Lab Session 11

MA-581 : Numerical Computations Lab R. Alam November 04, 2035

1. This problem illustrates SVD based image compression algorithm. The MATLAB commdand `A = im2double(imread('photo.jpg'))` reads an image (color as well as black and white) and converts it into a matrix `A`. The command `AG = rgb2gray(A)` converts the image to a grayscale image `AG`. The command `image(X)` displays the image (color or gray) stored in a matrix `X`.

   A compressed image of the grayscale image `AG` is computed as follows. Compute the SVD $AG = U\Sigma V^\top$ and the best k rank approximation $A_k := U \begin{bmatrix} \mathrm{diag}(\sigma_1, \ldots, \sigma_k) & 0 \\ 0 & 0 \end{bmatrix} V^T$ for a chosen value of $k < \mathrm{rank}(A)$. Then A_k represents a compressed image. The compressed image can be displayed by the command `image(`A_k`)`. Use the following commands:

```
[U, S, V] = svd(AG); ; Ak = U(:, 1:k)*S(1:k, 1:k)*V(:,1:k)'; image(Ak)
```

   The storage required for A_k is $k(m+n)$ whereas the storage required for the full image is mn. Therefore, $\frac{(m+n)k}{mn}$ gives the compression ratio for the compressed image. Also the error in the representation is $\frac{\sigma_{k+1}}{\sigma_1}$.

   Choose a photo (jpg, png or any format) and run the above commands for various choices of k and make a table that records the relative errors and compression ratios for each choice.

   Alternatively, use the following commands which first loads a built-in 320×200 matrix X that represents the pixel image of a clown

```
load clown.mat; [U, S, V] = svd(X); colormap('gray');
image(U(:, 1:k)*S(1:k, 1:k)*V(:,1:k)')
```

   Run the above commands for various choices of k and make a table that records the relative errors and compression ratios for each choice.

   How does the choice of approximating rank k affect the visual qualities of the images? There are no precise answers here. Your results will depend upon the images you choose and the judgments you make.

2. **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 $\texttt{tol} = \texttt{eps}\max\{n, m\}\|A\|_2$ where `eps` is the 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 this example 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.

   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

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

   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

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

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, P] = qr(A)`. Verify that no pivoting was done in this case by examining the value of `dif = norm(eye(90) − P)`. Examine $R(90, 90)$ and infer that the rank revealing QR decomposition failed to detect the numerical rank deficiency of A.

3. Determine the polynomial of degree 19 that best fits the function $f(t) = \sin(\frac{\pi}{5}t) + \frac{t}{5}$ for $t_1 = -5, t_2 = -4.5, \ldots, t_{23} = 6$. Setup the LSP $Ax = b$ and determine the polynomial p in three different ways:

   (a) By using the matlab command
   `>> A \ b`
   This uses QR factorization to solve the LSP $Ax = b$. Call this polynomial p_1.

   (b) By solving the normal equation $A^*Ax = A^*b$. Use `x = (A'*A)\(A'*b)`. Call this polynomial p_2.

   (c) By solving the system $\begin{bmatrix} I_m & A \\ A^* & 0 \end{bmatrix} \begin{bmatrix} -r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$. Call this polynomial p_3.

   Compute the condition number (use the matlab command `cond(A)`) of the coefficient matrix associated with each of the systems that you are solving. If $\text{cond}(A) = 10^t$ then we can expect the solution to be have $16 - t$ correct digits and hence lose t digits of accuracy.

   Print the result to 16 digits (use `format long e`). Which one is the most ill conditioned?

   The norm of the residual $\|r\|_2 = \|Ax - b\|_2$ gives an idea of the goodness of the fit. Compute residual for each method.

   Finally, plot the polynomials p_1, p_2, p_3 and the function f on $[-5, 6]$. Use different colors to distinguish these plots. Do you observe any difference? If yes, which polynomial is a better approximation of f?

4. The Least Squares Problem (LSP) $Ax = b$ has a solution where the fit is good if b is nearly in the range $R(A)$ of A or in other words the angle θ between b and Ax is very small. The purpose of the following exercise is to show that in such cases, the QR method of solving the LSP $Ax = b$ is better than Normal Equations method.

   Use the linspace command to generate a *column vector* X consisting of 50 equally spaced points between 0 and 1. Generate the Vandermonde matrix which has columns $X.^{i-1}$ for $i = 1 : 7$. Choose `w = randn(7, 1)` and `b = A * w`. Then $b \approx p(X)$ where p is the polynomial $p(t) = w(1) + w(2)t + \cdots + w(7)t^6$. This ensures that $\theta \approx 0$ for the LSP $Ax = b$. Solve this problem via Normal Equations method and QR method via reflectors (this is the default procedure so that you just have to type `A\b` for this!) and denote the solutions as `xhat` and `xtilde`, respectively. Examine the relative errors $\frac{\|\texttt{xhat} - \texttt{w}\|_2}{\|\texttt{w}\|_2}$, and $\frac{\|\texttt{xtilde} - \texttt{w}\|_2}{\|\texttt{w}\|_2}$ in the solutions as well as those in the fits $\frac{\|\texttt{rhat}\|_2}{\|\texttt{b}\|_2}$ and $\frac{\|\texttt{rtilde}\|_2}{\|\texttt{b}\|_2}$ where `rhat := b − A * xhat` and `rtilde := b − A * xtilde`. Which method fares better? Also find the condition number of A.

*** End ***