# 计算物理学作业 3

朱寅杰 1600017721

## 3.1 Householder 与 Givens 在 QR 分解中的比较

首先我使用代码实现了这两个算法，源文件分别是Householder.py和Givens.py。

如果使用 Householder 反射法来对一个 $N \times N$ 方阵进行 QR 分解，总共需要乘 $(N-1)$ 次反射矩阵。在第 $k+1$ 次反射时，需要算两次一个长度为 $N-k$ 的矢量的模长（每次计算量为 $2N-2k$），并作一次归一化（$N-k$ 次除法）。之后的反射，需要对 $(N-k-1)^2$ 个元素进行操作，每个元素算一次点乘和一次旋转，同时把计算 $Q$ 和 $R$ 的工作都算上，要 10 次计算（可以从我的代码里数出来）。所以总共需要的领头阶计算次数就是 $10N^2-20N^k+10k^2$。对 $k$ 求和，得到总的计算次数的领头阶数是 $5/3N^3$。

如果使用 Givens 转动的话，计算的次数会相对少一些。总共有 $N(N-1)/2$ 个下三角元要被归零，每个元 $a_{ij}$ 要被归零，涉及到的只是下标出现 $i$、$j$ 的合计四个元素。设这其中涉及到 $g$ 步计算（这是一个与 $N$ 无关的常数），那么总共的计算次数就是 $g \times N(N-1)/2 \approx gN^2/2$。从我的程序中数运算符，得到 $g=33$，故总的计算次数约为 $16.5N^2$。

对代码进行实际测验，Householder 跑 $N=6,12,18$ 跑 10000 次分别用时 1.36s、6.38s、18.5s，Givens 跑 $N=6,12,18$ 跑 10000 次分别用时约 0.58s、2.33s、5.35s。分别约摸是 $N^{2.4}$ 和 $N^{2.0}$。和理论粗估相差还是蛮大的，一方面和机器的优化等因素有关，另一方面不同的运算的时间也是不一样的，其他如内存存取的时间也没有细细考虑在内。还是请教一个信科的同学让他帮忙估计一下比较靠谱啊。

## 3.2 幂次法求矩阵最大模的本征值和本征矢

对于带有周期性边界条件的微分方程组

$$\ddot{x_i}(t) = x_{i-1}(t) + x_{i+1}(t) - 2x_i(t), x_{i+N}(t) = x_i(t), i = 1,2,...,N$$

我们求出其（作为解的基底的）简正模式 $x_i(t) = x_i \exp(-\mathrm{i}\omega t)$，其中 $x_i \in \mathbb{C}$，则有

$$-x_{i-1} + 2x_i - x_{i+1} = \omega^2 x_i, i = 1,2,...,N$$

于是对 $x_i$ 与 $\omega$ 的求解就化为了对一个三对角矩阵 $(A_{ij})$ 的特征值与特征向量的求解，其中 $A_{ij}$ 的具体形式为

$$A_{ij} = 2\delta_{ij} - \delta_{(i+1) \bmod N, j} - \delta_{(i-1) \bmod N, j},\ \ i,j = 1,2,...,N$$

我们接下来使用 Power iteration 的办法求解其最大的特征值与特征向量。Power iteration 具体是说，对于任意一个可对角化且本征值（的模）非简并的系统 $A$，设其本征值（按模长排列）为 $\lambda_1 > \lambda_2 > ...$，对应的本征矢量构成一组正交归一的基矢 $v_1, v_2, ....$。现在在空间中任取一个单位矢量 $a_0 = \sum q_i v_i$，其中 $q_i \in \mathbb{C}$，则有 $Aa_0 = \sum \lambda_i q_i v_i$。现在我们构造一个迭代：

$$a_0 = \sum q_i v_i, a_{i+1} = \frac{Aa_i}{\left|Aa_i\right|}, i = 0,1,2,...$$

容易知道

$$a_n = \frac{\sum \lambda_i^n q_i v_i}{\left|\sum \lambda_i^n q_i v_i\right|} = \left(\frac{\lambda_1}{|\lambda_1|}\right)^n \frac{q_1 v_1 + \sum_{i=2} (\lambda_i/\lambda_1)^n q_i v_i}{\left|q_1 v_1 + \sum_{i=2} (\lambda_i/\lambda_1)^n q_i v_i\right|}$$

由于 $|\lambda_1| > |\lambda_2| > ...$，因此当 $n \to \infty$ 时，$a_n$ 中除了 $v_1$ 上分量外，其他 $v_2, v_3, ...$ 上的分量均以指数速度趋于零。故而有

$$\lim_{n\to\infty} a_n = \lim_{n\to\infty} [\left(\frac{\lambda_1}{|\lambda_1|}\right)^n \frac{q_1}{|q_1|} v_1)]$$

忽视掉那个相位因子，我们可以说 $a_\infty = \lim_{n\to\infty} a_n = v_1$，因此我们从任意的矢量开始迭代，最终都能收敛到本征矢 $v_1$（最多相差一个 $U(1)$）。再计算 $a_\infty^\dagger A a_\infty = v_1^\dagger A v_1 = \lambda_1$，我们获得了系统最大的本征值。

具体到题中这个一维环形原子链的例子，用上面的迭代算法进行计算（源代码在power_iteration.py），得到的最大本征值为4（双精度范围内为整数），对应的单位本征矢量为 [0.3162277660168382, -0.3162277660168381, 0.316227766016838, -0.3162277660168379, 0.3162277660168378, -0.3162277660168377, 0.3162277660168377, -0.31622776601683783, 0.31622776601683805, -0.31622776601683816]，即近似为 $[1, -1, 1, -1, 1, -1, 1, -1, 1, -1]/\sqrt{10}$。

### 3.3 关联函数的拟合与数据分析

这道题的所有源代码见cor.py。首先要对数据进行预处理（题目中叫做“对称化”）。预处理之后我们获得了函数 $C(t), t = 0, 1, ..., 32$ 在 200 次测量下的行为 $C_i(t), i = 0, ..., 199$。我们首先通过样本平均估计 $C(t)$ 的中心值，并通过样本方差的平方根来估计每个时间片上 $C(t)$ 的误差。从 $t = 0$ 到 $t = 32$ 各点的百分误差分别为:

| $t$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 百分误差 | 1.58 | 2.36 | 5.64 | 8.30 | 10.37 | 11.91 | 13.12 | 14.07 | 14.85 |
| $t$ | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 百分误差 | 15.20 | 15.39 | 16.01 | 16.56 | 17.15 | 17.79 | 18.29 | 18.88 | 19.43 |
| $t$ | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 百分误差 | 19.87 | 20.10 | 20.30 | 20.57 | 21.04 | 21.56 | 22.09 | 22.42 | 22.67 |
| $t$ | 27 | 28 | 29 | 30 | 31 | 32 | | | |
| 百分误差 | 22.90 | 23.17 | 23.48 | 23.71 | 24.17 | 24.48 | | | |

然后是分别通过两种方法算有效质量。通过直接做比值的办法，取 $m_{eff} = \log \frac{C(t)}{C(t+1)}$，按每自由度卡方最优拟合出来的平台位置在 [24,27] 处，自由度为 3，$\chi^2/d.o.f = 0.1093$，查表得相应的 $p$ 值为 0.0453。拟合出的粒子质量为1.157 124±0.000 105（误差按 $1\sigma$ 计，是从输出结果里面截取数据然后拿计算器解的二次方程）。

通过更加精确的 $m_eff = \cosh^{-1}(\frac{C(t+1)+C(t-1)}{2C(t)})$ 的办法，按每自由度卡方最优拟合出来的平台位置在 [25,29] 处，自由度数为 4，$\chi^2/d.o.f = 0.08979$，查表得相应的 $p$ 值为 0.0143。拟合出的粒子质量为1.157 100±0.000 109（误差也是敲计算器解的方程）

然后算的是两个相关系数咯。$\rho_{3,4} = 0.957\,06 \pm 0.006\,31$，$\rho_{3,5} = 0.901\,01 \pm 0.016\,21$，

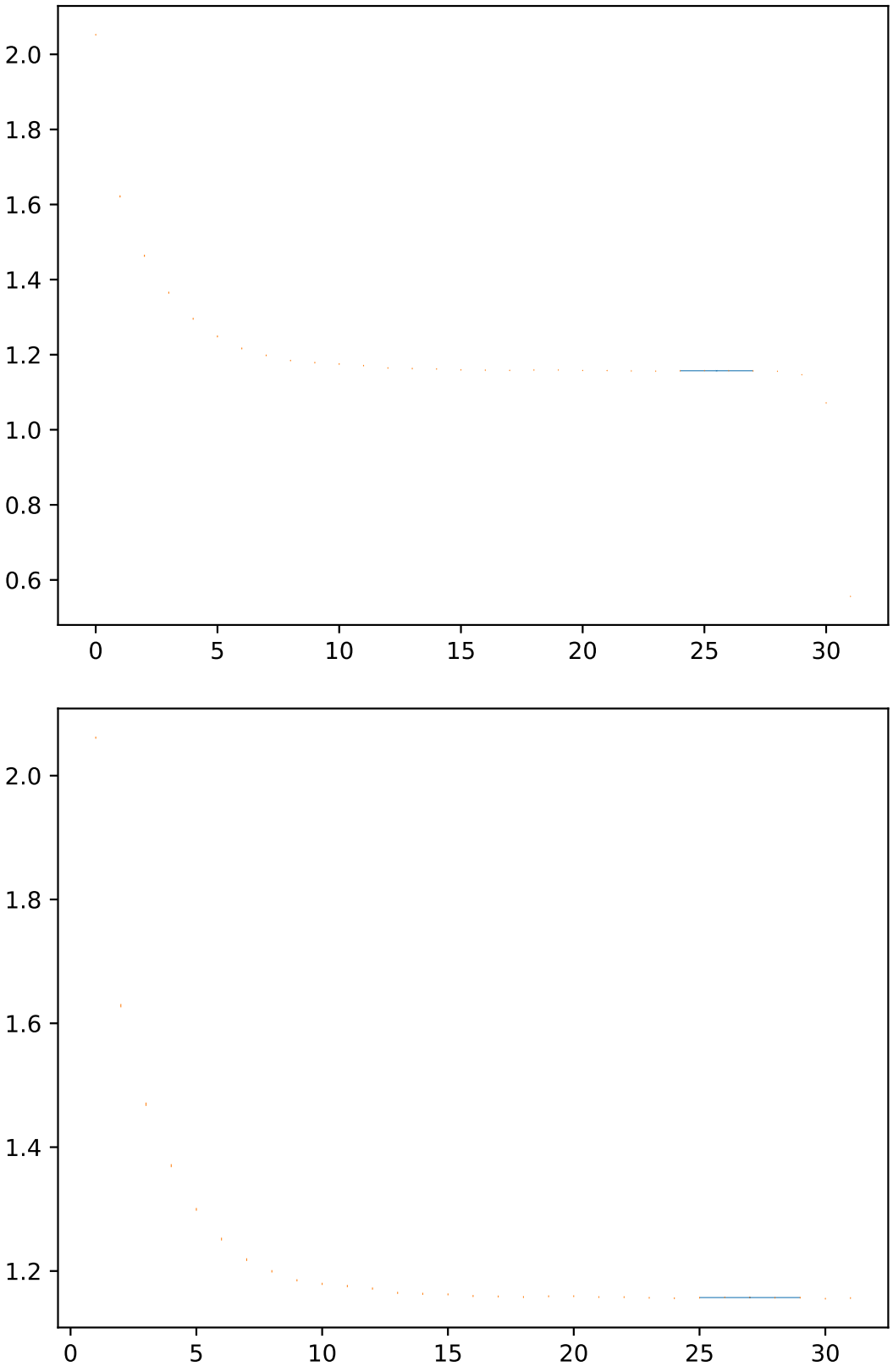


图 1: 横轴是 $t$，纵轴是 $m_{eff}$。如果你发现什么都看不见的话，那就对了，因为 error 实在是太小了，以至于我都不能把数据点画大，不然就都盖住什么都看不见了。上图是用比值法算的 $m_{eff}$ 及其误差，下图是用 arccosh 法算的。卡方拟合的区间和 error bar 也画在图里了（不对，似乎 error bar 也被盖住了。不管它，反正我数字算出来了）。如果你看不见的话，zoom in，zoom in，再 zoom in（我是矢量图我无所畏惧.eps）。如果还是嫌 error bar 看不清的话，就跑一下程序cor.py，看一看输出结果吧。各个点的位置和误差都在上面了。