Mat1120 oblig 2

Soran Hussein Mohmmed, brukernavn: soranhm

October 2016

1 Oppgave

velger tilfeldige $\mathbf{u} \in \mathbf{R}^4$ og $\mathbf{v} \in \mathbf{R}^5$, m = 4 og n = 5, som ikke er null. For å begrunne at rank $\mathbf{uv}^T$ =1. Bruker matlab med matrisene:

$$\mathbf{u} = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 2 \end{bmatrix} \text{ og } \mathbf{v}^T = \begin{bmatrix} 1 & 2 & 3 & 2 & 3 \end{bmatrix}, \mathbf{uv}^T = \begin{bmatrix} 2 & 4 & 6 & 4 & 6 \\ 3 & 6 & 9 & 6 & 9 \\ 4 & 8 & 12 & 8 & 12 \\ 2 & 4 & 6 & 4 & 6 \end{bmatrix},$$

$$rref(\mathbf{uv}^T) = \begin{bmatrix} 1 & 2 & 3 & 2 & 3 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

og ut fra dette kan jeg se at rank $\mathbf{uv}^T = 1$ pga den består av kunn et pivotelement.Annet grunn til dette er at både U og V er på formen m x 1,altså n = 1, og ranken kan ikke være større enn den minste n eller m verdi. Har også brukt rang kommandoen `rank(`$\mathbf{uv}^T$`)` til å finne ut rangen. Koden:

```
u = [2;3;4;2];        % tilfeldig u matrise
vt = [1 2 3 2 3];     % tilfeldig v^T matrise
uv = u*vt;            % regner ut uv^T
rref(uv);             % f?r den p? redusert trappeform
rank(uv);             % bruker rank funckjon til ? finne rang
```

2 Oppgave

B er en m x n matrise og C er en n x p matrise, så kan matriseproduktet BC skrives som:

$$BC = \sum_{j=1}^{n} kol_j(B) rad_j(C) \tag{1}$$

Skal nå bruke likningen (1) med B = UΣ og C = V til å begrunne:

$$A = \sum_{j=1}^{r} \sigma_j \mathbf{u}_j \mathbf{v}_j^T \tag{2}$$

A = UΣV, Grunnen til at det går fra n til r over summ tegnet er pga at restrenede etter r vil være nuller som vi ikke trenger å ta med.
Siden B = UΣ så vil $kol_j(B) \Rightarrow kol_j(U\Sigma)$. UΣ: U er en kolonne matrise og Σ er en diagonal matrise, dette vil da gi $kol_j(U\Sigma) \Rightarrow \sigma kol_j(U) \Rightarrow \sigma_j \mathbf{u}_j$.

$$\begin{bmatrix} \mathbf{u_1} & \mathbf{u_2} & ... & \mathbf{u_m} \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & ... & 0 \\ 0 & \sigma_2 & ... & ... \\ ... & ... & ... & 0 \\ 0 & 0 & ... & \sigma_r \end{bmatrix} = \begin{bmatrix} \sigma_1\mathbf{u_1} & \sigma_2\mathbf{u_2} & ... & \sigma_r\mathbf{u_r} \end{bmatrix}$$

Når det gjelder $rad_j(C) : rad_j(C) \Rightarrow rad_j(V^T) \Rightarrow kol_j(V)^T \Rightarrow \mathbf{v}_j^T$
Dette gir da:

$$A = U\Sigma V = BC = \sum_{j=1}^{n} kol_j(B)rad_j(C) = \sum_{j=1}^{r} kol_j(U\Sigma)rad_j(V) = \sum_{j=1}^{r} \sigma_j kol_j(U)kol_j(V^T) = \sum_{j=1}^{r} \sigma_j \mathbf{u}_j \mathbf{v}_j^T \tag{3}$$

3 Oppgave

3.1 a)

Brukt formel (3) fra oppgave 2 og ved hjelp av kommandoene har jeg skrevet en funksjon '`functionAk = svdApprox(A,k)`':

```
function functionAk = svdApprox(A,k)
    A = double(A);                  % konvertert til datatype
    r = rank(A);                    % velger r
    [U,S,V] = svd(A);               % returnerer SVD
    functionAk = zeros(size(A));    % lager tom matrise
    for i = 1:k
        functionAk = functionAk + S(i,i)*U(:,i)*V(:,i)';% likning (2)
        if k > r
            fprintf(' for stor k \n'); % feilmelding for stor k verdi
            break
        end
    end
end
```

3.2 b)

Bruker nå filen 'mm.gif' , leser ut matrisen og tester flere ting på den :

```
A = imread('mm.gif','gif'); % leser in matrisen
A = double(A);              % konvertert til datatype
% sjekker om a er en 256x256 matrise
storresle = size(A)
% bestemmer rang til A
r = rank(A)
% rank(A,eps), eps = 0.001
```

```
eps = 0.001;
reps = rank(A,eps)
```

```
storresle =

   256   256


r =

   256


reps =

   256
```

Figure 1: Utskrift fra oppgave 3b)

3.3 c)

Bruker nå `svdApprox(A,k)` med matrisen A fra Marilyn Monroe med k1 = 8 og k2 = 32:

```
A = imread('mm.gif','gif');
k1 = 8; k2 = 32;
A1 = svdApprox(A,k1); % setter inn matrisen og k i funksjonen svdApprox
A2 = svdApprox(A,55); % tilh?rer oppg 5a)
A3 = svdApprox(A,k2);
imageview(A1)
%imageview(A3)
```

3.4 d)

Figur 1 fra Oblig arket har gjentatte mønster som vil si at den ikke trenger så stor rang til å vises. Ved å se på figuren med 5 forskjellige farger (som går fra lys til mørk), jeg vil si at rangen til denne er 5 pga. gjentatte mønster.

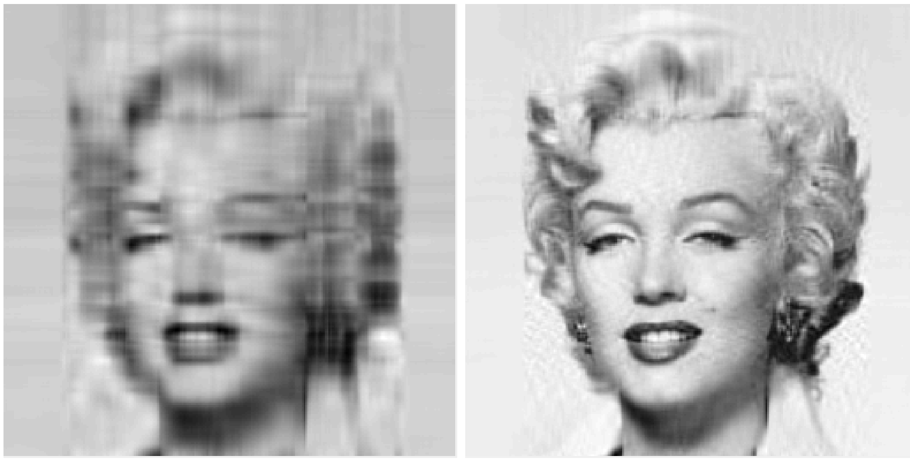

Figure 2: Utskrift fra oppgave 3c, med k = 8 og k = 32

4 Oppgave

4.1 a)

Bruker nå kunn singulærverdiene σ_j til å lage en funskjon av j og plotter dette:

```
A = imread('mm.gif','gif');
A = double(A);
[U,S,V] = svd(A);
x = 1:256;
y = diag(S); % siden sigma er diagonalmatrise
plot(x,y)   % plotter 1—256 mot sigmaene fra A
```

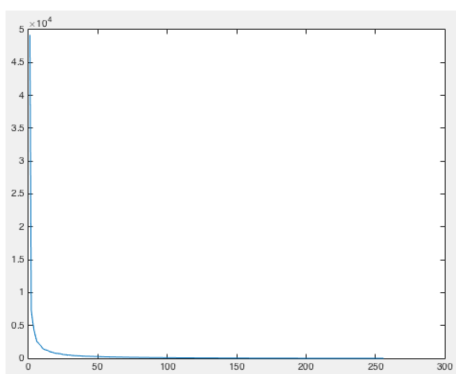

Figure 3: Utskrift fra oppgave 4a

4.2 b)

Bruker nå kommandoen 'B = `round`(255 ∗ `rand`(256, 256))' til å konstruere en ligenen de matrise som A men med helt tilfelldige tall, og det lager et bilde med piksler over alt. Når plotter σ fra A og B i samma plot ser jeg at grafen til A (blå) er mye mer nøyaktig enn grafen til B (rød), dette skyldes at B har σ som synker brått, mens i A har ganske ballensert synking av σ.

```
% tilfeldige tall mellom 1—256 i en 256x256 matrise
B = round(255 * rand(256, 256));
[U,S2,V] = svd(B);
y = diag(S2);
% gj?r det samma som i 3c) og bruker svdApprox
B1 = svdApprox(B,32);
imageview(B1)

A = imread('mm.gif','gif');
A = double(A);
[U,S,V] = svd(A);
x = 1:256;
y2 = diag(S); % siden sigma er diagonalmatrise
plot(x,y,'r',x,y2,'b')   % plotter sigma fra A og B sammen
```

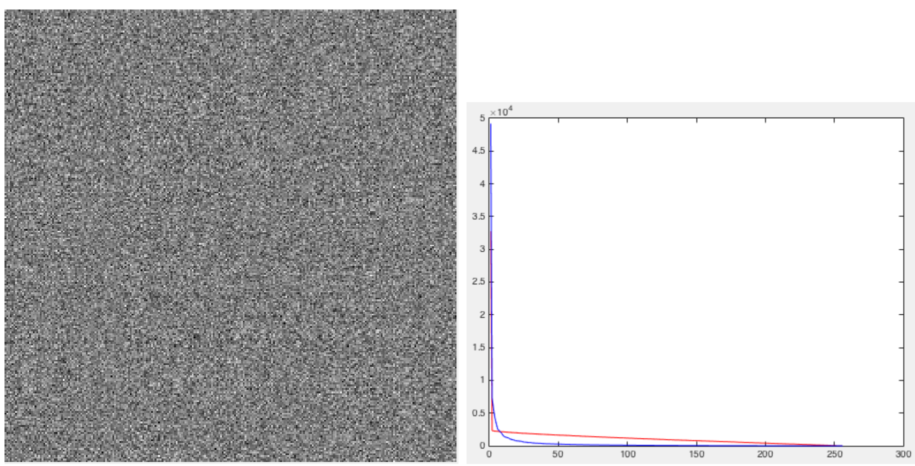

Figure 4: Utskrift fra oppgave 4b.σ A(rød) B(blå)

5 Oppgave

5.1 a)

For å kunne regne ut det kompinerte bilde av rang k til en m x n bildematrise A så trenger man 1 av hver $\sigma, \mathbf{v}$ og $\mathbf{u}$. For å få et visuelt bilde av Marilyn Monroe like godt som den opprinnlige må vi velge k = 55. Dette har jeg prøvd med matlab og zoomet inn på et sted i det opprinnnlige og tilnærmingen og fant den som hadde mest like piksler.

5.2 b)

Funkjsonen er laget i matlab, bruker nå funkjsonen svdApprox(A,55), med k = 55 som jeg fant var best tilnærming til den opprinnlige i a). Dette gir da en feil med 0.0045, desto nærmere 256 jeg velger k gir desto mindre feil.
Kode:

```
function functionerror = relError(A, Ak)
    A = double(A);  % konvertert til datatype
    functionerror = norm(A—Ak)/norm(A)  % relative feilen
end


A = imread('mm.gif','gif');
Ak = svdApprox(A,55);       % regner ut tiln?rmingen
% regner ut feil mellom tiln?rmingen og orginale
relError(A, Ak);
```

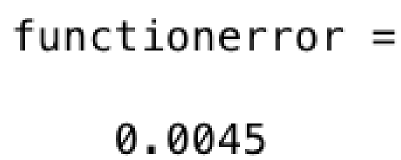

Figure 5: Utskrift fra oppgave 5b.Error