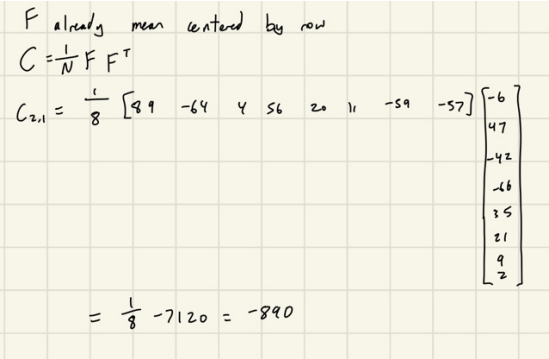
Homework #04

1

1a)



Represents the covariance between product 2 price and product 1 price

```
F=[ -6, 47, -42, -66, 35, 21, 9, 2;
89, -64, 4, 56, 20, 11, -59, -57];

[M, N] = size(F);
disp('Covariance Matrix:')
```

```
Covariance Matrix:
```

```
C=(1/8).*F * F.'
```

```
C = 2×2
10^3 ×
    1.2645   -0.8900
   -0.8900    2.8025
```

```
[U, lambda]=eig(C);
disp('Principal Components:')
```

```
Principal Components:
```

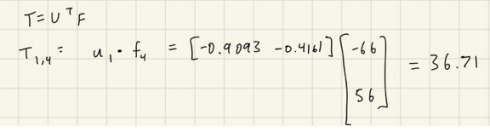
```
for i = 1:1:length(U)
    disp(U(:, i))
end
```

```
   -0.9093
   -0.4161

   -0.4161
    0.9093
```

```

```



Represents the projection of F data at time 4 onto the first PC (the amplitude of 1st PC in the data at time=4)

```
disp('Expansion Matrix:')
```

```
Expansion Matrix:
```

```
T=U.' * F
```

```
T = 2×8
   -31.5728   -16.1115    36.5280    36.7173   -40.1480   -23.6727    16.3632 ⋯
    83.4276   -77.7523    21.1117    78.3826     3.6249     1.2656   -57.3955
```

```
for i = 1:1:length(U)
    disp('For PC:')
    disp(U(:, i))
    disp(['variance explained = ' num2str(100*lambda(i,i)/sum(lambda(:))) '%'])
end
```

```
For PC:
   -0.9093
   -0.4161
variance explained = 21.0793%
For PC:
   -0.4161
    0.9093
variance explained = 78.9207%
```

1b) So the PC with the greatest eigenvalue, and therefore greatest variance explained, is:

```
disp('First PC:')
```

```
First PC:
```

```
disp(U(:,2))
```

```
   -0.4161
    0.9093
```

So this PC tells us that the two product price deviations are inversely correlated such that the first product price tends to be half as deviated and in the opposite direction as the second product price deviation.

So likely these are two products who are related in such a way that their prices are inversely correlated with each other.

1c) "Variance explained by each PCA mode" means how much of the variation in the F dataset can be attributed to each of the principal components (modes).

1d) So each row in F is a variable and therefore the variance for each row/variable m of F is:

$$\frac{1}{N}\sum_{n=1}^{N}(f_{mn})^2$$

Total variance is the sum of these for every row:

$$\sum_{m=1}^{M}(\frac{1}{N}\sum_{n=1}^{N}(f_{mn})^2) = \sum_{m=1}^{M}\sum_{n=1}^{N}\frac{1}{N}(f_{mn})^2$$

which is the trace/sum of eigenvalues of the covariance matrix:

$$= \text{trace}(\frac{1}{N}FF^T) = \text{trace}(C) = \sum_{j=1}^{M}(\lambda_j)$$

the fraction of explained variance for the ith PC is then:

$$\frac{\lambda_i}{\sum_{j=1}^{M}(\lambda_j)}$$

2

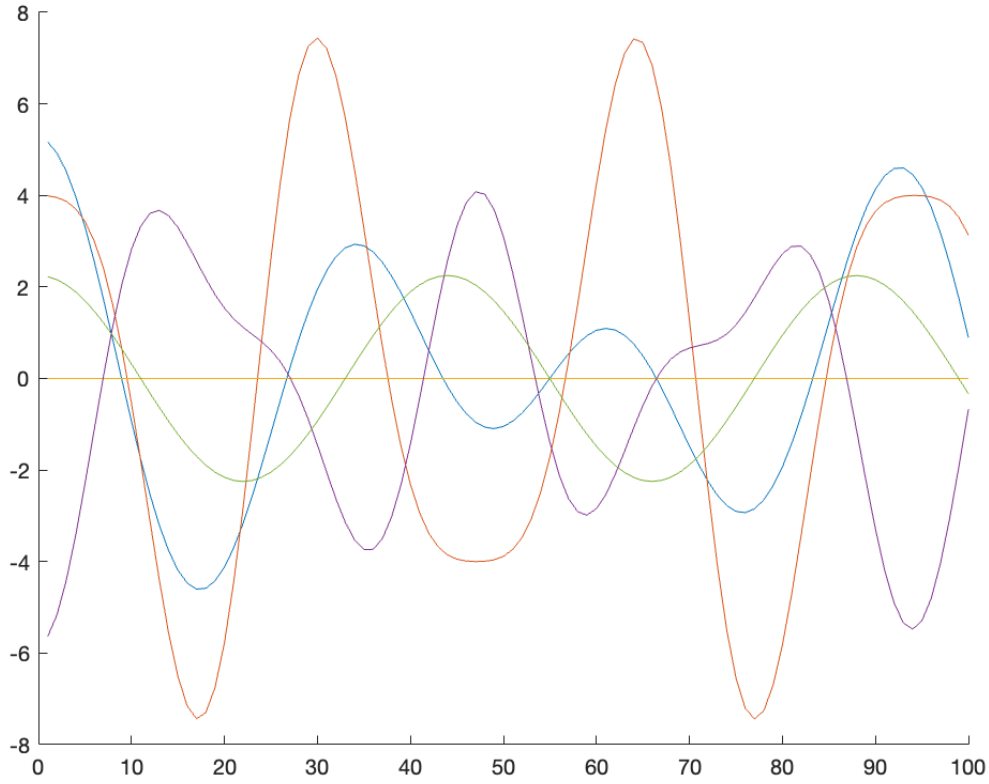
2a)

```
% Matlab
%
N=1300; t=1:N;
V1=[1;2;0;-1.1;0];
V2=[0;-1;0;-0.8;0];
V3=[1.5;0;0;-0.6;1.5];
F=3*V1*cos(t/5) ...
+2*V2*cos(t/3)...
+1.5*V3*cos(t/7);

[M, N] = size(F)
```

```
M = 5
N = 1300
```

```
figure; hold on
for i=1:1:M
    plot(t(1, 1:100), F(i, 1:100))
end
hold off
```



2b)

```
F_prime = F - mean(F,2);
disp('Covariance Matrix:')
```

```
Covariance Matrix:
```

```
C=(1/N).*F_prime * F_prime.'
```

```
C = 5×5
    6.9365    8.8994         0   -5.8796    2.4911
    8.8994   19.9962         0   -8.2378   -0.0898
         0         0         0         0         0
   -5.8796   -8.2378         0    7.0627   -0.9661
    2.4911   -0.0898         0   -0.9661    2.5354
```

Interpretation of covariance matrix: the matrix that contains the covariance of each stock's price with all the other stock prices and itself.

```
disp('Verifying C is symmetric:')
```

```
Verifying C is symmetric:
```

```
disp(C*C.')
```

```
   168.0892  287.8954         0 -158.0275   28.4769
   287.8954  546.9162         0 -275.1428   28.1051
          0         0         0         0         0
  -158.0275 -275.1428         0  153.2455  -23.1802
    28.4769   28.1051         0  -23.1802   13.5753
```

```
disp(C.'*C)
```

```
   168.0892  287.8954         0 -158.0275   28.4769
   287.8954  546.9162         0 -275.1428   28.1051
          0         0         0         0         0
  -158.0275 -275.1428         0  153.2455  -23.1802
    28.4769   28.1051         0  -23.1802   13.5753
```

$CC^T = C^TC$ so C is indeed symmetric.

The diagonal entries contain that specific stock price's own variance across the days, where C(i,i) is the variance for stock price i.

Stock 2 has the highest variance, then stock 4, then stock 1, then stock 5, and stock 3 does not vary/has 0 variance.

The non-diagonal entries above the diagonal contain the covariances between stock prices, so C(i,j) is the covariance between stock i price and stock j price.

Stock 1 is positively correlated with stocks 2 and somewhat with 5, and it is negatively correlated with stock 4.

Stock 2 is also negatively correlated with stock 4 and barely negatively correlated with stock 5.

Stock 4 is slightly negatively correlated with stock 5.

Stock 3 is not correlated with any of them.

2c)

```
[U, lambda]=eig(C);
disp('Principal Components:')
```

```
 Principal Components:
```

```
for i = 1:1:length(U)
    disp(U(:, i))
    disp(['with eigenvalue: ' num2str(lambda(i,i))])
end
```

```
   -0.4353
   -0.7969
         0
    0.4155
   -0.0531
 with eigenvalue: 29.1461
   -0.4977
    0.5057
         0
```

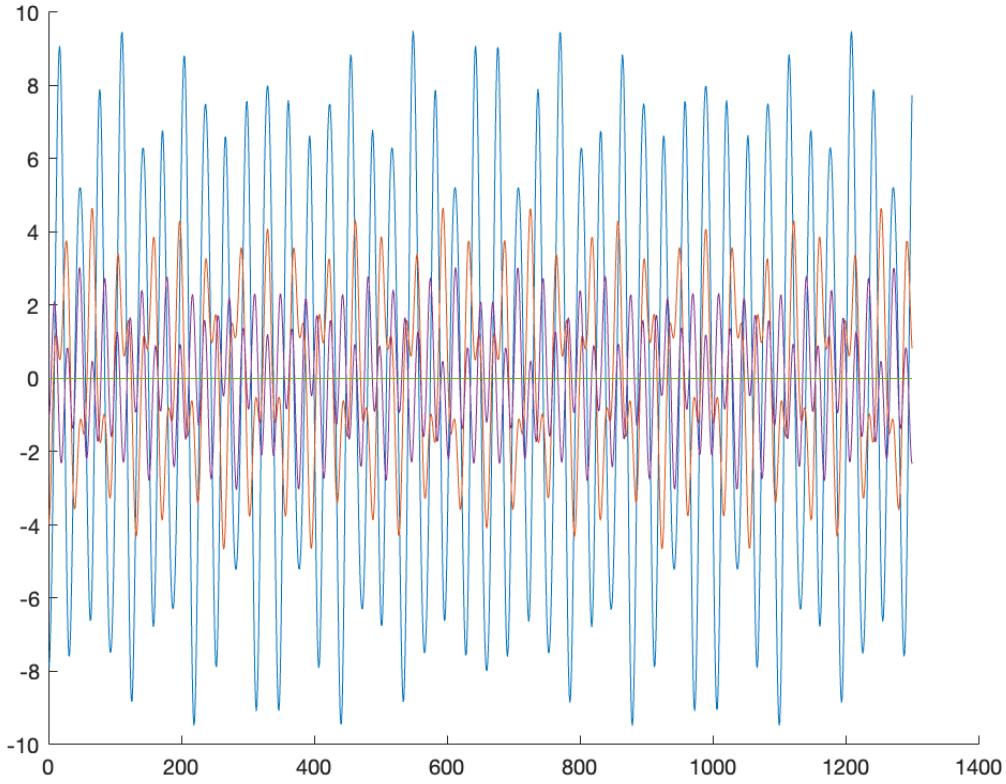
```
    0.3721
   -0.5984
with eigenvalue: 5.284
   -0.7160
    0.2121
         0
   -0.2652
    0.6099
with eigenvalue: -5.0555e-15
    0.2240
    0.2533
         0
    0.7865
    0.5168
with eigenvalue: 2.1008
     0
     0
     1
     0
     0
with eigenvalue: 0
```

For each PC, stock prices that are more strongly correlated with each other will be of similar sign and larger in magnitude. And then for each PC, the eigenvalue tells us how well it explains the total data variance.

2d)

```
T=U.'*F;

figure; hold on
n_N = N;
n=1:1:n_N;
for i = 1:1:M
    plot(n, T(i, 1:n_N))
end
hold off
```



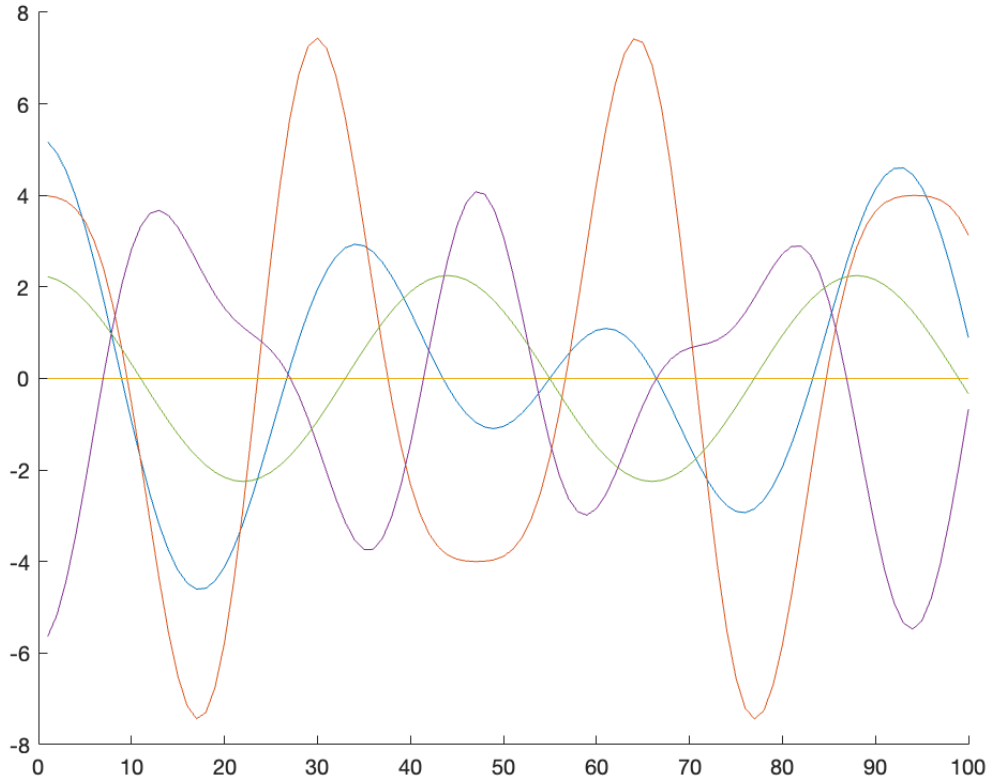
2e) 1st, 2nd, 3rd PC has significantly greater eigenvalue than the last two (which are basically 0). So we can use k=3.

```
reconst_PCs=[1,2,4];

F_reconst=U(:,reconst_PCs)*T(reconst_PCs, :)
```

```
F_reconst = 5×1300
    5.1673    4.9220    4.5225    3.9827    3.3209    2.5599    1.7256 ⋯
    3.9905    3.9546    3.8714    3.7098    3.4333    3.0064    2.4013
         0         0         0         0         0         0         0
   -5.6370   -5.1604   -4.4067   -3.4325   -2.3098   -1.1191    0.0580
    2.2271    2.1588    2.0465    1.8925    1.7000    1.4729    1.2157
```

```
figure; hold on
for i=1:1:M
    plot(t(1, 1:100), F_reconst(i, 1:100))
end
hold off
```

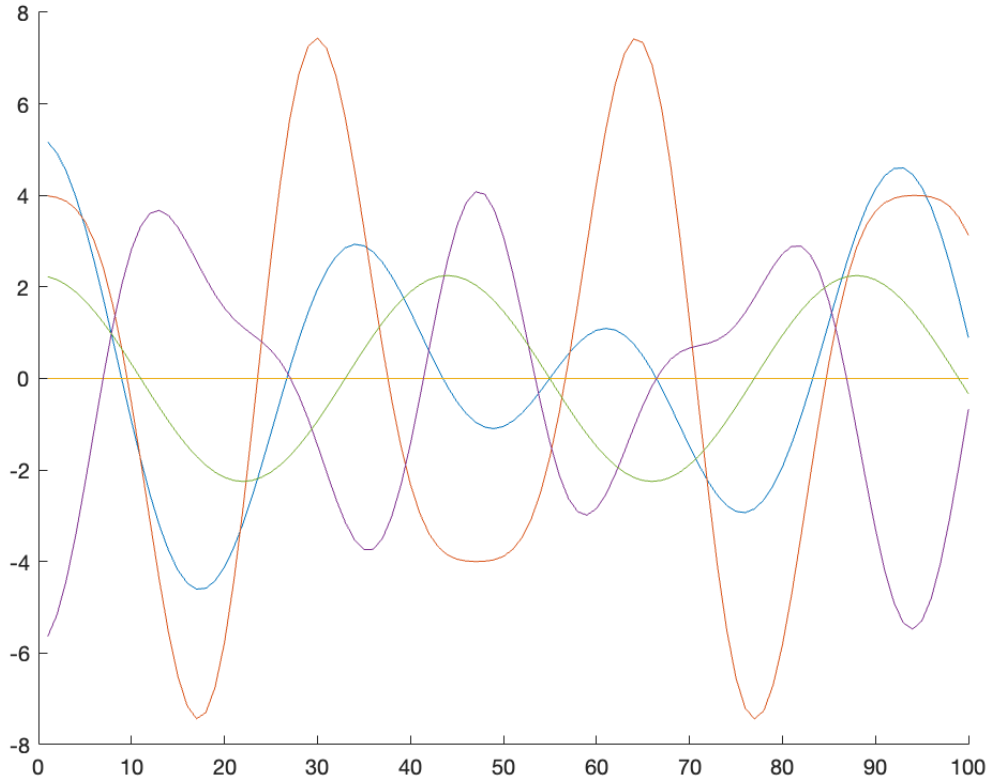


This reconstruction from 3 PC's has similar values and importantly, similar trends to F:

```
F
```

```
F = 5×1300
    5.1673    4.9220    4.5225    3.9827    3.3209    2.5599    1.7256 ⋯
    3.9905    3.9546    3.8714    3.7098    3.4333    3.0064    2.4013
         0         0         0         0         0         0         0
   -5.6370   -5.1604   -4.4067   -3.4325   -2.3098   -1.1191    0.0580
    2.2271    2.1588    2.0465    1.8925    1.7000    1.4729    1.2157
```

```
figure; hold on
for i=1:1:M
    plot(t(1, 1:100), F(i, 1:100))
end
hold off
```

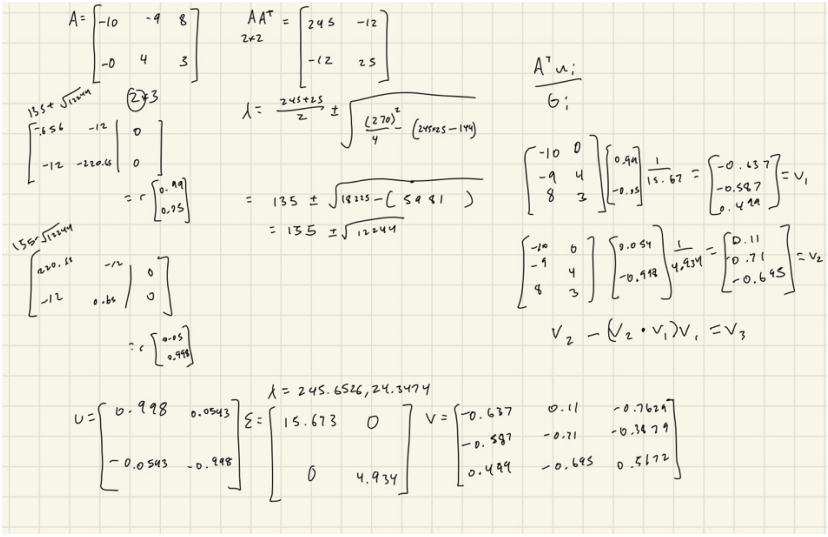


The data were originally constructed using 3 cosine function signals of varying amplitudes, so it makes sense that it is best reconstructed with 3 PCs.

3

3a)

A is 2x3 so A*A^T will have the smaller dimension and therefore will be used for this SVD.



```
A=[-10,-9,8;-0,4,3];

[U,S,V]=svd(A)
```

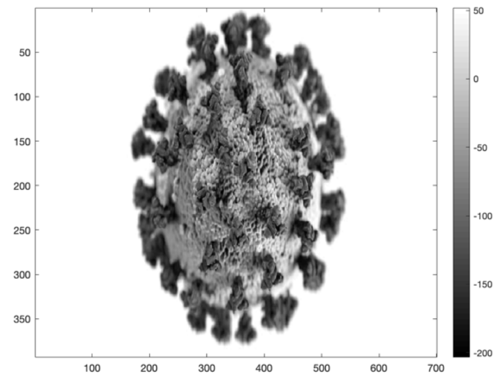
```
U = 2×2
    0.9985   -0.0543
   -0.0543   -0.9985
S = 2×3
   15.6733         0         0
         0    4.9343         0
V = 3×3
   -0.6371    0.1101    0.7629
   -0.5872   -0.7104   -0.3879
    0.4993   -0.6951    0.5172
```

They are the same

4

4a)

```
% Matlab
A=double(rgb2gray(imread('./coronavirus.jpg'))); A=A-mean(A(:));
figure(1); clf; imagesc(A); colormap(gray); colorbar
```



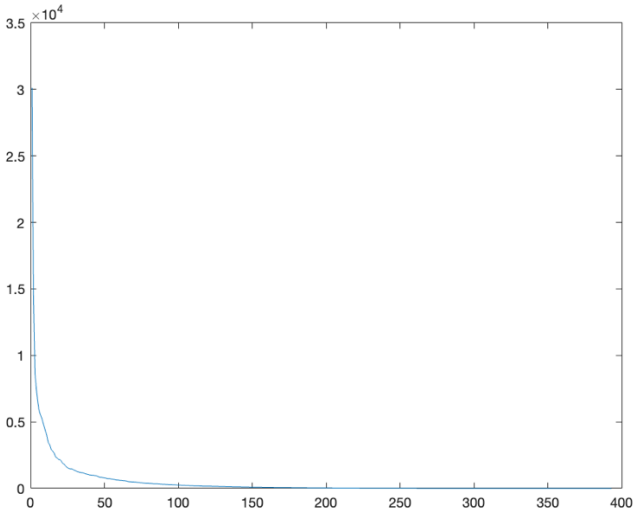

```
[M, N]=size(A);
[U,S,V]=svd(A)
```

```
U = 393×393
   -0.0063   -0.0625    0.0099   -0.0796    0.0304   -0.0600    0.0484 ⋯
   -0.0062   -0.0627    0.0100   -0.0797    0.0302   -0.0592    0.0485
   -0.0060   -0.0630    0.0098   -0.0796    0.0303   -0.0592    0.0489
   -0.0056   -0.0637    0.0094   -0.0796    0.0300   -0.0581    0.0497
   -0.0046   -0.0650    0.0088   -0.0801    0.0287   -0.0549    0.0510
   -0.0036   -0.0664    0.0083   -0.0806    0.0267   -0.0511    0.0525
   -0.0021   -0.0679    0.0083   -0.0820    0.0225   -0.0428    0.0536
   -0.0003   -0.0694    0.0090   -0.0841    0.0174   -0.0317    0.0540
    0.0019   -0.0709    0.0105   -0.0870    0.0114   -0.0176    0.0526
    0.0040   -0.0719    0.0130   -0.0906    0.0064   -0.0030    0.0507
      ⋮
S = 393×700
10⁴ ×
    3.0118         0         0         0         0         0         0 ⋯
         0    1.4580         0         0         0         0         0
         0         0    0.8663         0         0         0         0
         0         0         0    0.7432         0         0         0
         0         0         0         0    0.6421         0         0
         0         0         0         0         0    0.5737         0
         0         0         0         0         0         0    0.5461
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
      ⋮
V = 700×700
    0.0313   -0.0219    0.0072   -0.0202    0.0102   -0.0118    0.0130 ⋯
    0.0313   -0.0219    0.0072   -0.0202    0.0102   -0.0118    0.0130
    0.0313   -0.0219    0.0072   -0.0202    0.0102   -0.0118    0.0130
    0.0313   -0.0219    0.0072   -0.0202    0.0102   -0.0118    0.0130
    0.0313   -0.0219    0.0072   -0.0202    0.0102   -0.0118    0.0130
    0.0313   -0.0219    0.0072   -0.0202    0.0102   -0.0118    0.0130
    0.0313   -0.0219    0.0072   -0.0202    0.0102   -0.0118    0.0130
    0.0313   -0.0219    0.0072   -0.0202    0.0102   -0.0118    0.0130
    0.0313   -0.0219    0.0072   -0.0202    0.0102   -0.0118    0.0130
    0.0313   -0.0219    0.0072   -0.0202    0.0102   -0.0118    0.0130
      ⋮
```

```
m=1:1:M;
s=m;
for i = 1:1:M
    s(i) = S(i,i);
end
figure
disp('non-log plot:')
```

```
non-log plot:
```

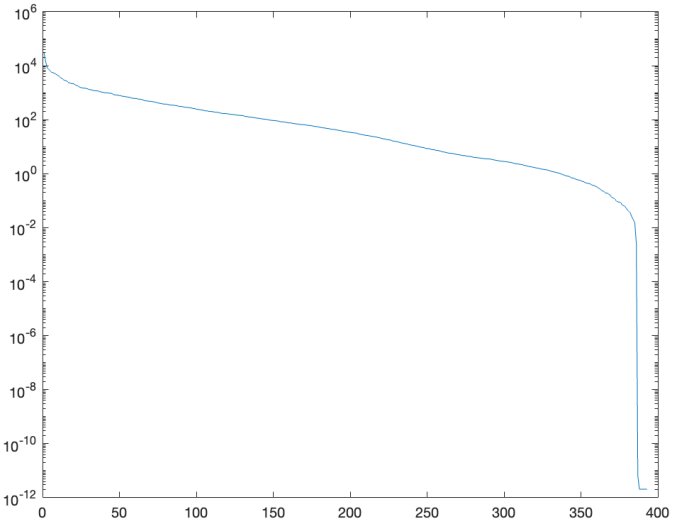
```
plot(m,s)
```




```
figure
disp('log10 plot:')
```

```
log10 plot:
```

```
semilogy(m,s)
```

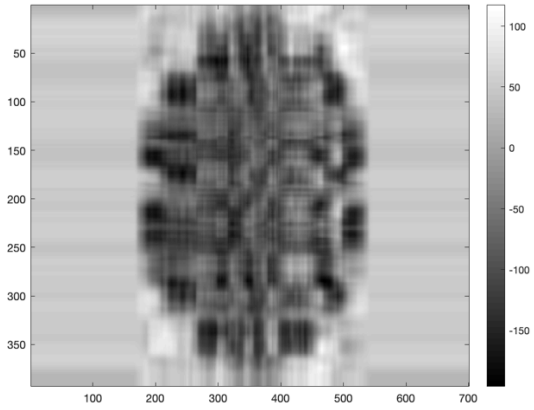


Looks like singular values get close to 0 after around 25. So I predict 25 modes.

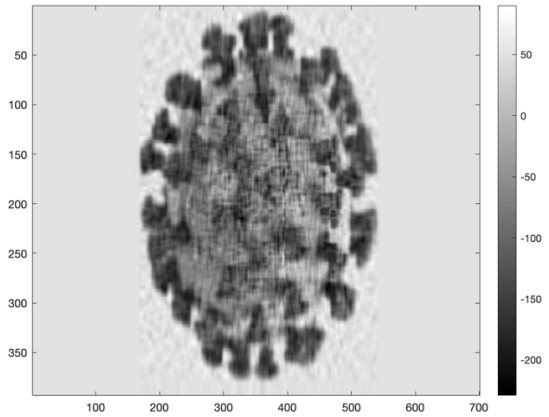
4b)

```
for n_modes = [5, 25, 50, 100]
    S_reconst = S;
    S_reconst(n_modes+1:end, n_modes+1:end) = 0;
    A_reconst = U*S_reconst*V.';
    disp(['# of modes: ' num2str(n_modes)])
    exp_var = 100.*sum(diag(S(1:n_modes,1:n_modes)).^2)/sum(diag(S(:,:)).^2);
    disp(['variance explained: ' num2str(exp_var) '%'])
    figure; imagesc(A_reconst); colormap(gray); colorbar
end
```

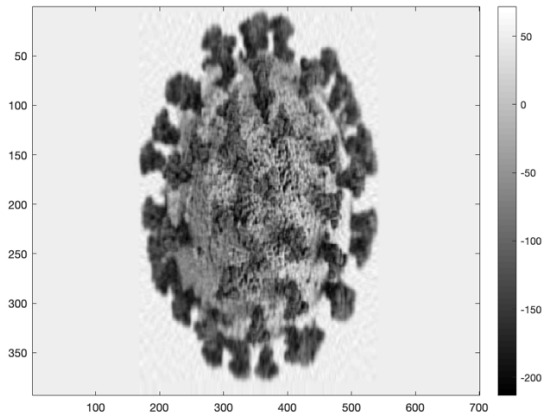
```
# of modes: 5
variance explained: 82.5591%
```



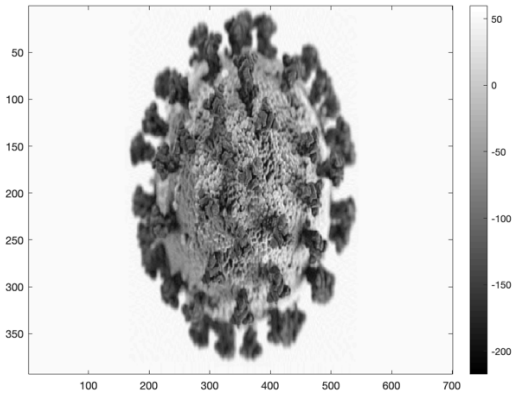

```
# of modes: 25
variance explained: 97.1502%
```




```
# of modes: 50
variance explained: 99.1807%
```




```
# of modes: 100
variance explained: 99.9035%
```



4c) n_modes=25 looks good and its explained variance is pretty close to 100%.

```
n_modes=25
```

```
n_modes = 25
```

```
compression_ratio=(n_modes+(n_modes*M)+(n_modes*N))/(M*N);
disp(['storage saved as 1-compression ratio: ' num2str(100-
(100*compression_ratio)) '%'])
```

```
storage saved as 1-compression ratio: 90.0582%
```

5

5a)

```
A=[ 1.97, 3.59, -0.177, 0.726;
-4.06, 0.879, -0.513, 3.4;
0.411, 5.16, -0.308, 0.756;
3.34, -2.41, -0.0974, 3.33];
[M, N]=size(A);

disp('From SVD:')
```

```
From SVD:
```

```
[U,S,V]=svd(A);
svd_norm=S(1,1)
```

```
svd_norm = 6.8390
```

```
svd_cond=S(1,1)/S(M,M)
```

```
svd_cond = 5.1683e+04
```

```

disp('From Matlab:')
```

```
From Matlab:
```

```
norm(A)
```

```
ans = 6.8390
```

```
cond(A)
```

```
ans = 5.1683e+04
```

Norm and condition from SVD are in fact equal to Matlab's norm() and cond() functions.

5b)

```
[V,D] = eig(A.'*A)
```

```
V = 4×4
   -0.0337    0.1127    0.9788   -0.1676
    0.0477   -0.0626    0.1768    0.9811
    0.9935   -0.0900    0.0342   -0.0602
```

```
    0.0974    0.9876   -0.0974    0.0759
D = 4×4
    0.0000         0         0         0
         0   23.7682         0         0
         0         0   31.3893         0
         0         0         0   46.7723
```

max eigenvalue is 4th one

```
x_for_normA=V(:,4)
```

```
x_for_normA = 4×1
   -0.1676
    0.9811
   -0.0602
    0.0759
```

5c)

```
b =[ 0.476; 0.268; 0.741; -0.391];
delta_b=[-0.744; -0.153; 0.626; 0.176];

x=inv(A)*b;
x_error=inv(A)*(b + delta_b)
```

```
x_error = 4×1
10^3 ×
   -0.2543
    0.3608
    7.5063
    0.7357
```

```
disp('relative solution error:')
```

```
relative solution error:
```

```
disp(norm(x_error - x)/norm(x))
```

```
   6.6595e+03
```

```
disp('relative error of b:')
```

```
relative error of b:
```

```
disp(norm(delta_b)/norm(b))
```

```
       0.9997
```

5d)

```
A1=[ 2.98, 0.383, 2.11, -0.942;
0.938, 2.14, 3.81, -1.98;
```

```
-0.649, 0.775, 1.16, -0.416;
1.49, 0.118, 3.61, 0.483];
[M, N]=size(A1);

[U,sigma,V]=svd(A1)
```

```
U = 4×4
   -0.4887    0.5889    0.5416   -0.3479
   -0.6816   -0.5802    0.2520    0.3678
   -0.1495   -0.4523   -0.1939   -0.8576
   -0.5236    0.3347   -0.7782    0.0907
sigma = 4×4
    6.7873         0         0         0
         0    2.5970         0         0
         0         0    1.8682         0
         0         0         0    0.0000
V = 4×4
   -0.4094    0.7712    0.4372   -0.2155
   -0.2687   -0.5110    0.2701   -0.7706
   -0.8386   -0.1094   -0.4986    0.1902
    0.2386    0.3634   -0.6981   -0.5689
```

so Vmax is the 1st and Vmin is the 4th

```
b1=V(:,1)
```

```
b1 = 4×1
   -0.4094
   -0.2687
   -0.8386
    0.2386
```

```
delta_b1=V(:,4)*0.01
```

```
delta_b1 = 4×1
   -0.0022
   -0.0077
    0.0019
   -0.0057
```

```
disp(norm(delta_b1))
```

```
    0.0100
```

```
x1=inv(A1)*b1;
delta_x1=inv(A1)*delta_b1;
disp('max relative solution error:')
```

```
max relative solution error:
```

```
disp(norm(delta_x1)/norm(x1))
```

```
       0.0054
```

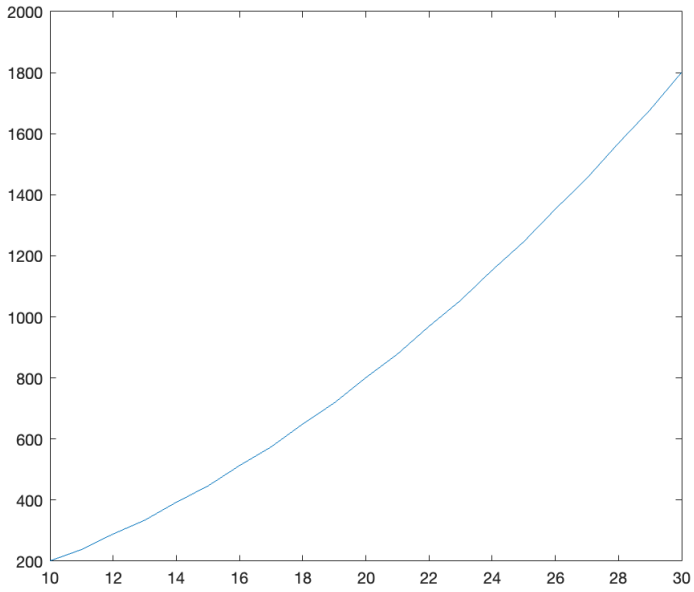
6

```
c=10:1:30;
n=1;
for N=10:1:30
    dx=(1/N);
    gamma=2;

    A=zeros([N,N]);
    for i=1:1:N
       A(i, i)=(-2/(dx^2))-gamma;
       if i==1
           A(i, N)=1/(dx^2);
       else
           A(i, i-1)=1/(dx^2);
       end
       if i==N
           A(i, 1)=1/(dx^2);
       else
           A(i, i+1)=1/(dx^2);
       end
    end
    c(n)=norm(inv(A))*norm(A);
    disp(['condition number for N=' num2str(N) ': ' num2str(c(n))])
    n=n+1;
end
```

```
 condition number for N=10: 201
 condition number for N=11: 238.0986
 condition number for N=12: 289
 condition number for N=13: 334.0892
 condition number for N=14: 393
 condition number for N=15: 446.0832
 condition number for N=16: 513
 condition number for N=17: 574.0792
 condition number for N=18: 649
 condition number for N=19: 718.0764
 condition number for N=20: 801
 condition number for N=21: 878.0744
 condition number for N=22: 969
 condition number for N=23: 1054.0729
 condition number for N=24: 1153
 condition number for N=25: 1246.0717
 condition number for N=26: 1353
 condition number for N=27: 1454.0708
 condition number for N=28: 1569
 condition number for N=29: 1678.07
 condition number for N=30: 1801
```

```
plot(10:1:30, c)
```



But from OH, TF said to insert gamma within the parentheses:

```
c=10:1:30;
n=1;
for N=10:1:30
    dx=(1/N);
    gamma=2;

    A=zeros([N,N]);
    for i=1:1:N
       A(i, i)=((-2-gamma)/(dx^2));
       if i==1
           A(i, N)=1/(dx^2);
       else
           A(i, i-1)=1/(dx^2);
       end
       if i==N
           A(i, 1)=1/(dx^2);
       else
           A(i, i+1)=1/(dx^2);
       end
    end
    c(n)=norm(inv(A))*norm(A);
    disp(['condition number for N=' num2str(N) ': ' num2str(c(n))])
    n=n+1;
end
```

```
condition number for N=10: 3
```

```
condition number for N=11: 2.9595
condition number for N=12: 3
condition number for N=13: 2.9709
condition number for N=14: 3
condition number for N=15: 2.9781
condition number for N=16: 3
condition number for N=17: 2.983
condition number for N=18: 3
condition number for N=19: 2.9864
condition number for N=20: 3
condition number for N=21: 2.9888
condition number for N=22: 3
condition number for N=23: 2.9907
condition number for N=24: 3
condition number for N=25: 2.9921
condition number for N=26: 3
condition number for N=27: 2.9932
condition number for N=28: 3
condition number for N=29: 2.9941
condition number for N=30: 3
```

```
plot(10:1:30, c)
```