Oppgave 1

La $\mathbf{u} = [u_1 \quad \cdots \quad u_m]^T$ være en kolonnevektor i $\mathbb{R}^m$, og $\mathbf{v}^T = [v_1 \quad \cdots \quad v_n]$ være en rad vektor i $\mathbb{R}^n$, da har vi:

$$\mathbf{u}\mathbf{v}^T = \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix} [v_1 \quad \cdots \quad v_n] = \begin{bmatrix} v_1 u_1 & \cdots & v_n u_1 \\ \vdots & \ddots & \vdots \\ v_1 u_m & \cdots & v_n u_m \end{bmatrix} = \left[v_1 \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix} \quad \cdots \quad v_n \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix} \right] = [v_1 \mathbf{u} \quad \cdots \quad v_n \mathbf{u}]$$

La $c_1, \dots, c_n$ være vilkårlige tall i $\mathbb{R}$, da kan vi skrive kolonnerommet til $\mathbf{u}\mathbf{v}^T$ slik:

$$\Leftrightarrow \mathrm{Col}(\mathbf{u}\mathbf{v}^T) = c_1(v_1\mathbf{u}) + \cdots + c_n(v_n\mathbf{u}) = (c_1 v_1 + \cdots + c_n v_n)\mathbf{u} = c\mathbf{u}$$

Siden c_i er vilkårlige for alle $i = 1, \dots, n$, så er $c \in \mathbb{R}$. Dette betyr at $\{\mathbf{u}\}$ danner en basis for $\mathrm{Col}(\mathbf{u}\mathbf{v}^T)$, og da:

$$\mathrm{rank}(\mathbf{u}\mathbf{v}^T) = \dim(\mathrm{Col}(\mathbf{u}\mathbf{v}^T)) = 1$$

MATLAB utskrift:

```
>> u = floor(9*rand(4,1))
u =
     6
     7
     1
     4
>> v = floor(9*rand(5,1))
v =
     8
     3
     5
     2
     6
>> A = u*v'
A =
    48    18    30    12    36
    56    21    35    14    42
     8     3     5     2     6
    32    12    20     8    24
>> rank(A)
ans =
     1
>>
```

Oppgave 2

La $\Sigma = [\mathbf{z}_1 \quad \cdots \quad \mathbf{z}_r \quad \mathbf{z}_{r+1} \quad \cdots \quad \mathbf{z}_n]$ være den samme matrise som er vist på den tredje side av oppgaveteksten. Da kan vi beregne $U\Sigma$ slik:

$$\begin{aligned} U\Sigma = U[\mathbf{z}_1 \quad \cdots \quad \mathbf{z}_r \quad \mathbf{z}_{r+1} \quad \cdots \quad \mathbf{z}_n] &= [U\mathbf{z}_1 \quad \cdots \quad U\mathbf{z}_r \quad U\mathbf{z}_{r+1} \quad \cdots \quad U\mathbf{z}_n] \\ &= [\sigma_1 \mathbf{u}_1 \quad \cdots \quad \sigma_r \mathbf{u}_r \quad \mathbf{0} \quad \cdots \quad \mathbf{0}] \end{aligned}$$

$$\Leftrightarrow \mathrm{kol}_i(U\Sigma) = \begin{cases} \sigma_i \mathbf{u}_i, & \text{for } \ 1 \le i \le r \\ \mathbf{0}, & \text{for } \ r < i \le n \end{cases} \qquad (E1)$$

Nå skal vi skrive V^T slik:

$$V^T = \begin{bmatrix} \mathbf{v}_1{}^T \\ \vdots \\ \mathbf{v}_r{}^T \\ \mathbf{v}_{r+1}{}^T \\ \vdots \\ \mathbf{v}_n{}^T \end{bmatrix} \qquad (E2)$$

Fra $(E1)$ og $(E2)$, får vi:

$$\begin{aligned} A = U\Sigma V^T &= \sum_{i=1}^{n} \mathrm{kol}_i(U\Sigma)\,\mathrm{rad}_i(V^T) \\ &= \sigma_1 \mathbf{u}_1 \mathbf{v}_1{}^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r{}^T + \mathbf{0}\mathbf{v}_{r+1}{}^T + \cdots + \mathbf{0}\mathbf{v}_n{}^T \\ &= \sigma_1 \mathbf{u}_1 \mathbf{v}_1{}^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r{}^T \\ &= \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i{}^T \end{aligned}$$

Oppgave 3

(a) I stedet for å bruke en løkke til å regne ut $A^{(k)}$, har jeg valgt å løse problemet ved å bruke "vectorization" slik at algoritmen blir mer effektiv (dvs. jeg vil bruke matrise multiplikasjon istedenfor en løkke). Her er MATLAB utskrift:

```
function AK = svdApprox(A,k)

AK = [];
if k < 1
    error('k must be a positive integer!');
    return;
end

[U,S,V] = svd(A);
try
    U  = U(:,1:k);      % extract only columns 1 to k in U
    S  = S(1:k,1:k);    % extract a k*k square matrix from S
    V  = V(:,1:k);      % extract only columns 1 to k in V
    AK = U*S*V';        % vectorization to calculate AK
catch ME
    if strcmp(ME.identifier, 'MATLAB:badsubscript')
        error('k cannot be bigger than the rank of A!');
    else
        rethrow(ME);
    end
end
```

(b) MATLAB utskrift:

```
>> A = double(imread('mm.gif', 'gif'));
>> size(A)
ans =
   256   256
>> rank(A)
ans =
   256
>> rank(A,0.001)
ans =
   256
```

(c) MATLAB utskrift:

```
>> A = double(imread('mm.gif', 'gif'));
>> mm8 = svdApprox(A,8);
>> mm32 = svdApprox(A,32);
>> imwrite(uint8(mm8), 'mm8.gif', 'gif');
>> imwrite(uint8(mm32), 'mm32.gif', 'gif');
```

$k = 8$ $k = 32$

(d) Siden bildet har mange gjentatte mønstre (alle punkter som ligger på samme "kolonne" har samme farge, og det finnes bare 5 "kolonner" i hele bildet, dvs. 5 gråtoner), så kan bildet komprimeres mye. Og da kan vi konkludere at matrisen til bildet har en veldig lav rang. For å være mer presis, kan vi bruke resultat vi fikk fra oppgave 1 til å utlede at denne matrisen må ha rangen lik 1.

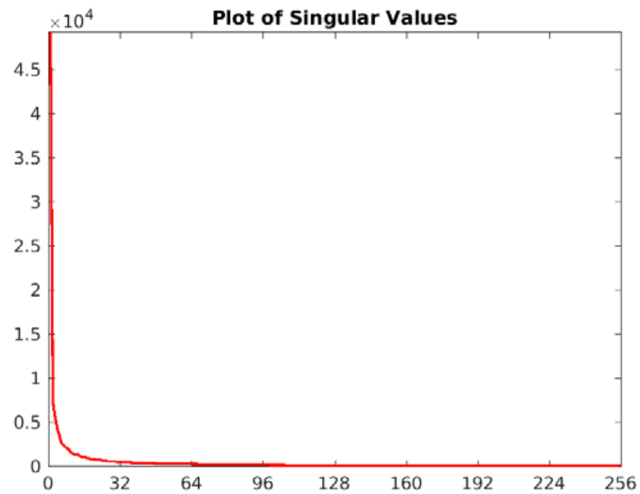
La en $m \times n$ matrise A være matrisen til bildet i figur 1. La $\mathbf{u} \in \mathbb{R}^m$ være en kolonne vektor hvor alle elementene er lik 1, dvs. $\mathbf{u} = \text{ones}(m, 1)$. Og la $\mathbf{v}^T \in \mathbb{R}^n$ være en rad vektor som er lik en tilfeldig rad i matrisen A. Da kan vi se at $A = \mathbf{u}\mathbf{v}^T$. Men siden vi har allerede vist i oppgave 1 at alle matriser på formen $\mathbf{u}\mathbf{v}^T$ (hvor vi antar at $\mathbf{u}$ og $\mathbf{v}$ ikke er lik $\mathbf{0}$) må ha rangen 1, da følger det at matrisen som beskriver bildet må ha rangen 1.

Oppgave 4

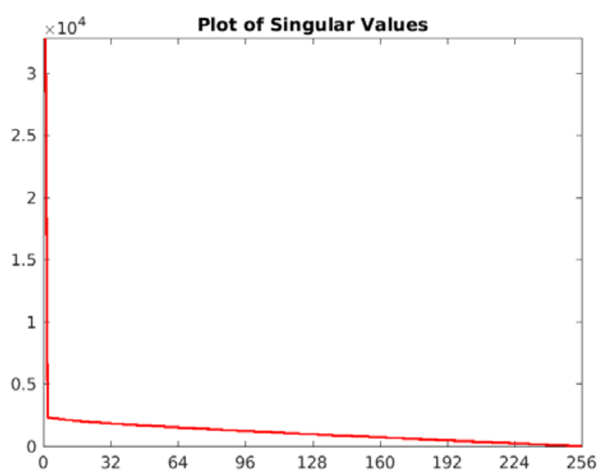
(a) MATLAB version 2016a

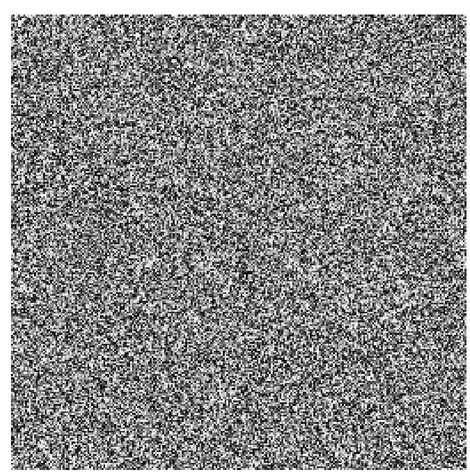
```
A = double(imread('mm.gif', 'gif'));
[U,S,V] = svd(A);
s = diag(S); % vector of all singular values

figure;
plot(s, 'r', 'LineWidth', 1.5);      % plot the line
set(gca, 'XTick', 0:32:length(s));  % put the ticks at every 32nd place
axis([0,256,0,inf]);                % set the x-axis limit
title('Plot of Singular Values');
print('plot1', '-dpng');
```



(b) MATLAB version 2016a:





Plottet i oppgave $4a$ avtar raskere enn plottet i oppgave $4b$. Dette betyr at det er flere singulærverdier som er lik 0 (eller omtrent 0) i bildet til Marilyn Monroe enn i bildet med tilfeldige punkter. Siden antall positive singulærverdier er lik rangen til matrisen, da kan vi konkludere at bildet til Marilyn Monroe har lavere rang enn det tilfeldige bildet. Merk at dette her er bare en tilnærming når toleransen ε er stor nok (f.eks. $\varepsilon = 0.1$):

```
>> A = double(imread('mm.gif', 'gif'));
>> B = round(255*rand(256,256));
>> rank(A, 0.1)
ans =
   254
>> rank(B, 0.1)
ans =
   256
```

Som vi kan se på MatLab utskrift ovenfor, hvis $\varepsilon = 0.1$ da får vi en mindre rang ($r = 254$) for Marilyn Monroe bildet, mens rangen til det tilfeldige bildet er fortsatt 256. Det vil si at det er vanskeligere å komprimere det tilfeldige bildet. Dette her er åpenbart siden det ikke finnes mange gjentatte mønstre når vi ser på et bild med tilfeldige punkter.

Oppgave 5

(a) SVD til en $m \times n$ matrise A er $A = U\Sigma V^T$, hvor U og V er $m \times m$ og $n \times n$, henholdsvis. Siden vi trenger k kolonner fra U og k kolonner fra V, samt med k singulær verdier til å regne ut $A^{(k)}$, da er antall tall vi må lagre er: $km + kn + k = k(m + n + 1)$. Siden A er $m \times n$, så har vi mn punkter i det originale bildet. Da kan vi regne ut den "compression ratio" slik:

$$Ratio = \frac{mn}{k(m + n + 1)}$$

Hvis k er små, vil vi få ratio > 1, og det betyr at komprimering gir et bildet med mindre størrelse. Hvis vi velger $k = 80$ for Marilyn Monroe bildet (jeg synes at 80 er godt nok), da er ratio lik:

$$Ratio = \frac{256 \times 256}{80 \times (256 + 256 + 1)} = 1.6$$

(b) Feilen for $k = 80$ er lik 0.0134. Her er MATLAB utskrift:

```
function error = relError(A, AK);
f = @(M) sqrt(sum(sum(M.*M))); % this computes |M| for a matrix M
error = f(A-AK) / f(A);
```

```
>> A = double(imread('mm.gif', 'gif'));
>> AK = svdApprox(A,80);
>> error = relError(A, AK)
error =
    0.0134
```