Numerical Analysis

Homework 1. Gram-Schmidt Process

Due: Mar. 14, 2020

Given an $n \times n$ nonsingular matrix $\mathbf{A}$, an orthogonal matrix $\mathbf{G}$ can be constructed using Gram-Schmidt process as following. Let

$$\mathbf{A} = [\mathbf{A}_1 \quad \mathbf{A}_2 \quad \cdots \quad \mathbf{A}_n] \tag{1.1}$$

and

$$\mathbf{G} = [\mathbf{G}_1 \quad \mathbf{G}_2 \quad \cdots \quad \mathbf{G}_n] \tag{1.2}$$

where $\mathbf{A}_i$ denotes the i-th column of matrix $\mathbf{A}$. Similarly, $\mathbf{G}_i$ denotes the i-th column of matrix $\mathbf{G}$. Then,

$$\begin{aligned} \mathbf{G}_1 &= \mathbf{A}_1 \\ \mathbf{G}_k &= \mathbf{A}_k - \sum_{i=1}^{k-1} \frac{(\mathbf{A}_k^T \mathbf{G}_i)\mathbf{G}_i}{\mathbf{G}_i^T \mathbf{G}_i}, \qquad k = 2, \cdots, n. \end{aligned} \tag{1.3}$$

Since, $\mathbf{G}$ is an orthogonal matrix $\mathbf{G}^T\mathbf{G} = \mathbf{D}$, where $\mathbf{D}$ is a diagonal matrix.

The mathematical description of the Gram-Schmidt process has been shown in Eq. (1.3). However, there are still more than one way to implement this process. Four are shown below.

Algorithm 1. Gram-Schmidt Process

```
$\mathbf{G}_1 = \mathbf{A}_1$;
for ($k = 2, \cdots, n$) {
    $\mathbf{G}_k = \mathbf{0}$;
    for ($i = 1, \cdots, k-1$) {
        $\mathbf{G}_k = \mathbf{G}_k + ((\mathbf{A}_k^T \mathbf{G}_i)\mathbf{G}_i)/(\mathbf{G}_i^T \mathbf{G}_i)$;
    }
    $\mathbf{G}_k = \mathbf{A}_k - \mathbf{G}_k$;
}
```

Algorithm 2. Modified Gram-Schmidt Process, 1

```
$\mathbf{G}_1 = \mathbf{A}_1$;
for ($k = 2, \cdots, n$) {
    $\mathbf{G}_k = \mathbf{A}_k$;
    for ($i = 1, \cdots, k-1$) {
        $\mathbf{G}_k = \mathbf{G}_k - ((\mathbf{G}_k^T \mathbf{G}_i)\mathbf{G}_i)/(\mathbf{G}_i^T \mathbf{G}_i)$;
    }
}
```

Algorithm 3. Modified Gram-Schmidt Process, 2

```
$\mathbf{G}_1 = \mathbf{A}_1$;
for ($k = 2, \cdots, n$) {
    $\mathbf{G}_k = \mathbf{A}_k$;
    for ($i = 1, \cdots, k-1$) {
        $\mathbf{G}_k = \mathbf{G}_k - (\mathbf{G}_k^T \mathbf{G}_i)/(\mathbf{G}_i^T \mathbf{G}_i)\mathbf{G}_i$;
    }
}
```

Algorithm 4. Modified Gram-Schmidt Process, 3

```
for ($i = 1, \cdots, n$) $\mathbf{G}_i = \mathbf{A}_i$
for ($j = 1, \cdots, n-1$) {
    $\alpha = \mathbf{G}_j^T \mathbf{G}_j$
    for ($k = j+1, \cdots, n$) {
        $\mathbf{G}_k = \mathbf{G}_k - (\mathbf{G}_k^T \mathbf{G}_j)/\alpha \cdot \mathbf{G}_j$;
    }
}
```

For this homework, please complete the C++ classes `VEC` and `MAT` definitions, and then use these two classes to write a C++ program that

1. Reads matrix $\mathbf{A}$ from a file,
2. Performs Gram-Schmidt process to find matrix $\mathbf{G}$ using one of the algorithms above,
3. Verifies $\mathbf{G}$ is an orthogonal matrix by performing matrix-matrix multiplication to get

$$\mathbf{D} = \mathbf{G}^T \mathbf{G} \tag{1.4}$$

   And then find the value of the scalar σ by

$$\sigma = \sqrt{\sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} d_{i,j}^2}. \tag{1.5}$$

The scalar σ is the square root of the sum of squares of all the off-diagonal elements of matrix $\mathbf{D}$. If the matrix $\mathbf{G}$ is orthogonal, the this number should be 0.

For your testing, seven matrices have been prepared. They are `m3.dat`, `m4.dat`, `m5.dat`, `m6.dat`, `m7.dat`, `m8.dat`, and `m9.dat` files. The first line of each file is the linear dimension of the matrix, n, followed by n lines for the n rows. The linear dimensions of these seven matrices are 3, 10, 100, 200, 400, 800, and 1600. Please use the unix **time** command to record the CPU time for solving each matrix using various algorithms. Plot out the CPU time vs. matrix dimension and state your observations.

Example of program execution is as following.

```
$ time ./a.out < m3.dat
Algorithm 1. Gram-Schmidt Process
sigma = 1.33227e-15
0.001u 0.000s 0:00.00 0.0% 0+0k 0+0io 0pf+0w
```

The last line is the result of the `time` command, and the first number of the line is the CPU time, in seconds, which is of interest to us. The postfix ‘u’ stands for ‘user’ not micro-seconds.

Notes.

1. For this homework you need to turn in a C++ program that performs the Gram-Schmidt process. All four algorithms should be available in your program but only one is executed each time in order to compare the performance of each algorithm. Please name this program `hw01.cpp`. The header and definition files for `VEC` and `MAT` classes should also be turned in.
2. A `pdf` file is also needed. This file should include a plot that shows the CPU times as a function of the linear dimensions of the matrices. Also, state your observations in this report. Please name this report `hw01a.pdf`.

3. Submit your `hw01.cpp`, `VEC.h`, `VEC.cpp`, `MAT.h`, `MAT.cpp` and `hw01a.pdf` on EE workstations. Please use the following command to submit your homework 1.

```
$  ~ee4070/bin/submit hw01 hw01.cpp VEC.h VEC.cpp MAT.h MAT.cpp hw01a.pdf
```

   where `hw01` indicates homework 1.

4. Your report should be clearly written such that I can understand it. The writing, including English grammar, is part of the grading criteria.

5. So far, we have not defined any operator or function to extract a column from any matrix, but we do have the indexing operator to extract row m from a matrix. To take advantage of this operator, you can perform Gram-Schmidt process on a transposed matrix using row operations. After the process is completed, matrix $\mathbf{G}$ can be obtained by transposing the resultant matrix.