MA398 Assignment 2

1. $A = \begin{pmatrix} 5 & 3 & 4 \\ 3 & 5 & -4 \end{pmatrix}$

$A^T = \begin{pmatrix} 5 & 3 \\ 3 & 5 \\ 4 & -4 \end{pmatrix}$

$AA^T = \begin{pmatrix} 50 & 14 \\ 14 & 50 \end{pmatrix}$; $A^T A = \begin{pmatrix} 34 & 30 & 8 \\ 30 & 34 & -8 \\ 8 & -8 & 32 \end{pmatrix}$

$\lambda_1 = 64$ $\quad\quad$ $\lambda_1 = 64$

$\lambda_2 = 36$ $\quad\quad$ $\lambda_2 = 36$

$\lambda_3 = 0$

Eigenvectors (unnormalised)

$\sigma_1 = \sqrt{64} = 8$ $\quad\quad$ $\tilde{\underline{V}}_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \tilde{\underline{V}}_2 = \begin{pmatrix} 1 \\ -1 \\ 4 \end{pmatrix}, \tilde{\underline{V}}_3 = \begin{pmatrix} -2 \\ 2 \\ 1 \end{pmatrix}$

$\sigma_2 = \sqrt{36} = 6$

Normalising:

$\Rightarrow \Sigma = \begin{pmatrix} 8 & 0 & 0 \\ 0 & 6 & 0 \end{pmatrix}$

$\Rightarrow \underline{V}_1 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{pmatrix}, \underline{V}_2 = \begin{pmatrix} 1/\sqrt{18} \\ -1/\sqrt{18} \\ 4/\sqrt{18} \end{pmatrix}$

$\underline{V}_3 = \begin{pmatrix} -2/\sqrt{7} \\ 2/\sqrt{7} \\ 1/\sqrt{7} \end{pmatrix}$

PTO

So far: $A = U \Sigma V^T$

$$= U \begin{pmatrix} 8 & 0 & 0 \\ 0 & 6 & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ -2/\sqrt{7} & 2/\sqrt{7} & 1/\sqrt{7} \end{pmatrix}$$

Finding U:

$$\underline{u}_i = \frac{1}{\sigma_i} A \underline{v}_i$$

$$\Rightarrow \underline{u}_1 = \frac{1}{8} A \underline{v}_1 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$$

$$\underline{u}_2 = \frac{1}{6} A \underline{v}_2 = \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$$

No 3rd column

$$\Rightarrow A = \underbrace{\begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} 8 & 0 & 0 \\ 0 & 6 & 0 \end{pmatrix}}_{\Sigma} \underbrace{\begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ -2/\sqrt{7} & 2/\sqrt{7} & 1/\sqrt{7} \end{pmatrix}}_{V^T}$$

PTO

$$\underline{B} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad B^T = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

$$B B^T = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

$\lambda_1 = 2$
$\lambda_2 = 2$
$\lambda_3 = 0$
$\lambda_4 = 0$

$$\hat{\underline{v}}_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad \hat{\underline{v}}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

$$\hat{\underline{v}}_3 = \begin{pmatrix} -1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad \hat{\underline{v}}_4 = \begin{pmatrix} 0 \\ -1 \\ 0 \\ 1 \end{pmatrix}$$

Normalised

$$\underline{v}_1 = \begin{pmatrix} 1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \\ 0 \end{pmatrix}, \quad \underline{v}_2 = \begin{pmatrix} 0 \\ 1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{pmatrix}$$

$$\underline{v}_3 = \begin{pmatrix} -1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \\ 0 \end{pmatrix}, \quad \underline{v}_4 = \begin{pmatrix} 0 \\ -1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{pmatrix}$$

$$B^T B = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

$\lambda_1 = 2$
$\lambda_2 = 2$

$$\cancel{\Rightarrow \Sigma = \begin{pmatrix} \sqrt{2} & 0 \\ 0 & \sqrt{2} \\ 0 & 0 \\ 0 & 0 \end{pmatrix}}$$

$$\hat{\underline{v}}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \hat{\underline{v}}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$\sigma_1 = \sqrt{2}, \quad \sigma_2 = \sqrt{2}$

$$\Sigma = \begin{pmatrix} \sqrt{2} & 0 & 0 & 0 \\ 0 & \sqrt{2} & 0 & 0 \end{pmatrix}$$

$$\underline{v}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \underline{v}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

PTO

$$B^T = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \sqrt{2} & 0 & 0 & 0 \\ 0 & \sqrt{2} & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}^T$$

belongs to this

$$\Rightarrow B = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \sqrt{2} & 0 \\ 0 & \sqrt{2} \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}^T$$

$$= \underbrace{\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} \sqrt{2} & 0 \\ 0 & \sqrt{2} \\ 0 & 0 \\ 0 & 0 \end{pmatrix}}_{\Sigma} \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{V^T}$$

$$\boxed{B = \underbrace{\begin{pmatrix} 1/\sqrt{2} & 0 & -1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} & 0 & -1/\sqrt{2} \\ 1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} & 0 & 1/\sqrt{2} \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} \sqrt{2} & 0 \\ 0 & \sqrt{2} \\ 0 & 0 \\ 0 & 0 \end{pmatrix}}_{\Sigma} \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{V^T}}$$

2.

a) $p = \min(m, n)$

$$A = U \Sigma V^T = \sum_{j=1}^{p} \sigma_j u_j v_j^T$$

So this is:

* not necessarily square, but no other values are needed

$$\begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_p \end{pmatrix}^{*} \left(\underline{u}_1, \underline{u}_2, \ldots, \underline{u}_p \right) \begin{pmatrix} \underline{v}_1^T \\ \underline{v}_2^T \\ \vdots \\ \underline{v}_p^T \end{pmatrix}$$

Since matrix of singular values is all 0s except for the singular values themselves, they can be stored in a 1D array, so only p elements need to be stored, i.e. $\sigma_1 \ldots \sigma_p$

$\underline{u}_i$ are column vectors ~~of~~ of size $m \times 1$, so each of $\underline{u}_i$ has m elements, $m \times p$ need to be stored.

$\underline{v}_i^T$ are row vectors of size $1 \times m$, so each of $\underline{v}_i^T$ has n elements, $p \times n$ need to be stored

So ~~for~~ in total for A, $\boxed{p + mp + np}$ elements need to be stored

For A_k, $\boxed{k + mk + nk}$ elements need to be stored

b) Firstly, proving the existence of a rank k matrix ~~with~~ A_k such that

$$\|A - A_k\|_2 = \sigma_{k+1}$$

As in (a):

$$A_k = \sum_{j=1}^{k} \sigma_j u_j v_j^T$$

$$\Rightarrow A - A_k = \sum_{j=k+1}^{m} \sigma_j u_j v_j^T$$

$$\Rightarrow \|A - A_k\|_2 = \sigma_{max}\left(\sum_{j=k+1}^{m} \sigma_j u_j v_j^T\right) = \sigma_{k+1} \quad *$$

$$\left(\max_x \frac{\|Ax\|_2}{\|x\|_2} = \max_x \sqrt{\frac{\|Ax\|_2^2}{\|x\|_2^2}} = \sqrt{\frac{x^T A^T A x}{x^T x}} \right.$$

$$\left. = \sqrt{\lambda_{max}(A^T A)} = \sigma_{max}(A) \right)$$

Next, we prove $\forall$ rank* k A_k, we have

$$\|A - A_k\|_2 \geq \sigma_{k+1}$$

* i.e. arbitrary k

so:

Assume A_k is an arbitrary rank k matrix

$$\forall A \in \mathbb{R}^{m \times n} : \text{rank}(A) + \text{kernel}(A) = n$$

$$\Rightarrow \text{kernel}(A_k) = n - \text{rank}(A_k) = n - k$$

$$\sigma_{k+1}(A)^2 = \lambda_{k+1}(A^T A)$$

PTO

$$\lambda_{k+1}(A^TA) = \min_{S:\dim(S)=n-k} \ \max_{x\in S} \frac{x^TA^TAx}{x^Tx}$$

$$\cancel{\leq \max_{x\in \text{kernel}(A_k)} \frac{x^T(A-A_k)^T(A-A_k)x}{x^Tx}}$$

$$\cancel{\leq \max}$$

$$\leq \max_{x\in \text{kernel}(A_k)} \frac{x^TA^TAx}{x^Tx}$$

$$= \max_{x\in \text{kernel}(A_k)} \frac{x^T(A-A_k)^T(A-A_k)x}{x^Tx} \quad ***$$

$$\leq \max_{x} \frac{x^T(A-A_k)^T(A-A_k)x}{x^Tx}$$

$$= \lambda_{max}\left((A-A_k)^T(A-A_k)\right) = \sigma_{max}(A-A_k)^2$$

$***$ $A_k x = 0$ for $x \in \text{kernel}(A_k)$

c) MA398_Assignment2_Exercise2.m

I used the default output of rgb2gray in matlab as it seems pointless to scale it between [0, 1]

I believe it stores between 0 and 100

This also comes with the benefit of not having to worry about scaling it.

d) For $k=0$, all images will just be black as there is no data

For the cat picture:

Rank 1: You can see generally which parts of the image are brighter and which are darker. No detail can be resolved, however.

Rank 10: You can see the shape of the cat, ~~as~~ its eyes, ears and body are discernible.

Rank 40: You can quite clearly see more detail, the cat's pupils are visible, ~~fur~~ is the fur and direction of the fur are visible

Rank $k=p=1791$:

This is just basically the original, there is no way to tell the difference

The resolution of the original is 1806×1791, hence $p = 1791$

For the diagonal lines:

Rank 1:

As with the cat, you can kind of tell ~~ish~~ which areas of the picture are lighter and darker, but due to the structure of a diagonal matrix, it is impossible for this to be more precise than it is

Rank 10:

You can see in a lot more detail the shape of the original image, however it is blurry and blocky as well as inaccurately coloured

Rank 40:

Much more precise, follows the shape of the original almost perfectly, however the colouring is still not perfect.

Rank $k=p=230$:

You can't tell the difference

resolution of original is 235×230 so $p=230$

For the vertical lines:

Rank 1: Identical to the original image

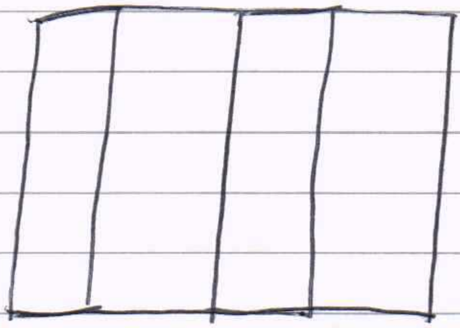
~~This is because~~

e) Talking generally, a diagonal ~~matrix~~ matrix would be harder to compress than a vertical (or ~~diag~~ horizontal) matrix.

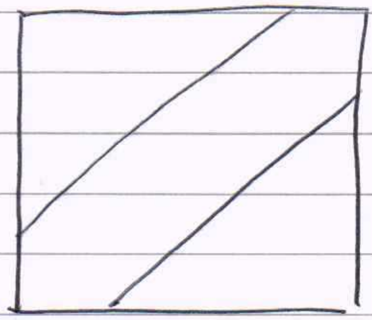
~~This is because compression~~

Someone with no knowledge of compression can clearly tell that it ~~works~~ ends up making images blocky, blocks in vertical lines make vertical rectangles - i.e. vertical lines with thickness, so imprecise compression can easily ~~sh~~ create accurate results. For diagonal lines, it would end up being like a staircase.

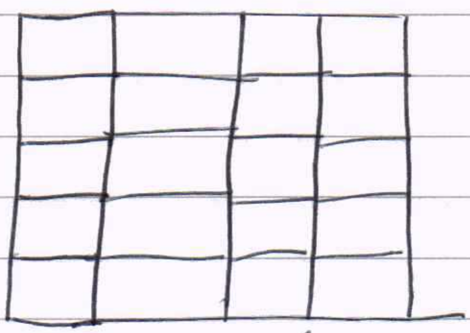
Block diagrams of images:



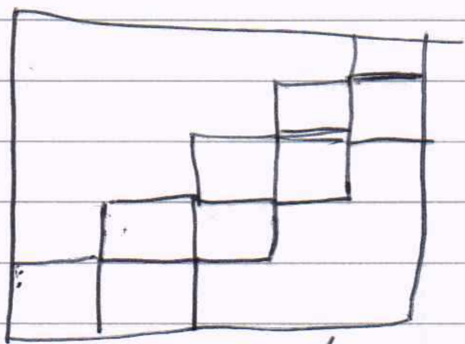
Original vertical



Original diagonal



Compressed Vertical



Compressed Diagonal

PTO

If we take the vertical one, we can see that the outlined blocks form the same, identical shape, whereas the ~~diagonal~~ diagonal forms a sort of staircase.

This is due to ~~its~~ its decomposition being of smaller dimensions. ~~This means~~ A diagonal line in any image will form a staircase, however ~~at~~ at higher and higher dimensions, i.e. higher resolutions, this is less and less visible. This is because pixels, or matrix entries themselves are blocks of size 1x1. ~~,~~

Another way to think of this is that the vertical lines are of the structure:

$$\begin{pmatrix} a_1 & a_2 & \cdots & a_n \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ a_1 & a_2 & \cdots & a_n \end{pmatrix} = A \in \mathbb{R}^{m \times n}$$

which can be written as

$(\underline{a}_1\ \underline{a}_2 \ldots \underline{a}_n)$ with $\underline{a}_i$ being columns of size m, with only ~~a_i~~ a_i as each element.

~~So~~ So a column of 1s $\in$ size n multiplied by a row of a_i, i.e.

$$\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} (a_1 a_2 \ldots a_n) = A$$

is identical to the original matrix

PTO

Obviously with SVD, this effect is not identical to how I have proposed.

Diagonal lines, on the other hand, are much harder to compress because the image is in the form:

$$
\begin{matrix}
a_1 & a_2 & a_3 & \cdots & \cdots & a_{n} \\
a_2 & a_3 & & & & \\
a_3 & & & & & \\
\vdots & & & & & \\
a_m & \cdots & \cdots & \cdots & \cdots & a
\end{matrix}
$$

$$
\begin{pmatrix}
a_1 & a_2 & a_3 & \cdots & a_{235} \\
a_2 & a_3 & & & \vdots \\
a_3 & & & & \vdots \\
\vdots & & & \ddots & \vdots \\
a_{230} & \cdots & \cdots & \cdots & a_{464}
\end{pmatrix} = A \in \mathbb{R}^{230 \times 235}
$$

This requires 464 separate values, i.e. $m+n-1$, whereas vertical lines only required n elements – so even for the same resolution, diagonal lines require a lot more data to store them even uncompressed.

f) ~~At~~ 10 Cat:

At ~~about~~ $k=10$, so rank 10 approx. you can tell that the image is a cat, the detail is hard to ~~distinguish though~~ see through, but you can see what's going on

At around $k=100$ it is so close, but with a high resolution display, you can find quite a few differences. With a lower resolution display, I ~~p~~ probably couldn't tell the difference. The background is not as smooth.

This persists until ~ $k \approx 180$ when it's not possible for me to tell ~~an~~ any difference.

Lines Diagonal:

At about rank 5, you can see the general shape but I would say rank 10 is the point at which I would say I'm certain it is the same image as the original.

This image's compression suffers from ~~low res~~ as it would be harder to spot differences with a lower resolution display, at $k=80$, I can still see ~~lines~~ differences. This persists up until about rank 120 where it is hard to tell any difference

Vertical Lines:

Rank 1 approx is identical to the original image

9) To quantify the quality, we can call quality the difference ~~of~~ the norms of the original image.

$$\text{i.e. quality} = (\text{norm}(A) - \cancel{\text{norm}(A_k)} - \text{norm}(A - A_k)) \times 100 / \text{norm}(A)$$

to scale it as a percentage of A rather than some numbers

This is a full reference method.

See ~~ARA~~ matlab code.

Doing this for 1791 points is ridiculous So I have done ~~the~~ [1 - 120, then in steps of 10 ~~to~~ until 320, steps of 50 until 1670 and ~~then~~ then 1791. * in steps of 5

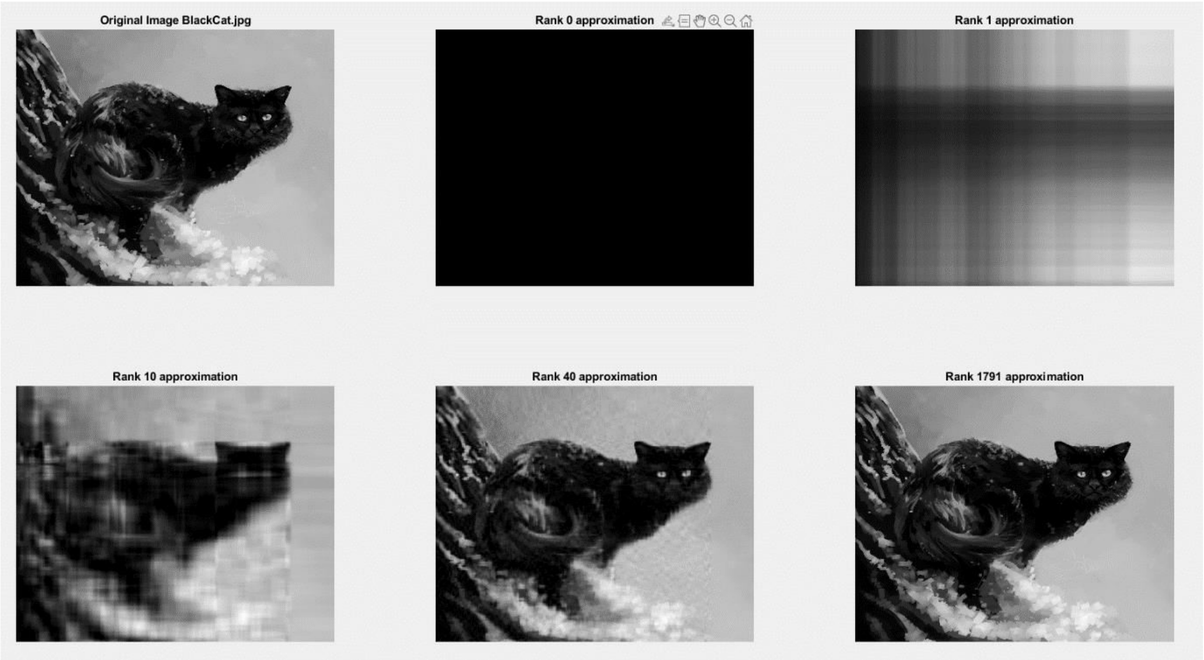
Perhaps a gpu ~~would take to~~ ~~into~~ could perform every single value within a reasonable amount of time. *

We can see that the increase of "detail" (by my defn.) is massive and is rapidly decreasing. The gradient of the line tends to 0 and it is at about $k \approx 150$ it is pretty much pointless to continue as most if not all people won't be able to see any difference.

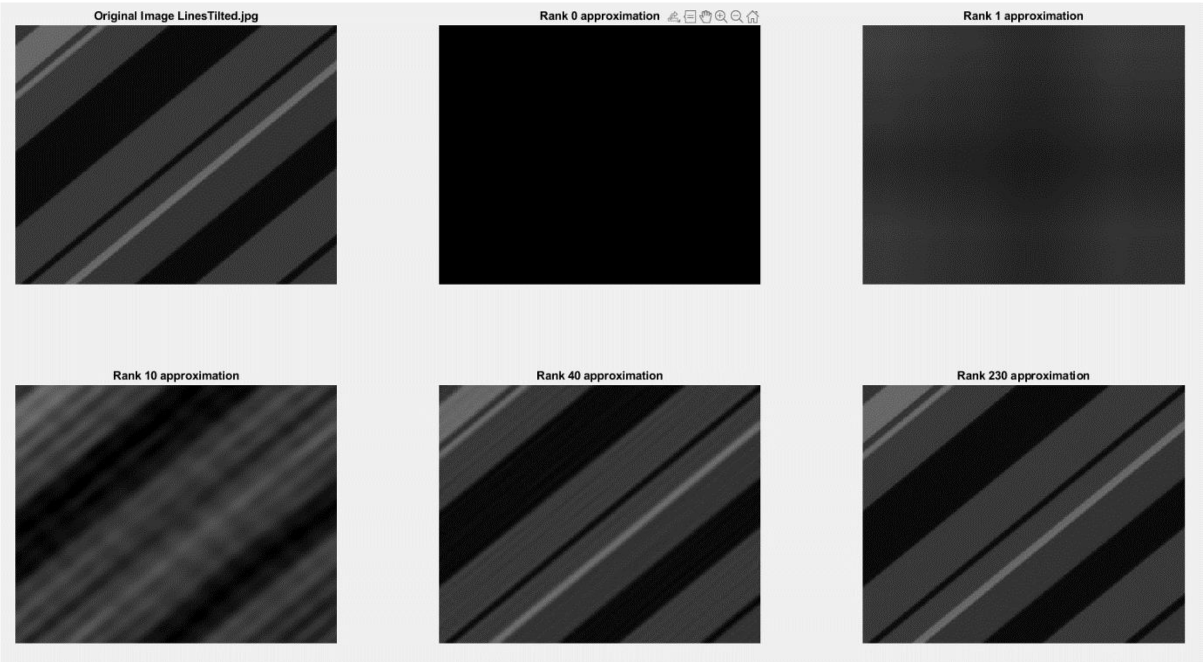
You could also suggest that the point where most people won't see any difference is ~~the~~ $k \approx 200$ as the gradient ~~is~~ close to 0. ($\approx 99.5\%$ of original detail)

I think more than 1% "detail" difference is enough to tell it's not the same

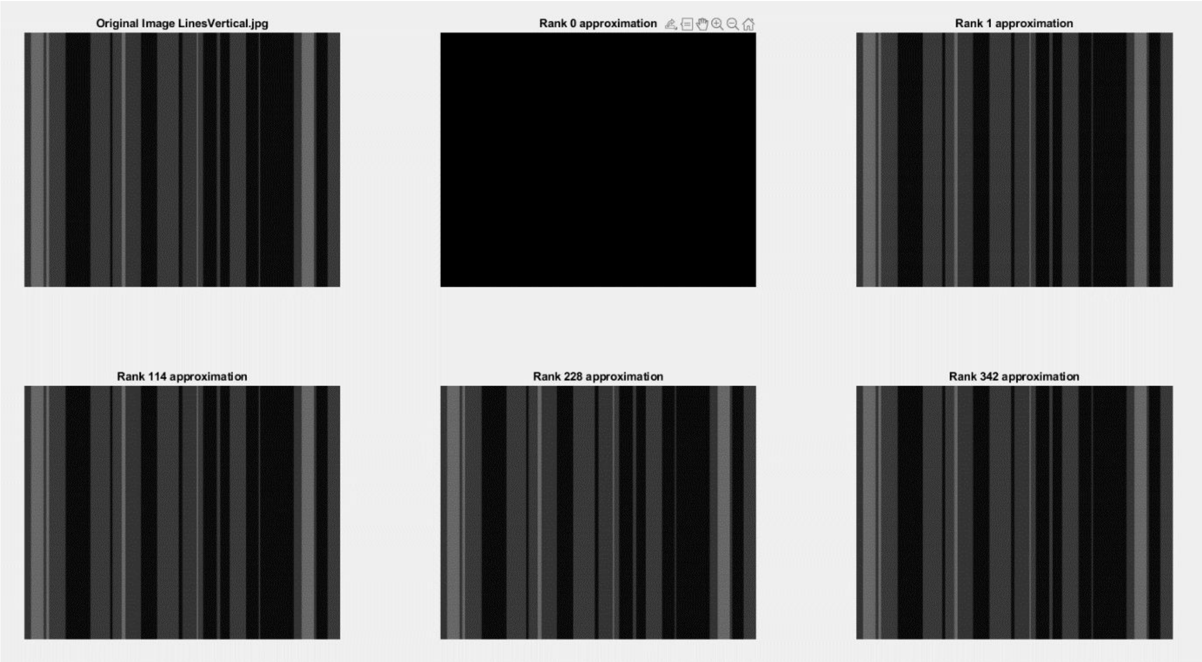
BlackCat.jpg

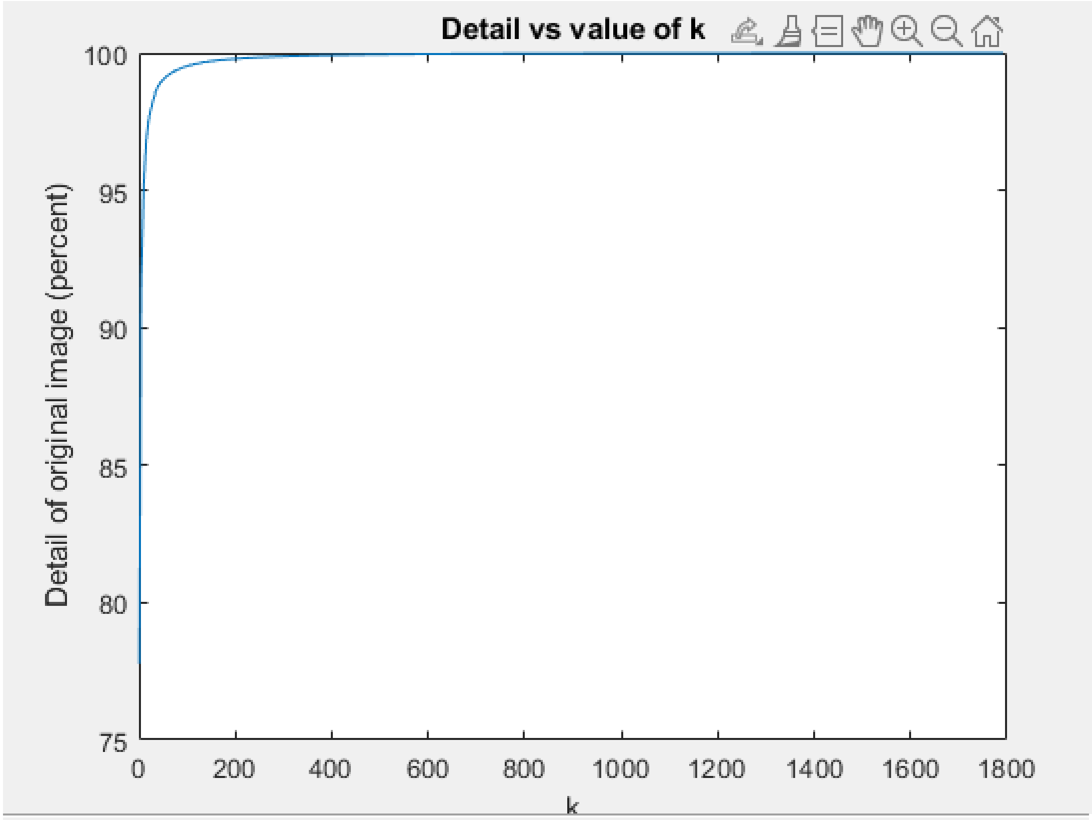


LinesTilted.jpg

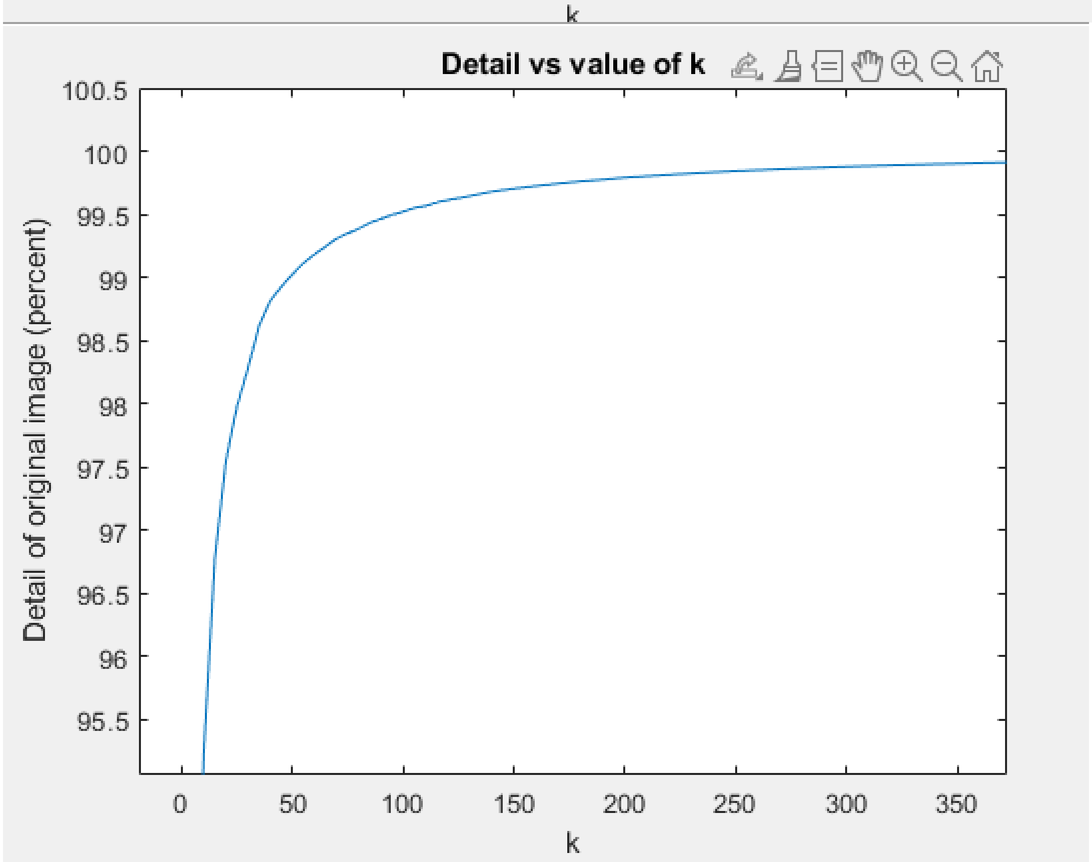


LinesVertical.jpg





Detail vs value of k
Detail of original image (percent)
100
95
90
85
80
75
0
200
400
600
800
1000
1200
1400
1600
1800
k



Detail vs value of k
Detail of original image (percent)
100.5
100
99.5
99
98.5
98
97.5
97
96.5
96
95.5
0
50
100
150
200
250
300
350
k

3.

a) vandermonde.m and MA398 .... Exercise 3.m

b) $p(x_i) = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$ ~~$= \begin{pmatrix} a_0 x_0 \\ a_1 x_1 \\ \vdots \\ a_n x_m^k \end{pmatrix}$~~ $y = a_0 + a_1 x + \dots$
$\dots + a_k x^k$
as k^{th} degree polynomial

Residual:

$$R^2 = \sum_{i=1}^{n} \left(y_i - (a_0 + a_1 x_i + \dots + a_k x_i^k) \right)^2$$

$$\frac{\partial (R^2)}{\partial a_0} = -2 \sum_{i=1}^{n} \left(y - (a_0 + a_1 x + \dots + a_k x^k) \right) = 0$$

$$\frac{\partial (R^2)}{\partial a_1} = -2 \frac{\partial (R^2)}{\partial a_0} x = 0$$

$\vdots$

$$\frac{\partial (R^2)}{\partial a_k} = -2 \frac{\partial (R^2)}{\partial a_{k-1}} x^k = 0$$

$$\Rightarrow a_0 n + a_1 \sum_{i=1}^{n} x_i + \dots + a_k \sum_{i=1}^{n} x_i^k = \sum_{i=1}^{n} y_i$$

$$a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + \dots + a_k \sum_{i=1}^{n} x_i^{k+1} = \sum_{i=1}^{n} x_i y_i$$

$$a_0 \sum_{i=1}^{n} x_i^k + a_1 \sum_{i=1}^{n} x_i^{k+1} + \dots + a_k \sum_{i=1}^{n} x_i^{2k} = \sum_{i=1}^{n} x_i^k y_i$$

$\sqrt{10}$

As ~~a m~~ matrices this is

$$\begin{pmatrix} n & \sum_{i=1}^{n} x_i & \cdots & \sum_{i=1}^{n} x_i^k \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \cdots & \sum_{i=1}^{n} x_i^{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} x_i^k & \sum_{i=1}^{n} x_i^{k+1} & \cdots & \sum_{i=1}^{n} x_i^{2k} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \vdots \\ \sum_{i=1}^{n} x_i^k y_i \end{pmatrix}$$

$*$

$*$ This is vandermonde matrix, hence it satisfies the given relation. (I used a_i instead of c_i but it's the same.

$\#\exists\!\!/$

c) Near the endpoints of $[-1,1]$, we see Runge's Phenomenon.

At $n=101$ it is a fine approximation until this issue occurs.

At 256 and 512, it goes crazy. This could be due to rounding errors.

d) Using polyfit, polyval to get coefficients and polyval to get y values

~~As~~ In the equality,

$$A = \begin{pmatrix} q(x_1) \\ q(x_2) \\ \vdots \\ q(x_d) \end{pmatrix}, \quad b = y \text{ values of } q(x_i)$$

~~My implementation would~~

~~Doing $V^T V$~~ Least squares can be done like this

$$V \underbrace{\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix}}_{\underline{a}_i} = \underbrace{\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}}_{\underline{y}_i}$$

$$V^T V \underline{a_i} = V^T \underline{y_i}$$

PTO

So
$$\begin{pmatrix} n & \sum_1^n x_i & \cdots & \sum_1^n x^k \\ \sum_1^n x_i & \sum_1^n x_i^2 & \cdots & \sum_1^n x_i^{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_1^n x_i^k & \sum_1^n x_i^{k+1} & \cdots & \sum_1^n x_i^{2k} \end{pmatrix} \underline{a_i} = \begin{pmatrix} \sum_1^n y_i \\ \sum_1^n x_i y_i \\ \vdots \\ \sum_1^n x_i^k y_i \end{pmatrix}$$

$$\Rightarrow \underline{y_i} = \underbrace{\begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^k \\ 1 & x_2 & x_2^2 & \cdots & x_2^k \\ \vdots & \vdots & & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^k \end{pmatrix}}_{\text{call } \underline{X}} \underline{a_i}$$

$$\underline{y_i} = \underline{X}\,\underline{a_i}$$

$$\underline{X}^T \underline{y_i} = \underline{X}^T \underline{X}\,\underline{a_i}$$

numerically $\Rightarrow \underline{a_i} = (\underline{X}^T \underline{X})^{-1} \underline{X}^T \underline{y_i}$

k is order, so k=1 is linear solution etc.

Due to my implementation, I think it would be more effective to implement Chebyshev nodes instead

PTO

For arbitrary n, Chebyshev nodes in $[a, b]$ are defined as follows

$$x_k = \frac{1}{2}(a+b) + \frac{1}{2}(b-a)\cos\left(\frac{2k-1}{2n}\pi\right)$$

with $k = 1 \ldots n$

So for $[-1, 1]$, we have

$$x_k = \frac{1}{2}(-1+1) + \frac{1}{2}(1-(-1))\cos\left(\frac{2k-1}{2n}\pi\right)$$

$$= \cos\left(\frac{2k-1}{2n}\pi\right)$$

This arrives at a much more satisfying fit

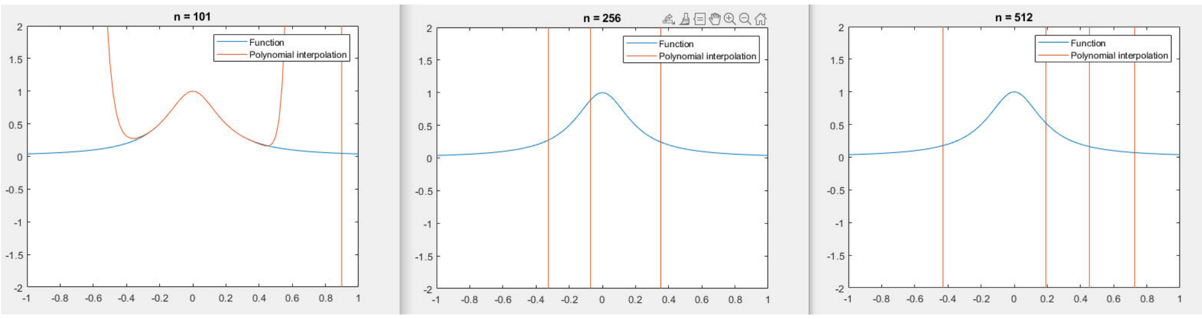
n isn't equivalent to d necessarily, so I will use values of n in $\{1, 5, 10, 20, 32, 64, 128\}$

At excessive n, i.e ≥ 64 (and possibly a bit lower) the fit is problematic.

However at $n = 32$, we get a perfect fit (or as close as a human can tell)

It gets closer as you increase n from 1, but as I mentioned, it eventually gets problematic.

Polynomial fit:



Chebyshev fit: