数值分析第一次大作业

院（系）名称：计算机学院
学 生 学 号：ZY2106339
学 生 姓 名：陈 铭 煊
指 导 老 师：谢 家 新

2021 年 11 月

1. 设计方案

题目描述

本次作业出自《数值分析（第四版）》（颜庆津 编著）计算实习说明书第一题：

设有501×501的实对称矩阵A，

$$A = \begin{bmatrix} a_1 & b & c & & \\ b & \ddots & \ddots & \ddots & \\ c & \ddots & \ddots & \ddots & c \\ & \ddots & \ddots & \ddots & b \\ & & c & b & a_{501} \end{bmatrix}$$

其中，$a_i = (1.64 - 0.024i)\sin(0.2i) - 0.64e^{\frac{0.1}{i}} (i = 1, 2, \ldots, 501)$，$b = 0.16$，$c = -0.064$。矩阵$A$的特征值为$\lambda_i (i = 1, 2, \ldots, 501)$，并且有

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{501}, |\lambda_s| = \min_{1 \leq i \leq 501} |\lambda_i|$$

1. 求λ_1, λ_{501}和λ_s的值。
2. 求A的与数$\mu_k = \lambda_1 + k\frac{\lambda_{501} - \lambda_1}{40}$最接近的特征值$\lambda_{i_k}$ $(k = 1, 2, \ldots, 39)$。
3. 求A的（谱范数）条件数$\mathrm{cond}(A)_2$和行列式$\det A$

说明：

1. 在所用算法中，凡是要给出精度水平ε的，都取$\varepsilon = 10^{-12}$
2. 选择算法时，应让矩阵A的所有零元素都不存储
3. 打印一下内容
    1. 全部源程序；
    2. 特征值$\lambda_1, \lambda_{501}, \lambda_\mathrm{s}, \lambda_{i_k}$ $(k = 1, 2, \ldots, 39)$ 以及 $\mathrm{cond(A)}_2, \det \mathrm{A}$的值
4. 采用e型输出实型数，并且至少显示12位有效数字

分析与总架构

实对称矩阵的所有特征值都是实数，因此所有的计算都可以在实数域下进行。

首先通过幂法，我们能求出绝对值最大的特征值λ，即λ_1或者λ_{501}。显然，如果$\lambda \leq 0$，那么求出的是λ_1，否则是λ_{501}。无论是哪一种情况，再次用幂法，求带原点平移过后的矩阵$A - \lambda I$，得到的特征值λ'，即得到A的另一个端点特征值为$\lambda' + \lambda$。

λ_s作为按模最小的特征值，可以用反幂法求出。

对于第二问，通过构造带原点平移的矩阵：

$$A_k = A - \mu_k I \; (k = 1, 2, \ldots, 39)$$

可以用反幂法求出按模最小特征值λ_k'，于是$\lambda_k = \lambda_k' + \mu_k$，即为最接近$\mu_k$的特征值。即需要40次反幂法。

由书本P30页：

$$\mathrm{cond}(A)_2 = \left| \frac{\lambda_{\max}}{\lambda_{\min}} \right|$$

我们用第一问的答案代入即可；而对于$\det A$，我们可以用LU分解（Doolittle分解）得到：

$$\det A = \det L \cdot \det U = \det U = \prod_{i=1}^{n} u_{ii}$$

此外，LU分解也会作为反幂法中求解线性方程组的子算法出现。

至此解决问题的基本框架已经确定。

矩阵与向量

对于一个带状矩阵A，我们可以采用P26页的方法来进行压缩存储：

$$C = \begin{bmatrix} 0 & 0 & a_{13} & a_{24} & a_{35} & a_{46} \\ 0 & a_{12} & a_{23} & a_{34} & a_{45} & a_{56} \\ a_{11} & a_{22} & a_{33} & a_{44} & a_{55} & a_{66} \\ a_{21} & a_{32} & a_{43} & a_{54} & a_{65} & 0 \end{bmatrix}$$

其中映射方式为$a_{ij} = c_{i-j+s+1,j}$。

而对于实对称带状矩阵，我们可以只存储其上三角部分。对于$i > j$的下三角位置用其对称位置$a_{ji} = c_{j-i+s+1,i}$代替，进一步节省了空间，也防止了不安全的非对称修改。

对于实对称带状矩阵的实现被抽象为C++类`SymBandMatrix`，使用`std::vector`容器存储元素，为了提升效率，节省空间，将二维结构压缩为一维结构，并且实际存储下标从0开始计数。此外，实现中包含了一些重要方法，例如和向量的乘法等。

向量的实现被抽象为类`Vector`，同样使用`std::vector`存储元素。此外由于幂法中需要随机初始化向量，这里给出了随机初始化的方式：每个元素在$\mu = 0, \sigma = 3.2$的高斯分布下进行独立采样。

幂法与反幂法

对于幂法和反幂法，本文实现了基于2-范数和无穷范数两种迭代方式。这里仅给出采用无穷范数的反幂法的迭代格式：

$$\begin{cases} \text{任取非零向量 } u_0 = (h_1^{(0)}, h_2^{(0)}, \ldots, h_n^{(0)}) \in \mathrm{R^n} \\ |h_r^{(k-1)}| = \max_{1 \le j \le n} |h_j^{(k-1)}| \\ y_{k-1} = u_{k-1} / |h_r^{(k-1)}| \\ A u_k = y_{k-1} \\ \beta_k = \mathrm{sgn}(h_r^{(k-1)} h_r^{(k)}) \\ \quad (k = 1, 2, \ldots) \end{cases}$$

在$|\beta_k - \beta_{k-1}| / |\beta_k| \le \varepsilon$时，终止迭代，认为精度满足要求，将$\frac{1}{\beta_k}$作为特征值，将$y_{k-1}$作为对应的特征向量。

其余迭代格式参考书本。

对于幂法和反幂法的实现，包含在类`EigenValueUtils`的四个静态函数中。

LU分解与线性方程组求解

在求行列式和反幂法中都会用到LU分解。

在反幂法中，我们经常需要求解$Ax = b$的解，其中A矩阵固定，而右侧的b不断改变，因此我们可以先求出矩阵A的LU分解，在每次求线性方程组时重复利用，节省时间。

这一部分的实现采用了Doolittle法，详见P21页。实现包含在了类`LinearEqLuSolver`中，其构造函数负责做LU分解，建立LU矩阵，调用`solveEq`函数可以求解一次线性方程组。此处LU矩阵的存储复用了之前的`SymBandMatrix`类。

2. 源程序代码

本程序采用C++17标准编写，模块清晰，通用性强，效率优秀，使用CMake（version ≥ 3.16）组织代码，使用MSVC 8.1(amd64)或GCC 9.3.0均可编译通过。文件结构如下：

```
.
├── CMakeLists.txt
├── EigenValueUtils.cpp
├── EigenValueUtils.h
├── LinearEqLuSolver.cpp
├── LinearEqLuSolver.h
├── SymBandMatrix.cpp
├── SymBandMatrix.h
├── TimerUtil.cpp
├── TimerUtil.h
├── Vector.cpp
├── Vector.h
└── main.cpp
```

代码内容见下页。

3. 上机计算结果

以某次运行为例：得到结果

第一问

```
lambda1=-0.107001136150E+02
lambda501=0.972463409878E+01
lambdaS=-0.555791079423E-02
```

第二问

```
-0.101829340331E+02
-0.958570742507E+01
-0.917267242393E+01
-0.865228400790E+01
-0.809348380868E+01
-0.765940540769E+01
-0.711968464869E+01
-0.661176433940E+01
-0.606610322660E+01
-0.558510105263E+01
-0.511408352981E+01
-0.457887217687E+01
-0.409647092626E+01
-0.355421121575E+01
-0.304109001813E+01
-0.253397031113E+01
-0.200323076956E+01
-0.150355761123E+01
-0.993558606008E+00
-0.487042673885E+00
0.223173624957E-01
0.532417474207E+00
0.105289896269E+01
0.158944588188E+01
0.206033046027E+01
0.255807559707E+01
0.308024050931E+01
0.361362086769E+01
0.409137851045E+01
0.460303537828E+01
0.513292428390E+01
0.559490634808E+01
0.608093385703E+01
0.668035409211E+01
0.729387744813E+01
0.771711171424E+01
0.822522001405E+01
0.864866606519E+01
0.925420034457E+01
```

第三问

```
cond2=0.192520427390E+04
det=0.277278614175E+119
```

根据书本提供的部分答案，可知结果正确，精度符合要求。

4. 讨论分析

重要的一点是迭代次数和向量初始值很相关，如果设置一些特殊的简单向量，迭代次数往往会增加。因此程序中采用方差为3.2的正太分布来采样，产生随机向量。

测试中发现，2-范数方案的程序运行时间在0.025~0.033s之间。无穷范数方案的程序运行时间在0.012~0.015之间。两者相差一倍。通过对2-范数和无穷范数两种方案的迭代次数，可以发现主要原因是无穷范数的方案收敛速度更快，迭代次数更少。以某次运行的采样为例：

迭代次数\方法	2-范数	无穷范数
幂法求λ_1	280	558
幂法求λ_{501}	1161	1777
反幂法求λ_s	86	14
反幂法求λ_k（总计）	3236	818

发现对于幂法而言，采用无穷范数并不会比采用2-范数优；而在反幂法中，2-范数法会在某些时候收敛过慢，迭代次数过多。