Linear Algebra 2

16171659

October 2018

1 My pic

2 Reading the file into an array

A = imread('mypic.jpg');

3 Extending to show matrix is a graphic image

```
image(A);
```

```
% converting to double precision real
a = double(A);

%breaking into each colour matrix
rk = a(:,:,1); %getting red matrix
gk = a(:,:,2); %getting green matrix
bk = a(:,:,3); %getting blue matrix

%checking to see of all 3 colour matrices create original image
[m,n,p] = size(a);
zk = zeros(m,n,p);
zk(:,:,1) = rk;
zk(:,:,2) = gk;
zk(:,:,3) = bk;
zk8bit = uint8(zk);
image(zk8bit);
```

4 SVD

4.1 Using built in command

```
[ur,sr,vr] = svd(rk);
[ug,sg,vg] = svd(gk);
[ub,sb,vb] = svd(bk)
```

4.2 usv.m

```
function[u,s,v] = usv(a)
[m,n] = size(a);
if m<n
    error('call with a* as argument')
end
as = a' ;
asa = as*a;
[v,s2] = eig(asa);
s = sqrt(s2);
ds = diag(s);
% sort the singular values into descreasing order
[dss,is] = sort(ds,1,'descend');
s = diag(dss);
% and apply the same sort to the column of v
v = v(:,is);
usigma = a*v;
u =usigma/s;
```

4.3 qrsvd.m

```
function [u,s,v] = qrsvd(a)
[m,n] = size(a);
p = min(m,n);
if m<n
    error('call with a* as argument')
end;
asa = a' * a;
[v,s2] = eig(asa);
s = sqrt(s2);
ds = diag(s);
%sort the singular values into decreasing order
[dss,is] = sort(ds,1,'descend');
s = diag(dss);
%applying same sort to columns of v
v = v(:,is);
av= a*v;
[q,r] = qr(av,0);
d = diag(r);
dphase= diag(s/r);
s = abs(r); %replacing original s
dphase = diag(dphase);
u = q*dphase;
```

4.4 Calculating SVD using usv method

```
[ur_usv,sr_usv,vr_usv] = usv(rk);
[ug_usv,sg_usv,vg_usv] = usv(gk);
[ub_usv,sb_usv,vb_usv] = usv(bk);
```

4.5 Calculating SVD using qrsvd method

```
[ur_qr,sr_qr,vr_qr] = qrsvd(rk);
[ug_qr,sg_qr,vg_qr] = qrsvd(gk);
[ub_qr,sb_qr,vb_qr] = qrsvd(bk);
```

4.6 Comparing || L - USV* || using built in svd method

```
comp_r_built = norm( (rk) -  ur .* sr .* vr');
comp_b_built = norm( (bk) -  ub .* sb .* vb' );
comp_g_built = norm( (gk) -  ug .* sg .* vg' );

 Results
 comparing red with built in : comp_r_built   = 5.717188120069027e+04
 comparing green with built in : comp_g_built = 4.982654517087696e+04
 comparing blue with built in : comp_b_built  = 3.027133679015769e+04
```

4.7 Comparing || L - USV* || using built in usv method

```
comp_r_usv = norm( (rk) -  ur_usv .* sr_usv  .* vr_usv' );
comp_b_usv = norm( (bk) -  ub_usv .* sb_usv .* vb_usv' );
comp_g_usv = norm( (gk) -  ug_usv .* sg_usv .* vg_usv' );

 Results
 comparing red with usv : comp_r_usv   = 5.717188120069027e+04
 comparing green with usv : comp_g_usv = 4.982654517087698e+04
 comparing blue with usv  : comp_b_usv = 3.027133679015770e+04
```

4.8 Comparing || L - USV* || using built in qrsvd method

```
comp_r_qr = norm( (rk) - ur_qr.*sr_qr.*vr_qr');
comp_b_qr = norm( (bk) - ub_qr.*sb_qr.*vb_qr');
comp_g_qr = norm( (rk) - ur_qr.*sr_qr.*vr_qr');

 Results
 comparing red with qrsvd : comp_r_qr   = 5.717188120069027e+04
 comparing green with qrsvd : comp_g_qr = 4.982654517087698e+04
 comparing blue with qrsvd  : comp_b_qr = 3.027133679015770e+04
```

4.9 Result of comparing || L - USV* ||

```
Comparing built in method to usv & qrsvd method above, The differences are:
comp_r_built - comp_r_usv = 0
comp_r_built - comp_r_qr  = 0

comp_g_built - comp_g_usv = 7.345336029813283e+03
comp_g_built - comp_g_qr = 7.345336029813290e+03

comp_b_built - comp_b_usv = -3.637978807091713e-12
comp_b_built - comp_b_qr  = -3.637978807091713e-12
```

4.10 Result of compaing ||U*U -I|| and ||V*V-I||

```
 %Red
norm_built_u_r - norm_usv_u_r = 1.430740415031551e-09
norm_built_v_r - norm_usv_v_r =-1.713587050033993e-10

norm_built_u_r - norm_sq_u_r = -8.754235114594167e-10
norm_built_v_r - norm_sq_v_r = -1.713587050033993e-10

%green
norm_built_u_g - norm_usv_u_g =3.699109907273623e-10
norm_built_v_g - norm_usv_v_g = 6.969047561256048e-11

norm_built_u_g - norm_sq_u_g = -8.959921693474371e-11
norm_built_v_g - norm_sq_v_g = 6.969047561256048e-11

%blue
norm_built_u_b - norm_usv_u_b =-1.878854849479694e-10
norm_built_v_b - norm_usv_v_b =  -8.357758929378178e-13

norm_built_u_b - norm_sq_u_b = 1.427533646847223e-10
norm_built_v_b - norm_sq_v_b = -8.357758929378178e-13
```

4.11 Evaluating ||U*U - I|| and ||V*V - I|| using built in svd

```
% red
norm_built_u_r = norm (ur' .*ur - eye(474)); = 1.091270922056828
norm_built_v_r = norm (vr' .*vr - eye(474)); = 1.101764757369113
% green
norm_built_u_g = norm (ug' .*ug - eye(474)); = 1.091159610620917
norm_built_v_g = norm (vg' .*vg - eye(474)); = 1.101013452094330
%blue
norm_built_u_b = norm (ub' .*ub - eye(474)); = 1.091384910224177
norm_built_v_b = norm (vb' .*vb - eye(474)); = 1.103934997637305
```

4.12 Evaluating ||U*U - I|| and ||V*V - I|| using usv

```
% red
norm_usv_u_r = norm (ur_usv' .* ur_usv - eye(474)); = 1.091270920626088
norm_usv_v_r = norm (vr_usv' .* vr_usv - eye(474)); = 1.101764757540471
%blue
norm_usv_u_b = norm (ub_usv' .* ub_usv - eye(474)); = 1.091384910412063
norm_usv_v_b = norm (vb_usv' .* vb_usv - eye(474)); = 1.103934997638141
%green
norm_usv_u_g = norm (ug_usv' .* ug_usv - eye(474)); = 1.091159610251006
norm_usv_v_g = norm (vg_usv' .* vg_usv - eye(474)); = 1.101013452024639
```

4.13 Evaluating ||U*U - I|| and ||V*V - I|| using qrsvd

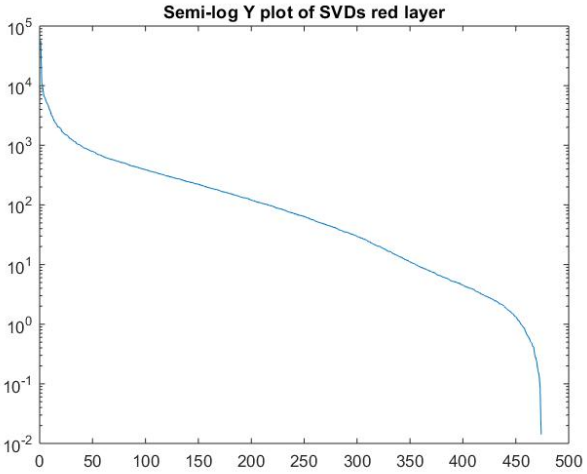
```
% red
norm_sq_u_r = norm (ur_qr' .* ur_qr - eye(474)); = 1.091270922932252
norm_sq_v_r = norm (vr_qr' .* vr_qr - eye(474)); = 1.101764757540471
% green
norm_sq_u_g = norm (ug_qr' .* ug_qr - eye(474)); = 1.091159610710516
norm_sq_v_g = norm (vg_qr' .* vg_qr - eye(474)); = 1.101013452024639
%blue
norm_sq_u_b = norm (ub_qr' .* ub_qr - eye(474)); = 1.091384910081424
norm_sq_v_b = norm (vb_qr' .* vb_qr - eye(474)); = 1.103934997638141
```

5 Semi log plots

Using usv.m method

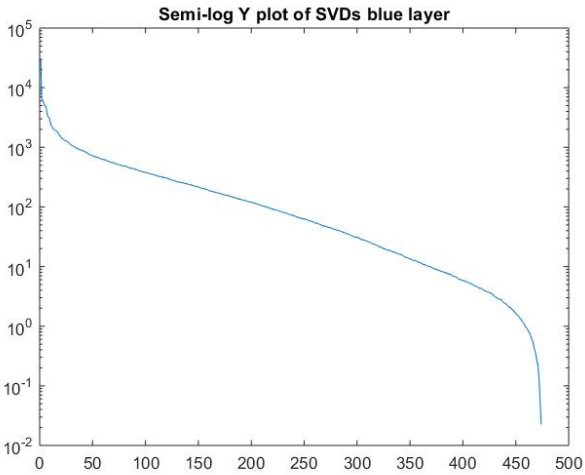
5.1 Red semi-log plot

```
%using usv
%red
figure('Name', 'semi log plot of red')
semilogy(diag(sr_usv));
title('Semi-log Y plot of SVDs red layer')
```



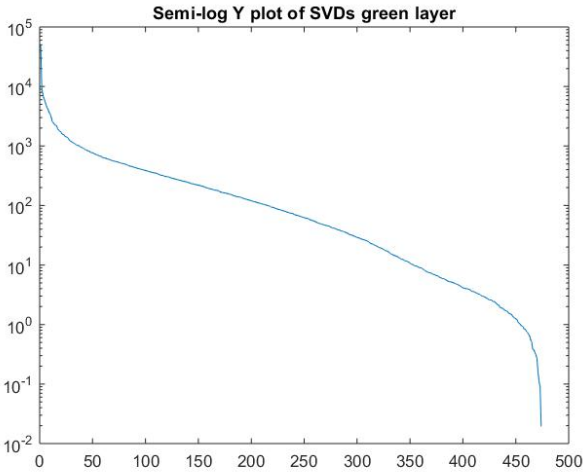
5.2 Blue semi-log plot

```
% blue
figure('Name', 'semi log plot of blue')
semilogy(diag(sb_usv));
title('Semi-log Y plot of SVDs blue layer')
```



5.3 Green semi-log plot

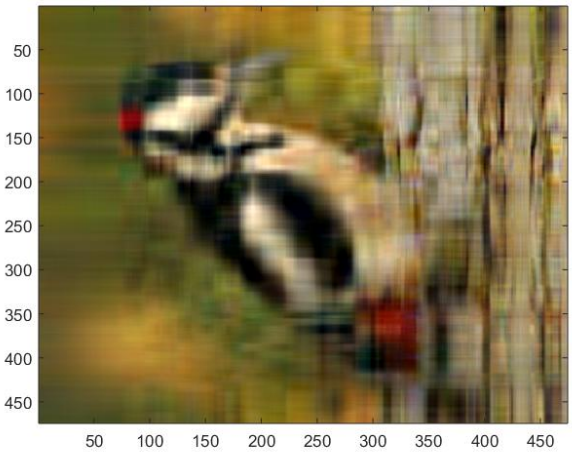
```
%green
figure('Name', 'semi log plot of green')
semilogy(diag(sg_usv));
title('Semi-log Y plot of SVDs green layer')
```



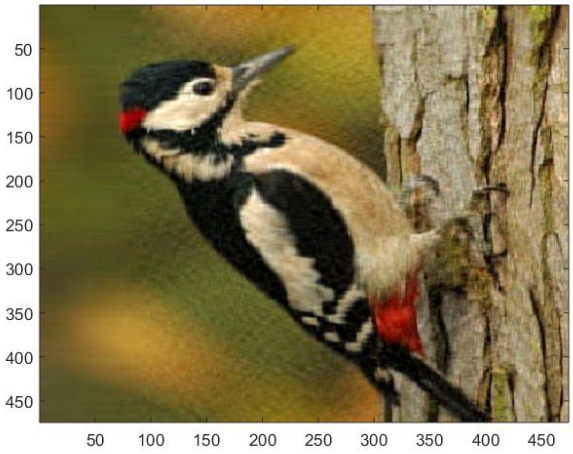
All semi log colour plots, look fairly similar and all decrease fairly rapidly, so a low rank approximation will capture most of the detail, by rank approximation 150 the two images are nearly identical. So I set $r_0 = 150$. For five equal steps from 10 to 150 we must move in steps of 35.

So now calculating A = USV' , but restricting size of matrices to n, where n is our rank approx.

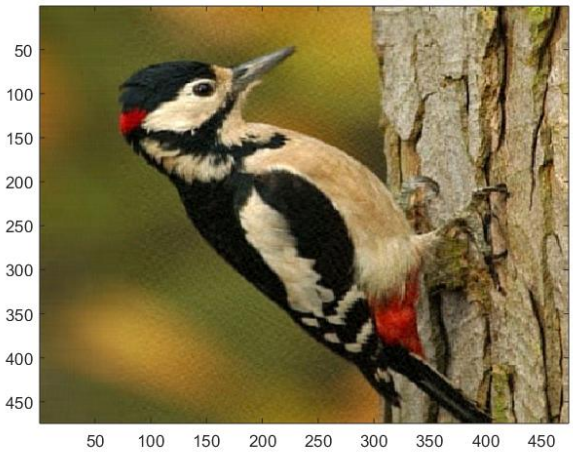
```
    %rank n approx
for  n = 10:35:150

vt_r = vr_usv';
vt_b = vb_usv';
vt_g = vg_usv';

%Calculating individual colours of A
outputimageR = ur_usv(:,1:n) * sr_usv(1:n,1:n)  * vt_r(1:n,:);
outputimageB = ub_usv(:,1:n) * sb_usv(1:n,1:n)  * vt_b(1:n,:);
outputimageG = ug_usv(:,1:n) * sg_usv(1:n,1:n)  * vt_g(1:n,:);

%Added all back together to construct rank approx image
constructedimage = cat(3,outputimageR,outputimageG,outputimageB);

%Showing image
figure('Name', 'low rank approx')
image(constructedimage/255)
end
```



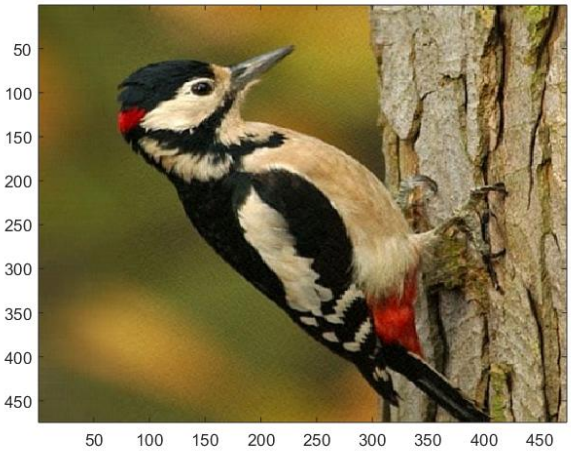
Rank 10



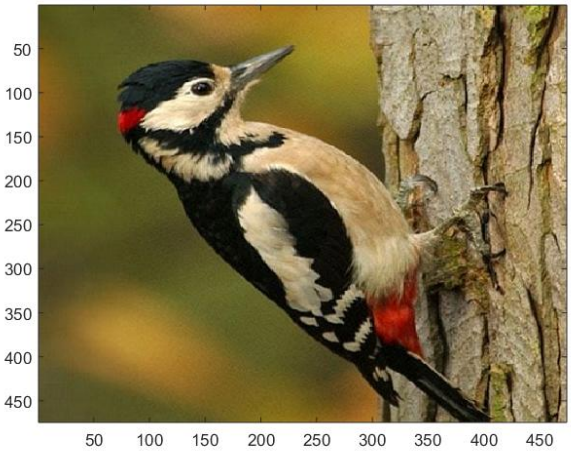
Rank 45



Rank 80



Rank 115



Rank 150