "%lf \n",Ek);
69
70
71     vector* pf, *pr;
72     pr=createNewVector (NUM);
73     pf=createNewVector (NUM);
74     for ( j = 0; j < NUM; j++)
75     {
76         pf->el [j]=0.5*dt*dqH(j ,q);

```

```

77 }
78
79 for ( i=1; i<=4000*SCALE; i++){
80     copy_vector( pf , pr ) ;
81     for ( j = 0; j < NUM; j++)
82     {
83         q->el [ j ]+=(pr->el [ j ] * dt ) ;
84     }
85     for ( j = 0; j < NUM; j++){
86         pf->el [ j ]=pr->el [ j ]+dt*dqH(j ,q) ;
87     }
88     average_vector( pr , pf ,p ) ;
89     fprintf( fdata , "%lf \t ",( double ) i /SCALE) ;
90
91     double Qk=eigen_transform( Acopy ,q ,0 ) ;
92     double Pk=eigen_transform( Acopy ,p ,0 ) ;
93     double wk=2*sin( PI/2/(NUM+1)) ;
94     double Ek=0.5*Pk*Pk+0.5*pow( wk*Qk ,2 ) ;
95     fprintf( fdata , "%lf \t ",Ek) ;
96     fprintf( fdata , "\n " );
97 }
98 return 0;
99 }
```

2.3 "q5.c"

```

1 #include "matrix.h"
2 #define NUM 32
3 #define OBSERVE 32
4 #define alpha 0.25
5 #define SCALE 1e+3
6
7 double dt ;
8
9 double dqH( int j ,vector* q){
10     if(j<0||j>NUM-1){printf( "wrong index: the index of the vector entry should
11         be in range 0 to NUM-1\n");return 0;}
12     switch (j)
13     {
14     case 0:
15         return -(q->el [ j ]-q->el [ j +1])-q->el [ j ]-alpha*pow(( q->el [ j ]-q->el [ j +1])
16             ,2)+alpha*pow(( q->el [ j ]),2);
17         break ;
18
19     case NUM-1:
20         return -(q->el [ j ])+(q->el [ j -1]-q->el [ j ])-alpha*pow(( q->el [ j ]),2)+alpha*
```

```

19         pow(( q->el [ j-1]-q->el [ j ]) ,2) ;
20     break;
21
22     default :
23         return -(q->el [ j ]-q->el [ j+1 ])+(q->el [ j-1]-q->el [ j ])-alpha*pow(( q->el [ j ]-
24             q->el [ j+1 ]) ,2)+alpha*pow(( q->el [ j-1]-q->el [ j ]) ,2);
25         break;
26     }
27
28 double eigen_transform(SqMatrix* A, vector* b, int k){
29     if(A->n!=b->n) return 0;
30     int i;
31     double s=0;
32     for ( i = 0; i < b->n ; i++)
33     {
34         s+=A->el [ k ] [ i ]*b->el [ i ];
35     }
36     return s;
37 }
38
39 int main(){
40     dt=PI/sin(PI/(2*(NUM+1)))/SCALE;
41     FILE* fdata=fopen("C:/C_files/HW4/q5_cmp_sparsify.txt","w");
42     SqMatrix* A=createZeroSqMatrix(NUM);
43     int i,j,k;
44     for ( k = 0; k < NUM; k++)
45     {
46         for ( j = 0; j < NUM; j++)
47         {
48             A->el [ k ] [ j]=sqrt(2./(NUM+1))*sin(PI*(k+1)*(j+1)/(NUM+1));
49         }
50     }
51     SqMatrix* Acopy=createZeroSqMatrix(NUM);
52     copy_SqMatrix(A,Acopy);
53
54     vector* q=createNewVector(NUM);
55     q->el [ 0 ]=4;
56     if(Solve_RankN_LinearEq(A,q)){
57         print_vector(q);
58     }
59     vector* p=createNewVector(NUM);
60
61     fprintf(fdata,"0\\t");
62
63     for ( i=0;i<OBSERVE; i++)

```

```

64    {
65        double Qk=eigen_transform(Acopy,q,i);
66        double Pk=eigen_transform(Acopy,p,i);
67        double wk=2*sin((i+1)*PI/2/(NUM+1));
68        double Ek=0.5*Pk*Pk+0.5*pow(wk*Qk,2);
69        fprintf(fdata,"%lf\t",Ek);
70
71    }
72
73    fprintf(fdata,"\n");
74
75
76    vector* pf, *pr;
77    pr=createNewVector(NUM);
78    pf=createNewVector(NUM);
79    for ( j = 0; j < NUM; j++)
80    {
81        pf->el[j]=0.5*dt*dqH(j,q);
82    }
83
84    for ( i=1;i<=4000*SCALE; i++){
85        copy_vector(pf,pr);
86        for ( j = 0; j < NUM; j++)
87        {
88            q->el[j]+=(pr->el[j]*dt);
89        }
90        for ( j = 0; j < NUM; j++){
91            pf->el[j]=pr->el[j]+dt*dqH(j,q);
92        }
93        average_vector(pr,pf,p);
94
95        if ( i%10==0){
96            fprintf(fdata,"%lf\t", (double) i/SCALE);
97            for ( k=0;k<OBSERVE; k++)
98            {
99                double Qk=eigen_transform(Acopy,q,k);
100               double Pk=eigen_transform(Acopy,p,k);
101               double wk=2*sin((k+1)*PI/2/(NUM+1));
102               double Ek=0.5*Pk*Pk+0.5*pow(wk*Qk,2);
103               fprintf(fdata,"%lf\t",Ek);
104            }
105            fprintf(fdata,"\n");
106        }
107    }
108    return 0;
109 }
```

2.4 "plotTool.py"

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 plt.rcParams[ "font.sans-serif"]=[ "SimHei"]
5 plt.rcParams[ "axes.unicode_minus"]=False
6 a = np.loadtxt( "C:/ C_files/HW4/q6_to_q5_cmp.txt")
7
8 for i in range(32):
9     plt.figure( figsize=(10,5) ,dpi=150)
10    plt.xlim(-10,710)
11    plt.ylim(-0.01,0.5)
12    plt.xlabel( "t")
13    plt.ylabel( "E"+str( int( i+1)))
14    plt.title( chr(946)+"-FPU,""Q1(0)=10条件下E"+str( int( i+1))+"随 t 变化图")
15    plt.plot( a[:,0] ,a[:, i+1], linewidth=0.5)
16    plt.savefig( "C:\ C_files\HW4\ report\ q6_pics\ cmp\ E"+str( int( i+1))+".png")
)
```

2.5 "q6_to_q2.c"

```
1 #include "matrix.h"
2 #define NUM 32
3 #define beta 1
4 #define SCALE 1e+3
5
6 double dt;
7
8 double dqH( int j ,vector* q){
9     if(j<0||j>NUM-1){printf( "wrong index: the index of the vector entry should
10      be in range 0 to NUM-1\n");return 0;}
11     switch (j)
12     {
13     case 0:
14         return -(q->el[j]-q->el[j+1])-q->el[j]-beta*pow((q->el[j]-q->el[j+1]) ,3)
15             +beta*pow((-q->el[j]) ,3);
16         break;
17     case NUM-1:
18         return -(q->el[j])+(q->el[j-1]-q->el[j])-beta*pow((q->el[j]) ,3)+beta*pow
19             ((q->el[j-1]-q->el[j]) ,3);
20         break;
21     default :
22         return -(q->el[j]-q->el[j+1])+(q->el[j-1]-q->el[j])-beta*pow((q->el[j]-q
23             ->el[j+1]) ,3)+beta*pow((q->el[j-1]-q->el[j]) ,3);
```

```

22     break;
23 }
24 }
25
26 double eigen_transform(SqMatrix* A, vector* b, int k){
27     if(A->n!=b->n) return 0;
28     int i;
29     double s=0;
30     for ( i = 0; i < b->n ; i++)
31     {
32         s+=A->el [k] [ i ]*b->el [ i ];
33     }
34     return s;
35 }
36
37 int main(){
38     dt=PI/sin(PI/(2*(NUM+1)))/SCALE;
39     FILE* fdata=fopen( "C:/C_files/HW4/q6_to_q2.txt" , "w" );
40     SqMatrix* A=createZeroSqMatrix(NUM);
41     int i,j,k;
42     for ( k = 0; k < NUM; k++)
43     {
44         for ( j = 0; j < NUM; j++)
45         {
46             A->el [k] [ j]=sqrt(2./(NUM+1))*sin(PI*(k+1)*(j+1)/(NUM+1));
47         }
48     }
49     SqMatrix* Acopy=createZeroSqMatrix(NUM);
50     copy_SqMatrix(A,Acopy);
51
52     vector* q=createNewVector(NUM);
53     q->el [0]=10;
54     if(Solve_RankN_LinearEq(A,q)){
55         print_vector(q);
56     }
57     vector* p=createNewVector(NUM);
58
59     fprintf(fdata , "%f \n" );
60
61     for ( i=0;i<7;i+=2)
62     {
63         double Qk=eigen_transform(Acopy,q, i );
64         double Pk=eigen_transform(Acopy,p, i );
65         double wk=2*sin((i+1)*PI/2/(NUM+1));
66         double Ek=0.5*Pk*Pk+0.5*pow(wk*Qk, 2 );
67         fprintf(fdata , "%f \n" ,Ek);
68

```

```

69 }
70 fprintf(fdata , "\n");
71
72
73     vector* pf, *pr;
74     pr=createNewVector( NUM );
75     pf=createNewVector( NUM );
76     for ( j = 0; j < NUM; j++)
77     {
78         pf->el [ j ]=0.5*dt*dqH(j , q );
79     }
80
81     for ( i=1;i<=100*SCALE; i++){
82         copy_vector( pf , pr );
83         for ( j = 0; j < NUM; j++)
84         {
85             q->el [ j ]+=(pr->el [ j ]*dt );
86         }
87         for ( j = 0; j < NUM; j++){
88             pf->el [ j ]=pr->el [ j ]+dt*dqH(j , q );
89         }
90         average_vector( pr , pf , p );
91         fprintf(fdata , "%f \t ",(double) i /SCALE);
92         for ( k=0;k<7;k+=2)
93         {
94             double Qk=eigen_transform( Acopy , q , k );
95             double Pk=eigen_transform( Acopy , p , k );
96             double wk=2*sin (( k+1)*PI/2/(NUM+1));
97             double Ek=0.5*Pk*Pk+0.5*pow(wk*Qk, 2 );
98             fprintf(fdata , "%f \t ",Ek);
99         }
100        fprintf(fdata , "\n");
101    }
102    return 0;
103 }
```

2.6 "q6_to_q4.c"

```

1 #include "matrix.h"
2 #define NUM 32
3 #define beta 1
4 #define SCALE 1e+4
5
6 double dt;
7
8 double dqH( int j , vector* q){
```

```

9   if(j<0||j>NUM-1){printf("wrong index: the index of the vector entry should
10    be in range 0 to NUM-1\n");return 0;}
11
12  switch (j)
13  {
14    case 0:
15      return -(q->el[j]-q->el[j+1])-q->el[j]-beta*pow((q->el[j]-q->el[j+1]),3)
16      +beta*pow((-q->el[j]),3);
17      break;
18
19    case NUM-1:
20      return -(q->el[j])+(q->el[j-1]-q->el[j])-beta*pow((q->el[j]),3)+beta*pow
21      ((q->el[j-1]-q->el[j]),3);
22      break;
23
24    default :
25      return -(q->el[j]-q->el[j+1])+(q->el[j-1]-q->el[j])-beta*pow((q->el[j]-q
26      ->el[j+1]),3)+beta*pow((q->el[j-1]-q->el[j]),3);
27      break;
28  }
29
30 double eigen_transform(SqMatrix* A, vector* b, int k){
31   if(A->n!=b->n) return 0;
32   int i;
33   double s=0;
34   for ( i = 0; i < b->n ; i++)
35   {
36     s+=A->el[k][i]*b->el[i];
37   }
38   return s;
39 }
40
41 int main()
42 {
43   dt=PI/sin(PI/(2*(NUM+1)))/SCALE;
44   FILE* fdata=fopen("C:/C_files/HW4/q6_to_q4.txt","w");
45   SqMatrix* A=createZeroSqMatrix(NUM);
46   int i,j,k;
47   for ( k = 0; k < NUM; k++)
48   {
49     for ( j = 0; j < NUM; j++)
50     {
51       A->el[k][j]=sqrt(2./(NUM+1))*sin(PI*(k+1)*(j+1)/(NUM+1));
52     }
53   }
54   SqMatrix* Acopy=createZeroSqMatrix(NUM);
55   copy_SqMatrix(A,Acopy);
56 }
```

```

52     vector* q=createNewVector(NUM) ;
53     q->el[0]=10;
54     if(Solve_RankN_LinearEq(A,q)){
55         print_vector(q);
56     }
57     vector* p=createNewVector(NUM) ;
58
59     fprintf(fdata , "0\|t");
60
61
62     double Qk=eigen_transform(Acopy,q,0) ;
63     double Pk=eigen_transform(Acopy,p,0) ;
64     double wk=2*sin(PI/2/(NUM+1)) ;
65     double Ek=0.5*Pk*Pk+0.5*pow(wk*Qk,2) ;
66     fprintf(fdata , "%lf\|t",Ek);
67
68     fprintf(fdata , "\n");
69
70
71     vector* pf , *pr;
72     pr=createNewVector(NUM) ;
73     pf=createNewVector(NUM) ;
74     for ( j = 0; j < NUM; j++)
75     {
76         pf->el[j]=0.5*dt*dqH(j,q) ;
77     }
78
79     for ( i=1;i<=700*SCALE; i++){
80         copy_vector(pf,pr) ;
81         for ( j = 0; j < NUM; j++)
82         {
83             q->el[j]+=(pr->el[j]*dt) ;
84         }
85         for ( j = 0; j < NUM; j++){
86             pf->el[j]=pr->el[j]+dt*dqH(j,q) ;
87         }
88         average_vector(pr, pf, p) ;
89
90         if(i%10==0){
91             fprintf(fdata , "%lf\|t", (double)i /SCALE) ;
92
93             double Qk=eigen_transform(Acopy,q,0) ;
94             double Pk=eigen_transform(Acopy,p,0) ;
95             double wk=2*sin(PI/2/(NUM+1)) ;
96             double Ek=0.5*Pk*Pk+0.5*pow(wk*Qk,2) ;
97             fprintf(fdata , "%lf\|t",Ek) ;
98

```

```

99         fprintf(fdata, "\n");
100    }
101
102    }
103    return 0;
104 }
```

2.7 "q6_to_q5.c"

```

1 #include "matrix.h"
2 #define NUM 32
3 #define OBSERVE 32
4 #define beta 1
5 #define SCALE 1e+3
6
7 double dt;
8
9 double dqH(int j, vector* q){
10    if(j<0||j>NUM-1){printf("wrong index: the index of the vector entry should
11        be in range 0 to NUM-1\n");return 0;}
12    switch (j)
13    {
14        case 0:
15            return -(q->el[j]-q->el[j+1])-q->el[j]-beta*pow((q->el[j]-q->el[j+1]),3)
16                +beta*pow((-q->el[j]),3);
17            break;
18
19        case NUM-1:
20            return -(q->el[j])+(q->el[j-1]-q->el[j])-beta*pow((q->el[j]),3)+beta*pow
21                ((q->el[j-1]-q->el[j]),3);
22            break;
23
24        default:
25            return -(q->el[j]-q->el[j+1])+(q->el[j-1]-q->el[j])-beta*pow((q->el[j]-q
26                ->el[j+1]),3)+beta*pow((q->el[j-1]-q->el[j]),3);
27            break;
28    }
29 }
30
31 double eigen_transform(SqMatrix* A, vector* b, int k){
32     if(A->n!=b->n) return 0;
33     int i;
34     double s=0;
35     for ( i = 0; i < b->n ; i++)
36     {
37         s+=A->el[k][i]*b->el[i];
38     }
39 }
```

```

34     }
35     return s;
36 }
37
38 int main() {
39     dt=PI/sin(PI/(2*(NUM+1)))/SCALE;
40     FILE* fdata=fopen("C:/C_files/HW4/q6_to_q5.txt","w");
41     SqMatrix* A=createZeroSqMatrix(NUM);
42     int i,j,k;
43     for ( k = 0; k < NUM; k++)
44     {
45         for ( j = 0; j < NUM; j++)
46         {
47             A->el[k][j]=sqrt(2./(NUM+1))*sin(PI*(k+1)*(j+1)/(NUM+1));
48         }
49     }
50     SqMatrix* Acopy=createZeroSqMatrix(NUM);
51     copy_SqMatrix(A,Acopy);
52
53     vector* q=createNewVector(NUM);
54     q->el[0]=100;
55     if(Solve_RankN_LinearEq(A,q)){
56         print_vector(q);
57     }
58     vector* p=createNewVector(NUM);
59
60     fprintf(fdata,"%0\|t");
61
62     for ( i=0;i<OBSERVE; i++)
63     {
64         double Qk=eigen_transform(Acopy,q,i);
65         double Pk=eigen_transform(Acopy,p,i);
66         double wk=2*sin((i+1)*PI/2/(NUM+1));
67         double Ek=0.5*Pk*Pk+0.5*pow(wk*Qk,2);
68         fprintf(fdata,"%1f\|t",Ek);
69     }
70
71     fprintf(fdata,"\\n");
72
73     vector* pf, *pr;
74     pr=createNewVector(NUM);
75     pf=createNewVector(NUM);
76     for ( j = 0; j < NUM; j++)
77     {
78         pf->el[j]=0.5*dt*dqH(j,q);
79     }

```

```

81
82     for ( i=1; i<=700*SCALE; i++){
83         copy_vector(pf,pr);
84         for ( j = 0; j < NUM; j++)
85         {
86             q->el[j]+=(pr->el[j]*dt);
87         }
88         for ( j = 0; j < NUM; j++){
89             pf->el[j]=pr->el[j]+dt*dqH(j,q);
90         }
91
92         if ( i%10==0){
93             average_vector(pr,pf,p);
94             fprintf(fdata,"%lf\t", (double) i /SCALE);
95
96             for ( k=0;k<OBSERVE;k++)
97             {
98                 double Qk=eigen_transform(Acopy,q,k);
99                 double Pk=eigen_transform(Acopy,p,k);
100                double wk=2*sin((k+1)*PI/2/(NUM+1));
101                double Ek=0.5*Pk*Pk+0.5*pow(wk*Qk,2);
102                fprintf(fdata,"%lf\t",Ek);
103
104            }
105
106            fprintf(fdata,"\n");
107        }
108    }
109
110    return 0;
111}

```

2.8 "q7.c"

```

1 #include "matrix.h"
2 #define NUM 16
3 #define beta 1
4 #define SCALE 1e+4
5
6 double dt;
7
8 double dqH( int j ,vector* q){
9     if(j<0||j>NUM-1){printf( "wrong index: the index of the vector entry should
10       be in range 0 to NUM-1\n");return 0;}
11     switch (j)
12     {

```

```

12   case 0:
13     return -(q->el[j]-q->el[j+1])-q->el[j]-beta*pow((q->el[j]-q->el[j+1]),3)
14     +beta*pow((-q->el[j]),3);
15     break;
16
17   case NUM-1:
18     return -(q->el[j])+(q->el[j-1]-q->el[j])-beta*pow((q->el[j]),3)+beta*pow
19     ((q->el[j-1]-q->el[j]),3);
20     break;
21
22   default:
23     return -(q->el[j]-q->el[j+1])+(q->el[j-1]-q->el[j])-beta*pow((q->el[j]-q
24     ->el[j+1]),3)+beta*pow((q->el[j-1]-q->el[j]),3);
25     break;
26   }
27
28 double eigen_transform(SqMatrix* A, vector* b, int k){
29   if(A->n!=b->n) return 0;
30   int i;
31   double s=0;
32   for ( i = 0; i < b->n ; i++)
33   {
34     s+=A->el[k][i]*b->el[i];
35   }
36   return s;
37 }
38
39 int main(){
40   dt=PI/sin(PI/(2*(NUM+1)))/SCALE;
41   FILE* fdata=fopen("C:/C_files/HW4/q7_intensify.txt","w");
42   SqMatrix* A=createZeroSqMatrix(NUM);
43   int i,j,k;
44   for ( k = 0; k < NUM; k++)
45   {
46     for ( j = 0; j < NUM; j++)
47     {
48       A->el[k][j]=sqrt(2./(NUM+1))*sin(PI*(k+1)*(j+1)/(NUM+1));
49     }
50   }
51   SqMatrix* Acopy=createZeroSqMatrix(NUM);
52   copy_SqMatrix(A,Acopy);
53
54   vector* q=createNewVector(NUM);
55   q->el[10]=1;
56   if(Solve_RankN_LinearEq(A,q)){
57     print_vector(q);

```

```

56 }
57 vector* p=createNewVector(NUM);
58
59 fprintf(fdata , "0\|t");
60
61 for ( i=8;i<13;i++)
62 {
63     double Qk=eigen_transform(Acopy,q,i);
64     double Pk=eigen_transform(Acopy,p,i);
65     double wk=2*sin(( i+1)*PI/2/(NUM+1));
66     double Ek=0.5*Pk*Pk+0.5*pow(wk*Qk,2);
67     fprintf(fdata , "%e\|t",Ek);
68
69 }
70 fprintf(fdata , "\n");
71
72
73 vector* pf, *pr;
74 pr=createNewVector(NUM);
75 pf=createNewVector(NUM);
76 for ( j = 0; j < NUM; j++)
77 {
78     pf->el[j]=0.5*dt*dqH(j,q);
79 }
80
81 for ( i=1;i<=100*SCALE; i++){
82     copy_vector(pf,pr);
83     for ( j = 0; j < NUM; j++)
84     {
85         q->el[j]+=(pr->el[j]*dt);
86     }
87     for ( j = 0; j < NUM; j++){
88         pf->el[j]=pr->el[j]+dt*dqH(j,q);
89     }
90     average_vector(pr,pf,p);
91     fprintf(fdata , "%f\|t", (double) i /SCALE);
92     for ( k=8;k<13;k++)
93     {
94         double Qk=eigen_transform(Acopy,q,k);
95         double Pk=eigen_transform(Acopy,p,k);
96         double wk=2*sin(( k+1)*PI/2/(NUM+1));
97         double Ek=0.5*Pk*Pk+0.5*pow(wk*Qk,2);
98         fprintf(fdata , "%e\|t",Ek);
99     }
100    fprintf(fdata , "\n");
101 }
102 return 0;

```

103 }

2.9 "q8.c"

```

1 #include "matrix.h"
2 #define NUM 128
3 #define BETA 0
4 #define K 11
5 #define B 0.5
6 #define SCALE 1e+3
7
8
9
10 double dt;
11 double wk;
12
13 double dqH( int j ,vector* q){
14     if(j<0||j>NUM-1){printf("wrong index: the index of the vector entry should
15         be in range 0 to NUM-1\n");return 0;}
16     switch (j)
17     {
18     case 0:
19         return -(q->el[j]-q->el[j+1])-q->el[j]-BETA*pow((q->el[j]-q->el[j+1]),3)
20             +BETA*pow((-q->el[j]),3);
21         break;
22
23     case NUM-1:
24         return -(q->el[j])+(q->el[j-1]-q->el[j])-BETA*pow((q->el[j]),3)+BETA*pow
25             ((q->el[j-1]-q->el[j]),3);
26         break;
27
28     default :
29         return -(q->el[j]-q->el[j+1])+(q->el[j-1]-q->el[j])-BETA*pow((q->el[j]-q
30             ->el[j+1]),3)+BETA*pow((q->el[j-1]-q->el[j]),3);
31         break;
32     }
33 }
34
35 void fprintf_vector(FILE* fdata ,vector* b){
36     int i ;
37     for ( i = 0; i < b->n; i++)
38     {
39         fprintf(fdata,"%1f",b->el[i]);
40         if(i!=b->n-1)fprintf(fdata,"|t");
41     }
42     fprintf(fdata,"\\n");
43 }
```

```

39 }
40
41 int main() {
42     wk=2*sin(PI*K/(2*(NUM+1)));
43     dt=2*PI/wk/SCALE;
44     FILE* fdata=fopen("C:/C_files/HW4/q8_beta=0.txt","w");
45     int i,j,k;
46     vector* q=createNewVector(NUM);
47     vector* p=createNewVector(NUM);
48
49     for ( i = 0; i < NUM; i++)
50     {
51         q->el[i]=B*cos(PI*K*(i+1-NUM/2)/(NUM+1))/cosh(sqrt(3./2)*B*wk*(i+1-NUM/2));
52         p->el[i]=B/cosh(sqrt(3./2)*B*wk*(i+1-NUM/2))*(wk*(1+3./16*wk*wk*B*B)*sin(PI*K*(i+1-NUM/2)/(NUM+1))+sqrt(3./2)*B*cos(PI*K*(i+1-NUM/2)/(NUM+1))*sin(PI*K/(NUM+1))*tanh(sqrt(3./2)*B*wk*(i+1-NUM/2)));
53     }
54
55     fprintf(fdata,"%0\|t");
56     fprintf_vector(fdata,q);
57
58     vector* pf, *pr;
59     pr=createNewVector(NUM);
60     pf=createNewVector(NUM);
61     for ( j = 0; j < NUM; j++)
62     {
63         pf->el[j]=p->el[j]+0.5*dt*dqH(j,q);
64     }
65
66     for ( i=1;i<=200*SCALE; i++){
67         copy_vector(pf,pr);
68         for ( j = 0; j < NUM; j++)
69         {
70             q->el[j]+=(pr->el[j]*dt);
71         }
72         for ( j = 0; j < NUM; j++){
73             pf->el[j]=pr->el[j]+dt*dqH(j,q);
74         }
75         average_vector(pr,pf,p);
76         if (i%100==0){
77             fprintf(fdata,"%lf\|t", (double)i/SCALE);
78             fprintf_vector(fdata,q);
79         }
80     }
81     return 0;
82 }
```

2.10 "q8.py"

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math as m
4
5 NUM =128
6 BETA =0
7 K =11
8 B =0.5
9 SCALE =1000
10 x=np.arange(1,129)
11 wk=2*m.sin(m.pi*K/(2*(NUM+1)))
12 dt=2*m.pi/wk/SCALE
13
14 q=np.zeros(NUM)
15 p=np.zeros(NUM)
16
17 for i in range(NUM):
18     q[i]=B*m.cos(m.pi*K*(i+1-NUM/2)/(NUM+1))/m.cosh(m.sqrt(3./2)*B*wk*(i+1-NUM/2))
19     p[i]=B/m.cosh(m.sqrt(3./2)*B*wk*(i+1-NUM/2))*(wk*(1+3./16*wk*wk*B*B)*m.sin(m.pi*K*(i+1-NUM/2)/(NUM+1))+m.sqrt(3./2)*B*m.cos(m.pi*K*(i+1-NUM/2)/(NUM+1))*m.sin(m.pi*K/(NUM+1))*m.tanh(m.sqrt(3./2)*B*wk*(i+1-NUM/2)))
20
21 plt.figure(figsize=(10,8),dpi=150)
22 plt.xlim(0,130)
23 plt.ylim(-0.8,0.8)
24 plt.scatter(x,q)
25 plt.xlabel("Particle Index")
26 plt.ylabel("Displacement q")
27 plt.title(chr(946)+"+"+str(BETA)+" t="+str(0.0), fontsize='xx-large')
28 plt.savefig("C:\C_files\HW4\q8_beta="+str(BETA)+"(2.0)\t="+str(0.00)+".png")
29
30 pr=np.zeros(NUM)
31 pf=np.zeros(NUM)
32
33 def dp(q:np.array):
34     assert np.size(p)>=1,
35     "Displacement vector q should not be void"
36     dp=np.zeros(np.size(q))
37     for j in range(np.size(q)):
38         if j==0:
39             dp[j]=-(q[j]-q[j+1])+(-q[j])-BETA*(q[j]-q[j+1])**3+BETA*(-q[j])**3
40         elif j==np.size(q)-1:
```

```

41         dp[j]=-(q[j])+(q[j-1]-q[j])-BETA*(q[j])**3+BETA*(q[j-1]-q[j])**3
42     else:
43         dp[j]=-(q[j]-q[j+1])+(q[j-1]-q[j])-BETA*(q[j]-q[j+1])**3+BETA*(q[j-1]-q[j])**3
44     return dp
45
46 pf=p+0.5*dt*dp(q)
47
48 for i in range(1,200*SCALE+1):
49     pr=pf
50     q=q+pr*dt
51     pf=pr+dt*dp(q)
52     if i%100==0:
53         plt.figure(figsize=(10,8),dpi=150)
54         plt.xlim(0,130)
55         plt.ylim(-0.8,0.8)
56         plt.scatter(x,q)
57         plt.xlabel("Particle Index")
58         plt.ylabel("Displacement q")
59         plt.title(chr(946)+"= "+str(BETA)+" "+t)+" "+str(float(i)/SCALE),
60                   fontsize='xx-large')
61         plt.savefig("C:\\\\C_files\\\\HW4\\\\q8_beta="+str(BETA)+"(2.0)\\\\t="+str(float(i)/SCALE)+".png")

```

参考文献

- [1] 李庆扬, 王能超, 易大义: 数值分析, 261-263 页, 269-271 页, 2008 年 12 月第五版