计算物理学作业3

许嘉琪 1500011417

January 1, 2018

1 QR分解

1.1 计算次数

1.1.1 Householder

首先说明我的Householder方法的流程:

1. 对于矩阵A每列的对角线及以下的部分，将这一部分作为一个列矢量x，计算其模长norm，令矢量y为经过反射变换希望得到的矢量，其形式为$(norm, 0, 0, ..., 0)^T$。计算一个单位矢量$u = \frac{x-y}{||x-y||_2}$，它的方向为$x-y$的方向。(这一步的$Q_i$可以由这个u算出：$Q_i = I - uu^T$)。考虑到norm已经计算出来了在计算$||x-y||$的时候可以使用$||x-y|| = \sqrt{2norm*(norm-a_{ii})}$，利用norm简化计算。

2. 用此时得到的Q_i作用在A上，记$Q_iA = Q_i[C_1, C_2, ..., C_n]$更新后面几列的值。然而经过推导，将其用u来表示：$Q_iC_j = C_j - 2uu^TC_j = C_j - 2 < C_j, u > u$。

3. 重复这个步骤，进行第i列的时候，u的长度就会减短为n-i。

考虑到以上的情况，对于一个大小为N×N的矩阵，当我们进行第 i 列的操作时(i=0,1,...,N-1)，上述中的第 1 步，需要计算长度为N − i 的矢量模长，计算量约为 2 ×(N-i)。上述中的第 2 步，需要计算$C_j - 2 < C_j, u > u$(j=i,i+1,...,N-1)。计算量为$(N-i)^2$，这里产生的就是领头项了。因此，领头项应该为$\sum_{i=0}^{N-2}(N-i)^2 = \frac{2N^3}{3}$ 当然，这里并没有把计算Q的步骤：$Q_i = I - uu^T$，$Q = Q_0Q_1...Q_{N-2}$的计算量计入。

1.1.2 Givens

Givens转动的步骤更加明确，每次计算一个 2 ⋆ 2 矩阵的转动，领头项由矩阵矩阵乘法R_iA中给出，计算次数为$float(GT, R) = \sum_{i=0}^{N-2}\sum_{j=i+1}^{N-1}\sum_{k=i}^{N-1} 6 = 2N^3 - 2N$ 综上:

1.2 Householder 变换实现Q R分解

按照1 .1.1中给出的步骤，我写了如下几个函数:

method	Householder	Givens
Complexity	$\frac{2}{3}N^3$	$2N^3$

Table 1: Complexity of QR

1. take_out_col(int c,int u) 它从A中将第c列从u第个之下的元素提取出来，返回用他们组成的列矢量

2. householer() 它计算一个列矢量所对应的u，并返回它。

3. update(matrix u) 它利用参数u，来更新原矩阵A。效果等同于用Q作用上去

4. getQ(matrix *u) 利用$Q_i = I - uu^T$计算Q

5. QS_h(string s) 最终的Q S分解函数，s传入"householder"即代表使用的方法。函数流程如下：

```
matrix *matrix::QS_h(string s)
{
        //using reflectors(householer method)to do QS decompose
        matrix *u=new matrix[row-1];
        for(int i=0;i<row-1;i++){
                u[i]=this->take_out_col(i,i);
                u[i]=*(u[i].householer());
                this->update(u[i]);
                this->getQ(u);
        }
}
```

1.3 Givens变换实现Q R分解

我下了对应的几个函数：

1. getG(int n, int i, int j, mytype a, mytype b) 它计算将矢量(a,b)旋转至(r,0)应该使用的旋转矩阵$G_{i,j,\theta}$,需要注意的是，G的两个元素需要总是处于对角线上的，而且，在计算$sin\theta$是需要手动根据a,b来调整正负号。

2. QS_h(string s) 最终的Q S分解函数，s传入"givens"即代表使用的方法。函数流程如下：

```
matrix *matrix::QS_h(string s)
{
        for(int i=0;i<col-1;++i){
                for(int j=i+1;j<row;++j){
                        matrix G=getG(row,i,j,ele[i*row+i],ele[j*row+i]);
```

```
                *this=(G^(*this));
                //this->print();
            }
        }
    }
```

1.4 实际测试

1.4.1 运行结果展示

随机产生矩阵。这里使用的就是我写的matrix类。首先介绍一下matrix类所提供的构造方式和运算。一共有 4 个常用的构造方式:

1. matrix(int n,int m) 随机生产n*m的矩阵，其元素值为-1到 1 间。
2. matrix(int n,int m,mytype k) 生成n*m的矩阵，其元素值为 k 间。
3. matrix(int n,int m, string s) 若s传入"identity",则生成单位矩阵。若传入"string"则生成下一题中需要求解的矩阵 A 。
4. matrix(int n, int m, mytype *e) 按照数组 e 中的数值生成矩阵， e 中按行依次排列。

我们用第一种构造方式生成 6 ∗ 6 的随机矩阵，喂给上面的程序。分解结果如下:

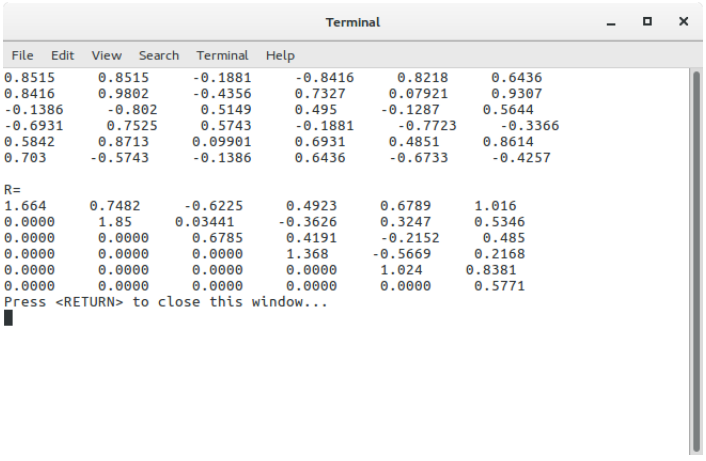

Figure 1: A and the resulting R

$\bar{t}_{Householder}/ms$	0.225
$\bar{t}_{Givens}/ms$	0.247

Table 2: Average time cost

1.4.2 时间比较

在 c ++中使用clock_t clock()计时，可以获得精确到1/CLOCKS_PER_SEC的时间。结果见Table 2。可以看到两者的差距不大，Householder方法稍快。

2 幂次法求最大本征值

2.1 本征方程

将解的形式$x = xe^{-i\omega t}$代入，可以得出：$\ddot{x}_i = x_{i-1} + x_{i+1} - 2x_i$。因此， x 满足的本征方程为:

$$(\delta_{i-1,j} + \delta_{i+1,j} - 2\delta_{i,j})x = -\omega^2 x$$

2.2 幂次法

2.2.1 证明：幂次法迭代最终会获得相应的本征值和本征矢

假设 A 有 n 个线性无关的特征向量，对应的特征值按照模的大小排列为：λ_1,λ_2,...,λ_n相应的单位特征向量为：v_1,v_2,...,v_n。这样考虑上述的迭代：$x^k = Ax^{k-1} = A^2x_{k-2} = A^k x_0$。由于本征矢量构成了完备的基，将向量 x 按本征矢量展开，$x_0 = a_1v_1 + a_2v_2 + ... + a_nv_n$，其中一定有$a_1 \neq 0$。于是，将这个式子带入迭代式可以得到：$x_k = a_1A^kv_1 + a_2A^kv_2 + ... + a_nA^kv_n = a_1r_1^kv_1 + a_2r_2^kv_2 + ... + a_nr_n^kv_n$。可以进一步改写为:

$$x_k = r_1^k(a_1v_1 + a_2(\frac{r_2}{r_1})^k v_2 + ... + a_n(\frac{r_n}{r_1})v_n)$$

可以看出，经过迭代，x_k的方向将逐渐变得与v_1一致。而迭代稳定的条件正是：$\lim_{k->\infty} x_k = v_1$

2.2.2 实现

我写了getEigenvalue()函数来实现获取本征值和对应本征矢量的功能。按照题中给出的方法，迭代40次之后，特征值就已经趋于稳定$\omega^2 = 4.0$，本征矢量如下图所示(下页):

3 拟合与数据分析

3.1 对称化

对于这个问题，我主要是在dataloader.h头文件之中处理的。首先将所有的64*200个实部读进来，每64个对称的两两平均，得到$\bar{C}(t)$。而这个量的误差利用$\Delta C(t) = \sqrt{\frac{1}{N(N-1)}\sum_{i=1}^{N}(C^{(i)}(t) - \overline{C}(t))^2}$来计算。结果如下:

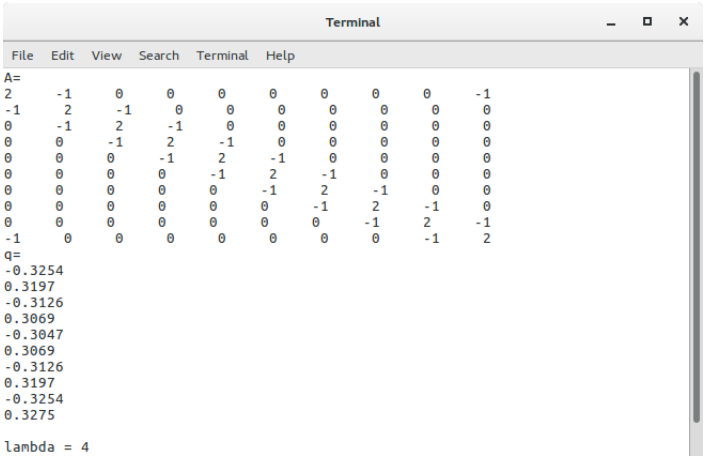

Figure 2: eigenvalue and it's eigenvector

t	0	1	2	3	4	5	6	7	8
$\overline{C}$	-15.4715	-1.98713	-0.392	-0.090	-0.0232	-0.0063	-0.0018	-0.00053	-0.00016
$\Delta\overline{C}/\overline{C}/\%$	0.111	0.167	0.398	0.587	0.733	0.842	0.927	0.995	1.050
t	9	10	11	12	13	14	15	16	17
$\overline{C}$	-4.983e-05	-1.533e-05	-4.733e-06	-1.468e-06	-4.58e-07	-1.431e-07	-4.478e-08	-1.405e-08	-4.409e-09
$\Delta\overline{C}/\overline{C}/\%$	1.075	1.088	1.132	1.171	1.213	1.258	1.293	1.335	1.374
t	18	19	20	21	22	23	24	25	26
$\overline{C}$	-1.385e-09	-4.344e-10	-1.363e-10	-4.282e-11	-1.345e-11	-4.232e-12	-1.332e-12	-4.188e-13	-1.316e-13
$\Delta\overline{C}/\overline{C}/\%$	1.405	1.421	1.435	1.454	1.488	1.524	1.562	1.585	1.603
t	27	28	29	30	31	32			
$\overline{C}$	-4.137e-14	-1.301e-14	-4.096e-15	-1.301e-15	-4.458e-16	-2.556e-16			
$\Delta\overline{C}/\overline{C}/\%$	1.619	1.638	1.660	1.676	1.709	1.731			

Table 3: Relative error of $\overline{C}(t)$

t	0	1	2	3	4	5	6	7	8
m	2.052	1.622	1.463	1.365	1.296	1.249	1.217	1.198	1.184
δm	0.0017	0.0027	0.0026	0.0024	0.0021	0.0022	0.0021	0.0018	0.0016
t	9	10	11	12	13	14	15	16	17
m	1.179	1.175	1.171	1.165	1.163	1.162	1.16	1.159	1.158
δm	0.0016	0.0017	0.0016	0.0015	0.0015	0.0014	0.0014	0.0014	0.0013
t	18	19	20	21	22	23	24	25	26
m	1.159	1.159	1.158	1.158	1.157	1.156	1.157	1.157	1.157
δm	0.0013	0.0012	0.0012	0.0012	0.0012	0.0012	0.0011	0.001	0.0011
t	27	28	29	30	31				
m	1.157	1.156	1.147	1.071	0.5561				
δm	0.0011	0.0011	0.0011	0.0012	0				

Table 4: using Jackknife to calculate m and its error

3.2 粲偶素的质量m及其误差

首先利用题目给出的公式计算出m(t)的值，之后，对于有200个块的数据进行jackknife操作， 图像见下:

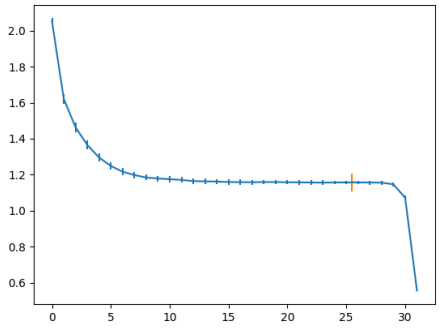

Figure 3: m and its errors

3.3 χ^2拟合

鉴于χ^2的表达式对于变量m来讲是一个二次多项式，我拿出第二次作业(interpolation)时写好的poly(多项式)类。先累加计算出χ^2的形式，再对m求导，得到极小值点。对于所有的长度大于 4 的区间进行扫描，找到使得单位自由度上的χ^2最小的位置。结果如输出： 其中，m的拟合值的误差按照$\Delta m = \sqrt{\frac{1}{\sum_{i=begin}^{end} \frac{1}{\Delta m_i}}}$计算，其值为:0.00053。因此最后的结果为$m = 1.5712 \pm 0.0005$ 直接用数值积分软件积分，得出：$p = 0.0459388$

```
fit result m= 1.1571227
the minimum chi-squre value = 0.110394 And the coresponded interval is [24, 27]
delta_m=0.000531472
Press <RETURN> to close this window...
```

Figure 4: best interval

3.4 新的ratio操作

只需利用新的函数形式重新操作，结果如输出： 其中，m的拟合值的误差按照$\Delta m = \sqrt{\frac{1}{\sum_{i=begin}^{end} \frac{1}{\Delta m_i}}}$计

```
fit result m= 1.1570929
the minimum chi-squre value = 7.67786e-06 And the coresponded interval is [25, 29]
Press <RETURN> to close this window...
```

Figure 5: best interval

算，其值为:0.00053。因此最后的结果为$m = 1.1571 \pm 0.0005$ 可以看到m的拟合值有明显的下降，然而其误差没有变化。直接用数值积分软件积分，得出：$p = 0.0143187$

3.5 相关系数矩阵

我写了一个函数:rho(mytype *in,int a,int b)。传入200*32的数据，指明需要计算的下表值a,b。函数中先产生1000个0到200的随机数，作为bootstrap抽样的代号，函数返回计算出的$\rho_{a,b}$。结果如下： 误差

```
rho(3,4)= 0.960691
rho(3,5)= 0.90806
Press <RETURN> to close this window...
```

Figure 6: best interval

可以使用公式法解出:$\delta\rho = \sqrt{\frac{1}{N_B - 1}\sum_{i=1}^{N_B}(\rho_i^* - \rho_0)^2}$ 最后的结果为：

$$\rho_{3,4} = 0.961 \pm 0.005$$

$$\rho_{3,5} = 0.908 \pm 0.015$$

从结果可以看出:3,4时间片之间的关联强于3,5之间的。