Lab Session 5

MA-423 : Matrix Computations Lab 2018 R. Alam

---

1. The purpose of this example is to illustrate that Householder QR factorization (or for that matter QR factorization obtained by using rotations) is backward stable, that is, computed QR factors are the exact QR factors of a slightly perturbed matrix.

```
>>R = triu(randn(50)); % compute a 50-by-50 random upper triangular matrix.
>>[Q, X] = qr(randn(50)); % compute a 50-by-50 random unitary matrix.
>>A = Q*R; % A is a matrix with known QR factors.
>>[S, T] = qr(A); % Compute Householder QR factorization of A.
```

   How accurate are `S` and `T`? Show that `S` and `T` are very far from the known QR factors Q and R of A. This illustrates that the Householder QR factrization algorithm is not what is called **forward stable**, that is, the computed factors are not close to the exact factors. Compute the errors

```
>>[norm( Q-S ), norm( R-T )] % what do you observe?
```

   Now test the backward stability of the Householder QR factorization. Setting `E = S*T - A`, we have `A+E = S*T`. Hence if $\|E\|_2/\|A\|_2 = \mathcal{O}(\mathbf{u})$ then clearly the algorithm `qr` is backward stable. Compute

```
>>norm(A-S*T) % what is your conclusion?
```

   Comment: The numerical rank of $A \in \mathbb{R}^{n\times m}$ is obtained in MATLAB by typing `rank(A)`. MATLAB uses the `SVD` of A to obtain this value. More specifically, it is obtained by calculating a tolerance level $tol = \epsilon \max\{n, m\}\|A\|_2$ where ϵ stands for machine epsilon and then setting `rank(A)` to be the number of singular values of A which are greater than this value of tol. It is possible for the user to change this tol value to something else. Type `help rank` for details. The purpose of the next exercise is to illustrate that the above method computes the rank of a matrix quite efficiently in the presence of rounding.

2. Type `A = randn(7,4)` and examine its rank by typing `rank(A)`. Since random matrices are usually full rank, its rank will be 4 in all probability. Add two more columns to A by typing `A(:,5,6) = A(: 2,3) + A(:,3,4)`; This will create a fifth column in A which is the sum of the 2nd and 3rd columns and a sixth column which is the sum of the 3rd and 4th columns. Theoretically, the rank of A should remain unchanged. What happens to the numerical rank? Type `rank(A)` to find out.

   Comment: The purpose of the next exercise is to illustrate that in the presence of rounding, the `SVD` is generally more efficient in determining the rank of a matrix than the rank revealing `QR` factorization.

3. The *Kahan matrix* $R_n(\theta)$ is an $n \times n$ upper triangular matrix depending on a parameter

θ. Let $c = \cos(\theta)$ and $s = \sin(\theta)$. Then

$$R_n(\theta) := \begin{pmatrix} 1 & & & & \\ & s & & & \\ & & s^2 & & \\ & & & \ddots & \\ & & & & s^{n-1} \end{pmatrix} \begin{pmatrix} 1 & -c & -c & \dots & -c \\ & 1 & -c & \dots & -c \\ & & 1 & & -c \\ & & & \ddots & \vdots \\ & & & & 1 \end{pmatrix}.$$

If θ and n are chosen so that s is close to 1 and n is modestly large, then none of the main diagonal entries are extremely small. It appears that the matrix is far from rank deficient, which is actually not the case. Consider $R_n(\theta)$ when $n = 90$ and $\theta = 1.2$ radians. Verify for yourself that the largest main diagonal entry of $R_n(\theta)$ is 1 and the smallest is .001.

(a) To generate $R_n(\theta)$ in MATLAB and find its singular values type

$$\texttt{A} = \texttt{gallery}(\texttt{'kahan'}, 90, 1.2, 0);$$
$$\texttt{sig} = \texttt{svd}(\texttt{A})$$

Type `format short e` and examine σ_1, σ_{89} and σ_{90}. Type `rank(A)` to get MATLAB's opinion of the numerical rank of A.

(b) Type `A = gallery('kahan',90,1.2,25)` to get a slightly perturbed version of the Kahan matrix. (This produces the Kahan matrix with very small perturbations to the diagonal entries. Type `help private/kahan` for more details.) Repeat part (a) for the perturbed matrix. Perform a QR decomposition by column pivoting on A by typing `[Q,R,E] = qr(A)`. Verify that no pivoting was done in this case by examining the value of `dif = norm(eye(90) - E)`. Examine $R(90, 90)$ and infer that the rank revealing QR decomposition failed to detect the numerical rank deficiency of A.

4. The purpose of this exercise is to test the text mining algorithms. Consider the documents

Doc. 1: The **Google matrix** G is a model of the **Internet**.
Doc. 2: G_{ij} is nonzero if there is a **link** from **web page** j to i.
Doc. 3: The **Google matrix** G is used to **rank** all **web pages**.
Doc. 4: The **ranking** is done by solving a **matrix eigenvalue** problem.
Doc. 5: **England** dropped out of the top 10 in the **FIFA ranking.**

Next, consider the dictionary { **eigenvalue, England, FIFA, Google, Internet, link, matrix, page, rank, web** } and determine the term-document matrix A. The list of **terms** created in alphabetic order is called **index.** Index is created after eliminating (a) all **stop words** - common words whose occurrence in a document does distinguish it from other documents - and (b) performing **stemming**. Stemming is the process of reducing each word that is conjugated or has a suffix to its stem. For example, for information retrieval, no information is lost in the following reduction:

$$\left.\begin{array}{l} \textbf{comput}\textit{able} \\ \textbf{comput}\textit{ation} \\ \textbf{comput}\textit{ing} \\ \textbf{comput}\textit{ed} \\ \textbf{comput}\textit{ational} \end{array}\right\} \longrightarrow \textbf{comput}$$

Public domain stemming algorithms are available on the Internet.

Suppose that we want to find all documents that are relevant to the query **ranking** of **web pages**. Let **v** be the query vector (the term-document matrix of the query). Then the information retrieval task can now be formulated as a mathematical problem.

Problem: Find the columns of A that are close to the query vector **v**.

Use cosine distance measure (with a tolerance) to return the relevant document. Your task is to determine the relevant documents (a) by comparing cosine distances from **v** to the columns of A and (b) by latent semantic indexing (LSI) method.

*** End ***