

# 大作业 3 Hofstadter 蝴蝶

张钰坤

2000011314

(C 语言实现)

2022 年 4 月 8 日

## 目录

<b>1 题目解答</b>	<b>2</b>
1.1 第一问	4
1.2 第二问	6
1.3 第三问	7
1.4 第四问	8
1.5 第五问	11
1.6 第六问	13
<b>2 附录</b>	<b>14</b>
2.1 “matrix.h” 头文件	14
2.2 第一问源代码	18
2.3 第二、三问源代码	19
2.4 第四、五问源代码	19
2.5 第六问源代码	20

# 1 题目解答

本题的求解核心问题是实对称矩阵的特征值问题。那么如果写出一个函数可以求出任意实对称矩阵的本征值，那么所有问题都将迎刃而解。

首先我们需要定义一个方阵结构体作为实对称矩阵变量类型。

```
1 typedef struct SqMetrix{  
2     int n; //储存方阵行数（列数）  
3     double** el;  
4 }SqMetrix;
```

其中，方阵元素用二维数组储存。值得一提的是，由于矩阵规模未定，如果这里方阵 SqMetrix 包含一个二维数组，需要将初始规模设置得很大，比如如下定义，

```
1 #define MAX 10000  
2 struct SqMetrix{  
3     int n; //储存方阵行数（列数）  
4     double el [MAX] [MAX];  
5 }
```

那么在处理小规模问题得时候，比如我算 2 维方阵得本征值，就浪费了大部分存储空间。因此，这里使用二维数组指针，方便动态申请储存空间。

接下来我们需要以下操作来辅助运算。

```
1 //SqMetrix抽象数据类型  
2 createZeroSqMatrix(int n); //建立边长为n的空矩阵  
3 is_sym_sq(SqMetrix*); //判断矩阵是否为对称阵  
4 printMetrix(SqMetrix*); //打印矩阵至终端  
5 eigenvalue_sym_sq(SqMetrix*) //求出实对称方阵的特征值组
```

前三个函数实现起来十分容易，具体代码实现参见附录-matrix.h 源代码中对应函数。

下面对函数 (**double** \*)**eigenvalue\_sym\_sq**(SqMetrix\* A) 作特别说明，笔者这里使用的是先 Householder 变换化为三对角矩阵，再使用位移 QR 迭代得到本征值组。

---

**Algorithm 1** 实对称阵本征值求解

---

**Input:** 待求本征值方阵的指针 SqMetrix\* A, 要求 A 对称非空;

**Output:** 方阵的本征值数组头指针 double\* b;

```
1: if A 不是对称阵或 A 阶数非正 then
2:     返回空指针;
3: end if
4: if A 阶数是 1 then
5:     A11 就是本征值, 写入本征值数组, 返回数组指针;
6: end if
7: if A 阶数是 2 then
8:     本征值可以解析地算出:  $\lambda_{\pm} = \frac{1}{2}(A_{11} + A_{22} \pm \sqrt{(A_{11} + A_{22})^2 + 4A_{12}^2})$ , 写入本征值数组, 返回数组指针;
9: end if
10: //A 通过 Householder 变换三对角化为 H
11: 从第一列开始直到倒数第三列, 如果低对角线以下的向量不为 0 向量, 那么进行 Householder 变换使它化为 0 向量。
12: //带原点位移的 QR 迭代
13: 迭代规模 n←H 的阶数;
14: while n>1 do
15:     单步位移 s← Hnn;
16:     Hin − sI = QR, Hout = RQ + sI
17:     if Hout(n, n − 1) 充分小 then
18:         得到一个本征值 Hout(n; n), 可以对较小的矩阵计算其他本征值, 故 n--;
19:     end if
20: end while
21: return H 主对角数组;
```

---

本算法中 Householder 变换参考 [1]261-263 页中 8.3.4 节计算方法, QR 迭代参考 [1]270 页中算法 1。位移 QR 算法中为何单步位移取  $H_{nn}$  在 [1]269 页定理 24 有十分清晰的论述。

值得一提的是, 在每一列 Householder 变换时需要求出反射矩阵, 然后和待变换的块矩阵作乘法。这里需要合理选取运算方法来减少计算次数。

假设将要进行第 k 次 Householder 变换。经过 k-1 次 Householder 变换得到如下矩阵。

$$\begin{pmatrix} a_{11} & -\sigma_1 & & & & \\ -\sigma_1 & a_{22} & -\sigma_2 & & & \\ & -\sigma_2 & \ddots & \ddots & & \\ & & \ddots & a_{k-1,k-1} & -\sigma_{k-1} & \\ & & & -\sigma_{k-1} & a_{kk} & \cdots & a_{kn} \\ & & & & \vdots & & \vdots \\ & & & & a_{nk} & \cdots & a_{nn} \end{pmatrix}$$

假设发现  $(a_{k+1,k}, a_{k+2,k}, \dots, a_{nn})^\top \neq \vec{0}$ , 我们需要进行单步 Householder 变换。其实只需对右下角的 n-k+1 阶矩阵作操作。

设

$$\begin{pmatrix} a_{kk} & a_{k,k+1} & \cdots & a_{kn} \\ a_{k+1,k} & \ddots & & \vdots \\ \vdots & & & \\ a_{n,k} & \cdots & & a_{nn} \end{pmatrix} \triangleq \begin{pmatrix} a_{kk} & \vec{c}_k^\top \\ \vec{c}_k & A_{k+1} \end{pmatrix}$$

根据 [1]262 页式 (3.6), 理论上我们需要求出反射矩阵  $R_k = I - \beta_k^{-1} \vec{u}_k \vec{u}_k^\top$  其中,

$$\sigma_k = \operatorname{sgn}(a_{k+1,k}) |\vec{c}_k| \quad (1)$$

$$\vec{u}_k = \vec{c}_k + \sigma_k \vec{e}_1 \quad (2)$$

$$\beta_k = \sigma_k (a_{k+1,k} + \sigma_k) \quad (3)$$

(4)

容易验证

$$\begin{pmatrix} 1 & \\ & R_k \end{pmatrix} \begin{pmatrix} a_{kk} & \vec{c}_k^\top \\ \vec{c}_k & A_{k+1} \end{pmatrix} \begin{pmatrix} 1 & \\ & R_k \end{pmatrix} = \begin{pmatrix} a_{kk} & \vec{c}_k'^\top \\ \vec{c}_k' & R_k A_{k+1} R_k \end{pmatrix}$$

其中  $\vec{c}_k' = (-\sigma_k, 0, \dots, 0)$

在程序设计过程中, 不能先将矩阵  $R_k$  算出来, 再做两次矩阵乘法算出  $R_k A_{k+1} R_k$ , 因为这样单步 Householder 变换时间复杂度是  $O(n-k+1)^3$  的, 整个 Householder 变换时间复杂度就是  $1^3 + 2^3 + \dots + (n-2)^3 = O(n^4)$ , 复杂度过高。

所以, 应当尽量选择向量进行运算来降低复杂度。注意到

$$R_k A_{k+1} R_k = A_{k+1} - \beta_k^{-1} \vec{u}_k (\vec{u}_k^\top A_{k+1}) - \beta_k^{-1} (A_{k+1} \vec{u}_k) \vec{u}_k^\top - \beta_k^{-2} \vec{u}_k (\vec{u}_k^\top A_{k+1} \vec{u}_k) \vec{u}_k^\top$$

注意到, 由于  $A_{k+1}$  的对称性,  $\vec{u}_k^\top A_{k+1}$  和  $A_{k+1} \vec{u}_k$  互为转置关系; 而且,  $\vec{u}_k^\top A_{k+1} \vec{u}_k$  是一个数。于是, 我们提出如下计算方法。

$$\begin{aligned} \begin{pmatrix} a_{kk} & \vec{c}_k^\top \\ \vec{c}_k & A_{k+1} \end{pmatrix} &\xrightarrow[O(1)]{\vec{u}_k \leftarrow \vec{c}_k + \sigma_k \vec{e}_1} \begin{pmatrix} a_{kk} & \vec{c}_k^\top \\ \vec{u}_k & A_{k+1} \end{pmatrix} \xrightarrow[O(n-k+1)^2]{\vec{c}_k^\top \leftarrow \vec{u}_k^\top A_{k+1}} \begin{pmatrix} a_{kk} & \vec{u}_k^\top A_{k+1} \\ \vec{u}_k & A_{k+1} \end{pmatrix} \\ &\xrightarrow[O(n-k+1)^2]{A_{k+1} \leftarrow A_{k+1} - \beta_k^{-1} \vec{u}_k (\vec{u}_k^\top A_{k+1}) - \beta_k^{-1} (A_{k+1} \vec{u}_k) \vec{u}_k^\top - \beta_k^{-2} \vec{u}_k (\vec{u}_k^\top A_{k+1} \vec{u}_k) \vec{u}_k^\top} \begin{pmatrix} a_{kk} & \vec{u}_k^\top A_{k+1} \\ \vec{u}_k & A'_{k+1} \end{pmatrix} \\ &\xrightarrow[O(n-k+1)]{\vec{u}_k \leftarrow (-\sigma_k, 0, \dots, 0), \vec{u}_k^\top A_{k+1} \leftarrow (-\sigma_k, 0, \dots, 0)^\top} \begin{pmatrix} a_{kk} & \vec{c}_k^\top \\ \vec{c}_k' & A'_{k+1} \end{pmatrix} \end{aligned}$$

其中每一步的时间复杂度标在了箭头下方。可以看出, 这种计算方法单步 Householder 变换时间复杂度是  $O(n-k+1)^2$  的, 总 Householder 变换的时间复杂度为  $1^2 + 2^2 + \dots + (n-2)^2 = O(n^3)$ , 时间复杂度减少了一个量级。

引言部分所有函数源代码见附录”matrix.h”头文件。

## 1.1 第一问

本题即求解矩阵 K 在阶数为 2, 16, 128, 1024 时的所有特征值, 特别地, 二阶 K 为

$$\begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix}$$

具体代码实现见附录。

$n=2, 16, 128, 1024$  时输出结果本别见 “q1-N-2.txt” “q1-N-16.txt” “q1-N-128.txt” “q1-N-1024.txt”。其中左列是数值结果, 右侧是解析结果。

特别的是，代码中在输出本征值数组时，同时输出了  $2(1 - \cos \frac{2\pi k}{n})$ ,  $k = 1, 2, \dots, n$  的值。但是他们未必很好的一一对应了起来，我们需要将输出数据（一列数值结果，一列解析结果）复制到 excel 中，两列分别降序排列，再计算两列差的绝对值，最后将差的绝对值求和。

得到如下结果

n=2			
数值结果	解析结果	差的 绝对值	差的绝对值 求和
0.000000	0.000000	0.000000	0.000000
4.000000	4.000000	0.000000	

图 1:  $q=1-n=2$ -解析数值比对

n=16			
数值结果	解析结果	差的 绝对值	差的绝对值 求和
0.000000	0.000000	0.000000	0.000000
0.152241	0.152241	0.000000	
0.152241	0.152241	0.000000	
0.585786	0.585786	0.000000	
0.585786	0.585786	0.000000	
1.234633	1.234633	0.000000	

图 2: q1-n=16-解析数值比对

n=128			
数值结果	解析结果	差的 绝对值	差的绝对值 求和
0.000000	0.000000	0.000000	0.000000
0.002409	0.002409	0.000000	
0.002409	0.002409	0.000000	
0.009631	0.009631	0.000000	
0.009631	0.009631	0.000000	
0.021647	0.021647	0.000000	

图 3: q1-n=128-解析数值比对

n=1024			
数值结果	解析结果	差的绝对值	差的绝对值求和
0.000000	0.000000	0.000000	0.000000
0.000038	0.000038	0.000000	
0.000038	0.000038	0.000000	
0.000151	0.000151	0.000000	
0.000151	0.000151	0.000000	
0.000329	0.000329	0.000000	

图 4: q1-n=1024-解析数值比对

可以看出，在六位精度下解析结果与数值结果完全相同。

原始 excel 表格处理结果见“dataAnalysis.xlsx”中 sheet “q1”。

## 1.2 第二问

矩阵  $K$  变为

$$\begin{pmatrix} 2 + \frac{g_0 + g_1 \cos 2\pi(\alpha)}{\kappa} & -1 & & -1 \\ -1 & 2 + \frac{g_0 + g_1 \cos 2\pi(2\alpha)}{\kappa} & -1 & \\ & -1 & \ddots & \ddots \\ & & \ddots & -1 \\ -1 & & -1 & 2 + \frac{g_0 + g_1 \cos 2\pi(n\alpha)}{\kappa} \end{pmatrix} \triangleq K$$

特别地， $g_0 = 0$  时有

$$\begin{pmatrix} 2 + \frac{g_1 \cos 2\pi(\alpha)}{\kappa} & -1 & & -1 \\ -1 & 2 + \frac{g_1 \cos 2\pi(2\alpha)}{\kappa} & -1 & \\ & -1 & \ddots & \ddots \\ & & \ddots & -1 \\ -1 & & -1 & 2 + \frac{g_1 \cos 2\pi(n\alpha)}{\kappa} \end{pmatrix} \triangleq K_0$$

且

$$K = K_0 + \frac{g_0}{\kappa} I$$

那么，如果  $K, K_0$  的本征值谱为  $\lambda, \lambda_0$ ，那么有  $\lambda = \lambda_0 + g_0/\kappa$ ，于是， $c = 2 - \lambda + g_0/\kappa = 2 - \lambda_0$ 。因此，为简便起见在之后的小问中，我们都将  $g_0$  置零，求出  $K_0$  的本征值谱，再用  $c = 2 - \lambda_0$  得到  $c$  值。

本文中， $\alpha = 1/2, \gamma = 2, n = 720, K_0$  如下

$$\begin{pmatrix} 0 & -1 & & & -1 \\ -1 & 4 & -1 & & \\ & -1 & 0 & -1 & \\ & & -1 & 4 & -1 \\ & & & -1 & 0 & -1 \\ & & & & -1 & \ddots & \ddots \\ & & & & & \ddots & -1 \\ -1 & & & & & & -1 & 4 \end{pmatrix}$$

求解源代码见附件。输出结果见”q2-N=720.txt” 第一列。

### 1.3 第三问

传输矩阵定义如下

$$\begin{pmatrix} x_3 \\ x_2 \end{pmatrix} = \begin{pmatrix} c + \gamma & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} c - \gamma & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_0 \end{pmatrix}$$

$$= \begin{pmatrix} c^2 - \gamma^2 - 1 & -c - \gamma \\ c - \gamma & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_0 \end{pmatrix}$$

于是

$$\begin{pmatrix} x_1 \\ x_0 \end{pmatrix} = \begin{pmatrix} x_{n+1} \\ x_n \end{pmatrix} = \begin{pmatrix} c^2 - \gamma^2 - 1 & -c - \gamma \\ c - \gamma & -1 \end{pmatrix}^{n/2} \begin{pmatrix} x_1 \\ x_0 \end{pmatrix}$$

这里需要 n 为偶数。

记传输矩阵的特征值  $\eta_{\pm}$

要使上式成立，必有  $\eta_{\pm} = \exp[\pm i4\pi k/n]$  于是有  $\eta_+ + \eta_- = 2 \cos \frac{4\pi k}{n}$ 。而根据传输矩阵  $\eta_+ + \eta_- = c^2 - \gamma^2 - 2$ ，联立得到

$$c^2 = \gamma^2 + 2(1 + \cos \frac{4\pi k}{n}) \quad (5)$$

由上式可到， $c^2 \in [\gamma^2, \gamma^2 + 4]$ ，则  $c \in [-\sqrt{\gamma^2 + 4}, -\gamma] \cup [\gamma, \sqrt{\gamma^2 + 4}]$

$\gamma > 0$ ，于是 c 分布在两个区间。

可在第二问的代码中增加一些输出，比对解析和数值结果。

同时输出特征值的平方，并输出  $\gamma^2 + 2(1 + \cos \frac{4\pi k}{n})$ ,  $k = 1, 2, \dots, n$ ，输出结果见”q2-N=720.txt” 第二、三列。

将结果复制到 excel 中，类似第一问的方法，列内排序、两列作差取绝对值、差的绝对值求和。得到

A	B	C	D
$n=720$			
数值结果	解析结果	差的 绝对值	差的 绝对值求 和
8.000000	8.000000	0.000000	0.000000
8.000000	8.000000	0.000000	
7.999695	7.999695	0.000000	
7.999695	7.999695	0.000000	
7.999695	7.999695	0.000000	

图 5: q3 解析数值比对

可见，解析结果与数值结果在六位小数精度下完全吻合。

原始 excel 表格处理结果见“dataAnalysis.xlsx” 中 sheet “q2”。

## 1.4 第四问

$\alpha = 1/3$  时

$$K_0 = \begin{pmatrix} 2 & -1 & & & & & -1 \\ -1 & 2 & -1 & & & & \\ & -1 & 4 & -1 & & & \\ & & -1 & 2 & -1 & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & \ddots & \ddots \\ & & & & & \ddots & \\ -1 & & & & & & -1 & 4 \end{pmatrix}$$

这里要求  $n$  是 3 的倍数。

$\alpha = 1/4$  时

$$K_0 = \begin{pmatrix} 2 & -1 & & & & & -1 \\ -1 & 0 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & -1 & 4 & -1 & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & \ddots & \ddots \\ & & & & & \ddots & \\ -1 & & & & & & -1 & 4 \end{pmatrix}$$

这里要求  $n$  是 4 的倍数。

本问编写了一个只需调制  $\alpha, N, r$  就可以给出本征值组的程序，具体见附录。

我们不妨还取  $n=720$ ,  $\gamma = 2$ , 数值计算的结果见“q4-a=0.33.txt”“q4-a=0.25.txt”。

将得到的数据降序排列画出如下图像

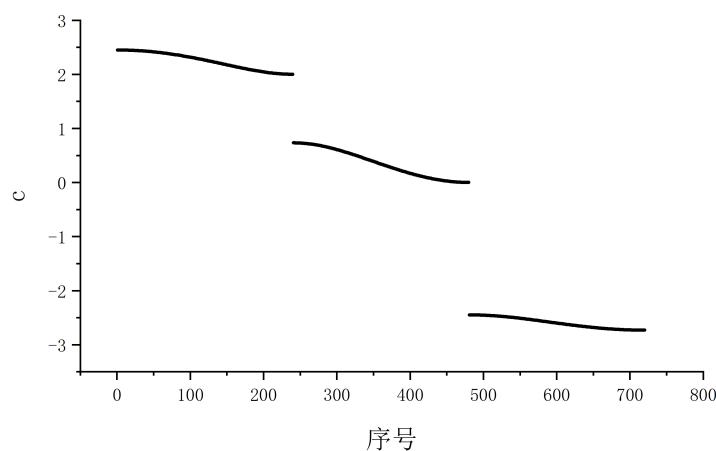


图 6: q4-a=0.33 特征值分布图

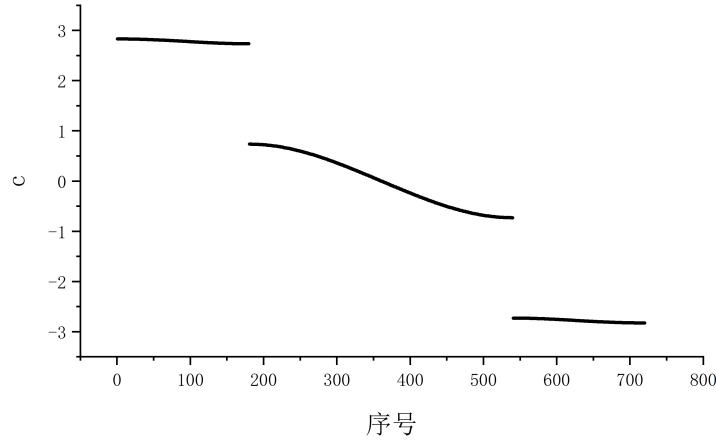


图 7: q4-a=0.25 特征值分布图

可以数值地给出

$$\alpha = 1/3, c \in [-2.732051, -2.449490] \cup [0, 0.732051] \cup [2, 2.449490] \quad (6)$$

$$\alpha = 1/4, c \in [-2.828427, -2.732051] \cup [-0.732051, 0, 732051] \cup [2.732051, 2.828427] \quad (7)$$

类似的，可以解析的给出 c 地取值范围

传输矩阵为

$$\alpha = 1/3 :$$

$$\begin{aligned} & \begin{pmatrix} c + \gamma & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} c - \frac{1}{2}\gamma & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} c - \frac{1}{2}\gamma & -1 \\ 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{4}(-8c + 4c^3 - 2\gamma - 3c\gamma^2 + \gamma^3) & 1 - (c - \gamma/2)(c + \gamma) \\ -1 + (c - \gamma/2)^2 & \frac{1}{2}(-2c + \gamma) \end{pmatrix} \end{aligned}$$

$$\alpha = 1/4 :$$

$$\begin{aligned} & \begin{pmatrix} c + \gamma & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} c & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} c - \gamma & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} c & -1 \\ 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 + c^4 - cr - c^2(3 + r^2) & c(2 - c^2 + r^2) \\ c(-2 + c^2 - cr) & 1 - c^2 + cr \end{pmatrix} \end{aligned}$$

类似第三问的推导过程，传输矩阵的特征值  $\eta_{\pm} = \exp[\pm i2\pi\alpha]$ ，再根据矩阵的性质，特征值之和为对角元之和，得到

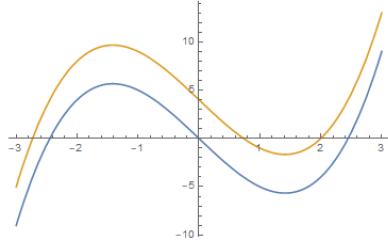
$\alpha = 1/3$  时，

$$c^3 + \frac{\gamma^3}{4} - \frac{3}{4}(\gamma^2 + 4)c = 2 \cos \frac{6\pi k}{n}$$

$\gamma = 2$  时，

$$c^3 - 6c + 2(1 - \cos \frac{6\pi k}{n}) = 0$$

于是  $c$  应取  $c^3 - 6c + h = 0, 0 \leq h \leq 4$  时的解。



画出示意图， $h$  从 0 变到 4 的时候，函数从蓝线上升到黄线。函数擦过的  $x$  轴就是  $c$  可以取到的值。于是解方程  $c^3 - 6c = 0$  得到

$$c_1 = -\sqrt{6}, c_2 = 0, c_3 = \sqrt{6}$$

解方程  $c^3 - 6c + 4 = 0$  得到

$$c_1 = 2, c_2 = -1 - \sqrt{3}, c_3 = -1 + \sqrt{3}$$

得到  $c \in [-1 - \sqrt{3}, -\sqrt{6}] \cup [0, -1 + \sqrt{3}] \cup [2, \sqrt{6}]$

$\alpha = 1/4$  时，

$$c^4 - c^2(4 + \gamma^2) + 2(1 - \cos \frac{8\pi k}{n}) = 0$$

于是

$$c_{\pm}^2 = \frac{4 + \gamma^2 \pm \sqrt{(4 + \gamma^2)^2 - 8(1 - \cos \frac{8\pi k}{n})}}{2}$$

即

$$c^2 \in [0, \frac{4 + \gamma^2 - \sqrt{(4 + \gamma^2)^2 - 16}}{2}] \cup [\frac{4 + \gamma^2 + \sqrt{(4 + \gamma^2)^2 - 16}}{2}, 4 + \gamma^2]$$

于是

$$\begin{aligned} c &\in [-c_1, -c_2] \cup [-c_3, c_3] \cup [c_2, c_1] \\ c_1 &= \sqrt{4 + \gamma^2} \\ c_2 &= \sqrt{\frac{4 + \gamma^2 + \sqrt{(4 + \gamma^2)^2 - 16}}{2}} \\ c_3 &= \sqrt{\frac{4 + \gamma^2 - \sqrt{(4 + \gamma^2)^2 - 16}}{2}} \end{aligned}$$

当  $\gamma = 2$  有

$$c \in [-2\sqrt{2}, -1 - \sqrt{3}] \cup [1 - \sqrt{3}, -1 + \sqrt{3}] \cup [1 + \sqrt{3}, 2\sqrt{2}]$$

综上

$$\alpha = 1/3, c \in [-1 - \sqrt{3}, -\sqrt{6}] \cup [0, -1 + \sqrt{3}] \cup [2, \sqrt{6}] \quad (8)$$

$$\alpha = 1/4, c \in [-2\sqrt{2}, -1 - \sqrt{3}] \cup [1 - \sqrt{3}, -1 + \sqrt{3}] \cup [1 + \sqrt{3}, 2\sqrt{2}] \quad (9)$$

可以验证，式 (6) 和 (8)、(7) 和 (9) 吻合得很好。

值得特别说明的是，根据第五问的结论， $\alpha = 1/4$  时特征值区间将劈裂成 4 份，但这里数值和解析结果都指出，特征值区间劈裂成 3 份。注意到特征值中央区间关于原点对称分布，如果把原点两侧的部分看成两个区间，那么特征值劈裂成了 4 份。这里猜测，第五问的结论是在把关于原点对称的区间看成 2 个意义上成立的。

## 1.5 第五问

利用第四问代码，只需设置参数，即可完成本问计算，取

$$n = 720, \gamma = 2, \alpha = 1/5, 1/6, 1/7$$

得到数据见“q5-a=0.2.txt”“q5-a=0.166.txt”“q5-a=0.142.txt”。

将数据降序排序后获得如下特征值分布图

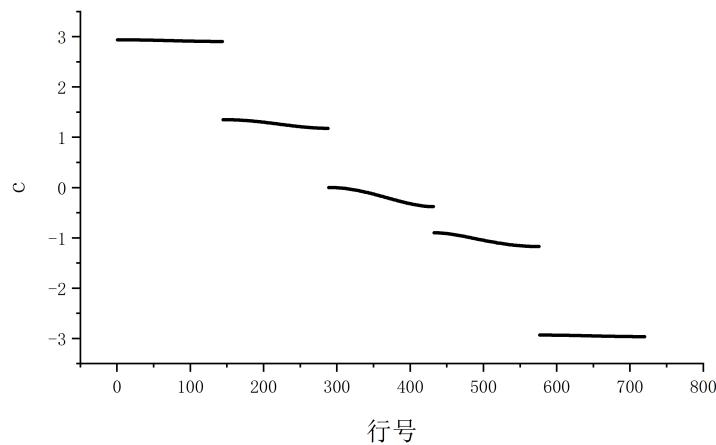


图 8: q5-a=0.2 特征值分布图

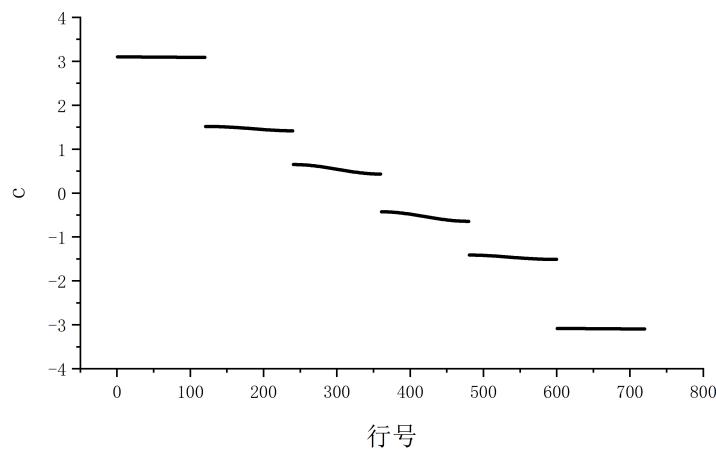


图 9: q5-a=0.166 特征值分布图

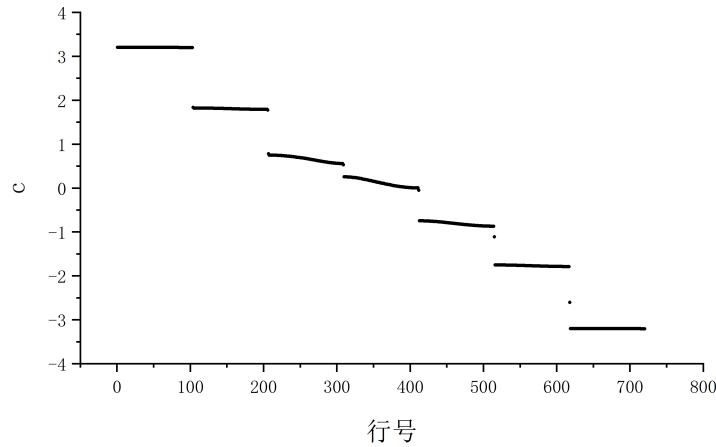


图 10:  $q_5-a=0.142$  特征值分布图

观察到  $\alpha = 1/5, 1/6$  时特征值区间都被很好的劈裂成  $1/\alpha$  份, 但  $\alpha = 1/7$  时特征值存在一些孤立散点。猜测可能和  $n=720$  不是 7 的倍数有关。改变条件

$$n = 770, \gamma = 2, \alpha = 1/7$$

得到数据见”q5-a=0.142-N=770.txt”, 降序排列绘成下图

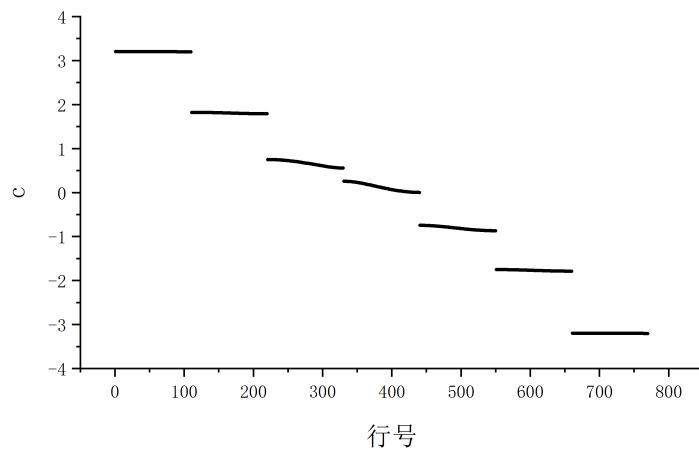


图 11:  $q_5-a=0.142-N=770$  特征值分布图

可以看出, 散点消失, 特征值区间又成了完美的  $1/\alpha$  份。这体现了, 想要观察到完美的区间劈裂,  $n$  需要是  $1/\alpha$  的倍数。

另外, 题目中的结论是特征值区间劈裂与  $\alpha$  分子无关, 下面作出验证。

取

$$n = 770, \gamma = 2, \alpha = 1/5, 2/5, 3/5, 4/5$$

得到数据见”q5-a=0.2.txt”, ”q5-a=0.4.txt”, ”q5-a=0.6.txt”, ”q5-a=0.8.txt”

作出  $c - \alpha$  图, 得到

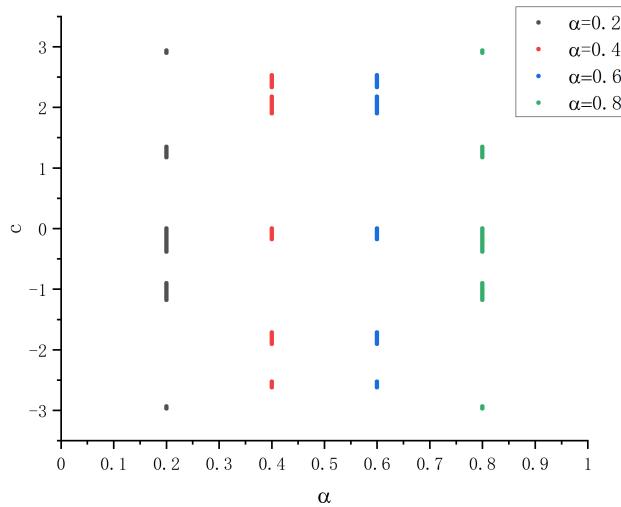


图 12: q5 验证特征值劈裂份数与 a 分子无关

可见, 不同的  $\alpha$  值虽然特征值取值有变化, 但是劈裂成的区间数都是一样的。

## 1.6 第六问

可以对第四问的代码稍加修改, 每取一个  $\alpha$ , 计算一组特征值并输出, 具体代码实现见附录。这里取

$$N = 720, \gamma = 2, \alpha = 0.001, 0.002, \dots, 0.999$$

得到数据见”q6.txt”。

绘制  $c - \alpha$  图得到 Hofstadter 蝴蝶。

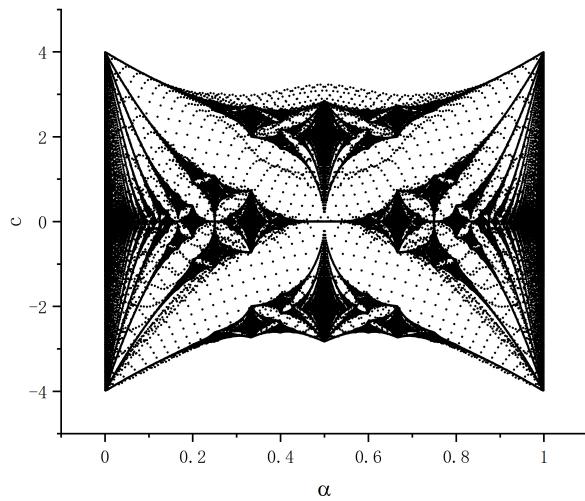


图 13: Hofstadter 蝴蝶

## 2 附录

### 2.1 “matrix.h” 头文件

```
1 #ifndef __MATRIX_H__
2 #define __MATRIX_H__
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<math.h>
7
8 #ifndef EPS
9 #define EPS 1e-7
10#endif
11
12#ifndef TOL
13#define TOL 1e-10
14#endif
15
16#ifndef PI
17#define PI 3.1415926535897932
18#endif
19
20typedef struct SqMatrix{
21    int n;
22    double** el;
23} SqMatrix;
24
25int sgn(double x){ //符号函数
26    if(x>=0) return 1;
27    return -1;
28}
29void printMatrix(SqMatrix* A){
30    int i, j;
31    for (i = 0; i < A->n; i++)
32    {
33        for (j = 0; j < A->n; j++)
34        {
35            printf("%lf", A->el[i][j]);
36            if(j!=A->n-1) printf("\t");
37            else printf("\n");
38        }
39    }
40}
41double** createNew2DArray(int m, int n){
42    int i, j;
43    double **a=(double **)malloc(m*sizeof(double *));
44    for(i=0;i<m;i++)
45    {
46        a[i]=(double *)malloc(n*sizeof(double));
47        for(j=0;j<n;j++)
48        {
49            a[i][j]=0;
50        }
51    }
52    return a;
53}
```

```

44     for ( i = 0; i < m; i++)
45     {
46         a[ i]=(double *)malloc( n*sizeof( double) );
47     }
48     for ( i = 0; i < m; i++)
49     {
50         for ( j = 0; j < n; j++)
51         {
52             a[ i ][ j]=0;
53         }
54     }
55     return a;
56 }
57 SqMatrix* createZeroSqMatrix( int n){
58     SqMatrix* p=(SqMatrix*)malloc( sizeof( SqMatrix) );
59     p->n=n;
60     p->el=createNew2DArray( n,n );
61     return p;
62 }
63 int is_sym_sq( SqMatrix* A){ //判断方阵是否对称
64     if (A->el==NULL) return 1;
65     int i ,j ;
66     for ( i = 0; i < A->n; i++)
67     {
68         for ( j = i+1; j < A->n; j++)
69         {
70             if (A->el[ i ][ j ]!=A->el[ j ][ i ]) return 0;
71         }
72     }
73     return 1;
74 }
75
76 double* eigenvalue_sym_sq( SqMatrix* A){ //求出实对称方阵的特征值组
77     if (!is_sym_sq(A) || A->n<=0) return NULL;
78     //Householder变换化为三对角阵
79     double *a1=(double *)malloc( (A->n+1)*sizeof( double) );
80     double *b=(double *)malloc( (A->n+1)*sizeof( double) );
81     double *au=(double *)malloc( (A->n+1)*sizeof( double) );
82     double *d=(double *)malloc( A->n*sizeof( double) );
83     if (A->n==1){
84         b[1]=A->el[ 0 ][ 0 ]; return b;
85     }
86     if (A->n==2){
87         double b1=A->el[ 0 ][ 0 ],b2=A->el[ 1 ][ 1 ],a=A->el[ 0 ][ 1 ];
88         b[1]=(b1+b2+sqrt(pow((b1-b2),2)+4*a*a))/2;
89         b[2]=(b1+b2-sqrt(pow((b1-b2),2)+4*a*a))/2;
90         return b;

```

```

91 }
92 int i,j,k;
93 //printf("entered Householder\n");
94 for ( i = 0; i < A->n-2; i++)
95 {
96     for ( j = i+2; j < A->n; j++)
97     {
98         if(fabs(A->el[j][i])>EPS) {
99             //printf("A(%d,%d)=%.15lf      ",j,i,A->el[j][i]);
100            goto Householder;
101        }
102    }
103    continue;
104
105 Householder:
106 {
107     //printf("start a householder transformation : %d\n",i);
108     double sigma=0,beta,uAu;
109     for ( j = i+1; j < A->n; j++)
110     {
111         sigma+=A->el[j][i]*A->el[j][i];
112     }
113     sigma=sqrt(sigma);
114     sigma*=sgn(A->el[i+1][i]);
115     beta=sigma*(sigma+A->el[i+1][i]);
116     A->el[i+1][i]+=sigma;//更新u
117     //计算Au, 储存在uT中
118     for ( j = 1; j < A->n-i; j++)
119     {
120         A->el[i][i+j]=0;
121         for ( k = 1; k < A->n-i; k++)
122         {
123             A->el[i][i+j]+=A->el[i+j][i+k]*A->el[i+k][i];
124         }
125     }
126     //计算uTAu
127     uAu=0;
128     for ( j = 1; j < A->n-i; j++)
129     {
130         uAu+=A->el[i+j][i]*A->el[i][i+j];
131     }
132     //更新A
133     for ( j = 1; j < A->n-i; j++)
134     {
135         for ( k = 1; k < A->n-i; k++)
136         {
137             A->el[i+j][i+k]-=1/beta*(A->el[i+j][i]*A->el[i][i+k]+A->el[i][i+k]-beta*A->el[i+j][i+k]);
138         }
139     }
140 }

```

```

                j ]*A->el [ i+k ] [ i ]) ;
138         A->el [ i+j ] [ i+k ]+=uAu/( beta*beta )*A->el [ i+j ] [ i ]*A->el [ i+k ] [ i ];
139     }
140 }
141 A->el [ i+1 ][ i ]=A->el [ i ][ i+1 ]=-sigma;
142 for ( j = 2; j < A->n-i ; j++)
143 {
144     A->el [ i+j ] [ i ]=A->el [ i ] [ i+j ]=0;
145 }
146 //printf( "%.20lf %.20lf\n ", sigma , beta );
147 }
148 }
149 b[1]=A->el [ 0 ][ 0 ];
150 for ( i = 2; i <= A->n; i++)
151 {
152     al [ i ]=A->el [ i-1 ][ i-2 ];
153     b [ i ]=A->el [ i-1 ][ i-1 ];
154     au [ i-1 ]=A->el [ i-2 ][ i-1 ];
155 }
156 //QR迭代求特征值
157 int n=A->n;
158 double* cosine=(double *) malloc (n*sizeof(double));
159 double* sine=(double *) malloc (n*sizeof(double));
160 while (n>=2){
161     double s=b[n];
162     double temp1,temp2;
163     b[1]-=s;
164     for ( k = 1; k < n; k++)
165     {
166         b[k+1]-=s;
167         double r=sqrt (b[k]*b[k]+al[k+1]*al[k+1]);
168         cosine [ k ]=b [ k ] / r ; sine [ k ]=al [ k+1 ] / r ;
169         //左变换
170         b [ k ]=r ; al [ k+1 ]=0;
171         temp1=au [ k ] ,temp2=b [ k+1 ];
172         au [ k ]=temp1*cosine [ k ]+temp2*sine [ k ];
173         b [ k+1 ]=-temp1*sine [ k ]+temp2*cosine [ k ];
174         if (k!=n-1){d [ k ]=au [ k+1 ]*sine [ k ]; au [ k+1 ]*=cosine [ k ];}
175     }
176     for ( k = 1; k < n; k++)
177     {
178         if (k!=1){
179             temp1=au [ k-1 ];temp2=d [ k-1 ];
180             au [ k-1 ]=temp1*cosine [ k ]+temp2*sine [ k ];
181             d [ k-1 ]=-temp1*sine [ k ]+temp2*cosine [ k ];
182         }
183         temp1=b [ k ] ;temp2=au [ k ];

```

```

184         b[k]=temp1*cosine [k]+temp2*sine [k];
185         au[k]=temp1*(-sine [k])+temp2*cosine [k];
186         temp1=al [k+1];temp2=b[k+1];
187         al [k+1]=temp1*cosine [k]+temp2*sine [k];
188         b[k+1]=temp1*(-sine [k])+temp2*cosine [k];
189         if(k!=n-1)al [k+2]*=cosine [k];
190
191         b[k]+=s;
192     }
193     b[n]+=s;
194     if(fabs (al [n])<EPS){
195         al [n]=0;
196         //printf("%d\n",n);
197         n--;
198     }
199 }
200 return b;
201 }
202
203 #endif

```

## 2.2 第一问源代码

```

1 //本程序求解第一问，需要手动设置N的取值
2 #include "matrix.h"
3 #define N 1024
4 int main(){
5     FILE *fdata=fopen( "C:/ C_files/HW3/q1-N=1024.txt ", "w");
6     SqMatrix* A;
7     A=createZeroSqMatrix(N);
8     //构建矩阵
9     A->el[A->n-1][0]--;A->el[0][A->n-1]--;A->el[0][0]=2;
10    int i;
11    for ( i = 1; i < A->n; i++)
12    {
13        A->el[ i ][ i ]=2;
14        A->el[ i -1][ i ]--;
15        A->el[ i ][ i -1]--;
16    }
17    //求特征值
18    double* test=eigenvalue_sym_sq(A);
19    if(test==NULL) return 0;
20    for ( i = 1; i <= A->n; i++)
21    {
22        fprintf(fdata,"%lf %lf\n",test[ i ],2*(1-cos(2*PI*i/N)));
23    }

```

```

24
25     return 0;
26 }
```

### 2.3 第二、三问源代码

```

1 //本程序求解第二问，并同时完成第三问中解析与数值的比对验算
2 #include "matrix.h"
3 double a=1./2;
4 double r=2;
5 int n=720;//n>=2
6
7 int main() {
8     FILE *fdata=fopen( "C:/C_files/HW3/q2-N=720.txt" , "w" );
9     SqMatrix* A;
10    A=createZeroSqMatrix(n);
11    A->el[A->n-1][0]=-1;A->el[0][A->n-1]=-1;
12    A->el[0][0]=2-r;
13    int i;
14    for ( i = 1; i < A->n ; i++)
15    {
16        A->el[i-1][i]=A->el[i][i-1]=-1;
17        A->el[i][i]=2+r*cos(2*PI*(i+1)*a);
18    }
19    double *test=eigenvalue_sym_sq(A);
20    if (test==NULL) return 0;
21    for ( i=1;i<=A->n; i++){
22        fprintf(fdata , "%lf %lf %lf\n",2-test[i],(2-test[i])*(2-test[i]) ,r*r
23            +2*(1+cos(4*PI*i/n)));
24    }
25 }
```

### 2.4 第四、五问源代码

```

1 #include "matrix.h"
2 \\\ 本程序求解第4, 5问；需要手动调置a, r, n;
3 double a=1./4;
4 double r=-2;
5 int n=720;//n>=2
6
7 int main() {
8     FILE *fdata=fopen( "C:/C_files/HW3/q4-explore-r=-2.txt" , "w" );
9     SqMatrix* A;
10    A=createZeroSqMatrix(n);
11    A->el[A->n-1][0]=-1;A->el[0][A->n-1]=-1;
```

```

12     A->el[0][0]=2+r*cos(2*PI*a);
13     int i;
14     for ( i = 1; i < A->n ; i++)
15     {
16         A->el[i-1][i]=A->el[i][i-1]=-1;
17         A->el[i][i]=2+r*cos(2*PI*(i+1)*a);
18     }
19     double *test=eigenvalue_sym_sq(A);
20     if (test==NULL) return 0;
21     for ( i=1;i<=A->n; i++){
22         fprintf(fdata , "%lf\n",2-test[i]);
23     }
24     return 0;
25 }
```

## 2.5 第六问源代码

```

1 #include "matrix.h"
2 #define ITV 1e-3//a的取值步长
3 #define N 720
4 double a=ITV;
5 double r=2;
6
7 int main(){
8     FILE *fdata=fopen( "C:/C_files/HW3/q6.txt" , "w");
9     for (a=ITV;a<1;a+=ITV)
10    {
11        SqMatrix* A;
12        A=createZeroSqMatrix(N);
13        A->el[A->n-1][0]=-1;A->el[0][A->n-1]=-1;
14        A->el[0][0]=2+r*cos(2*PI*a);
15        int i;
16        for ( i = 1; i < A->n ; i++)
17        {
18            A->el[i-1][i]=A->el[i][i-1]=-1;
19            A->el[i][i]=2+r*cos(2*PI*(i+1)*a);
20        }
21        double *test=eigenvalue_sym_sq(A);
22        if (test==NULL) return 0;
23        for ( i=1;i<=A->n; i++){
24            fprintf(fdata , "%lf %lf\n",a,test[i]);
25        }
26        printf( "a=%lf finished ... \n",a);
27    }
28    fclose(fdata);
29    return 0;
```

## 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