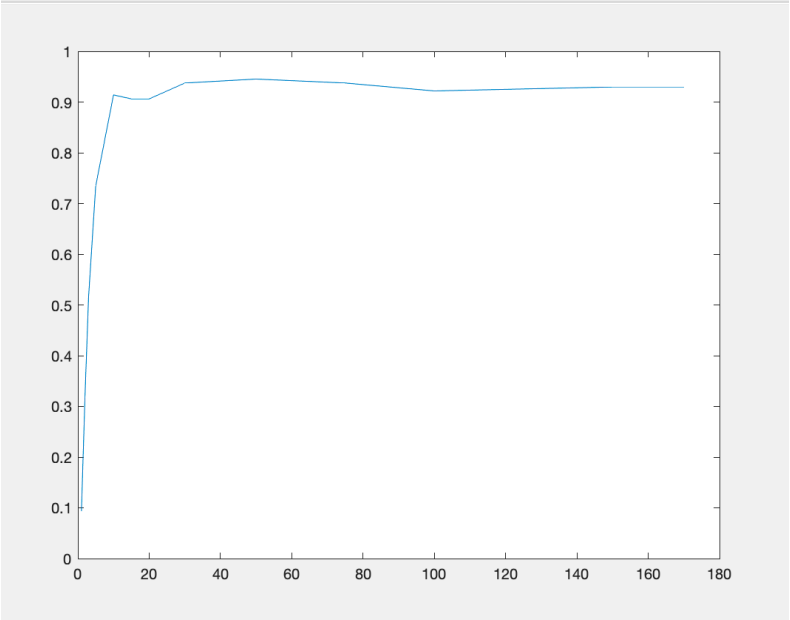
Q4)

ORL Dataset-



We see that using svd, we get a very good accuracy of face detection i.e. greater then 0.9.

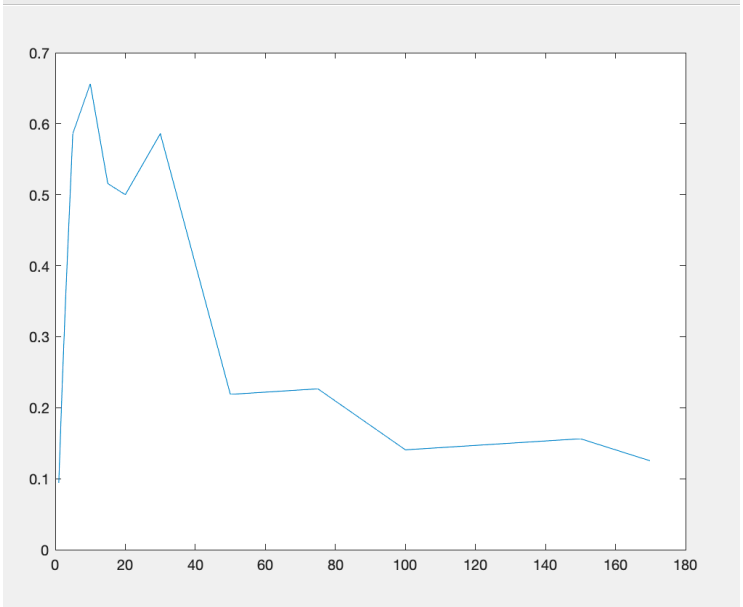
But we also see that this accuracy saturates after a certain value of number of vectors from the svd.

This is as the later sigma values become too small to actually affect the face recognition accuracy too much.

Here, in the eigenvalue method, we see that for a certain value of k, we reach maximum accuracy, after which we see steep declines in our accuracy.

This is as, we need a certain number of maximum eigenvalue eigenvectors, in order to accurately predict our data.

As we take more and more eigenvectors beyond that, we see that we are capturing the very fine details in the images. Hence our algorithm considers the little details in every image too much, hence finding considerable differences between images even for the same person's image. Thus our performance and accuracy sees a decline beyond a certain optimum value of k (around 15).



Code-

```
% INITIALIZATION

image_size = 112*92;
train_image_array = zeros(image_size,32,6);
test_image_array = zeros(image_size,32,4);
not_in_gallery_image_array = zeros(image_size,8,4);
K = [1 2 3 5 10 15 20 30 50 75 100 150 170];
final_rr = zeros([length(K),1]);

%FILE READING
directory_name = dir('ORL/');
is_directory = [directory_name.isdir]  & ~strcmp({directory_name.name},'.') &
~strcmp({directory_name.name},'..');
person_directory = directory_name(is_directory);

for i= 1:length(person_directory)
    image_files = dir(['ORL/' person_directory(i).name '/*.pgm']);
    for j=1:length(image_files)
        image = double(imread(['ORL/' person_directory(i).name '/' image_files(j).name]));
        if (i > 32) && (j<=4)
            not_in_gallery_image_array(:,i-32,j) = image(:);
        elseif (i <= 32) && (j>6)
            test_image_array(:,i,j-6) = image(:);
        elseif (i <= 32) && (j<=6)
            train_image_array(:,i,j) = image(:);
        end
    end
end

train_image_array = reshape(train_image_array,image_size,[]);
test_image_array = reshape(test_image_array,image_size,[]);
not_in_gallery_image_array = reshape(not_in_gallery_image_array,image_size,[]);
%imshow(reshape(train_image_array(:,1),112,92),[])

%MEAN
mean_images = mean(train_image_array(:,:),'all');
train_image_array= train_image_array-mean_images;

%SVD
[U,S,V] = svd(train_image_array);


%EIGENVALUES
%[W,D] = eigs(transpose(train_image_array)*train_image_array,192);
%U = train_image_array*W;
%U = U./sum(U);

TESTING
for i= 1:length(K)
   U_k = U(:,1:K(i));
   rr =0;
   projected_train_image_array = transpose(U_k)*train_image_array;
   for j= 1:32
      for k = 1:4
         test_image = test_image_array(:,32*(k-1)+j)-mean_images;
         projected_test_vector = transpose(U_k)*test_image(:);
         distance = vecnorm(projected_train_image_array-projected_test_vector,2,1);
         [M,I] = min(distance);
```

```
          if mod(l-1,32)+1 == j
             rr = rr+1;
          end
       end
    end
   final_rr(i) = rr/(32*4);
end

plot(K,final_rr)
```

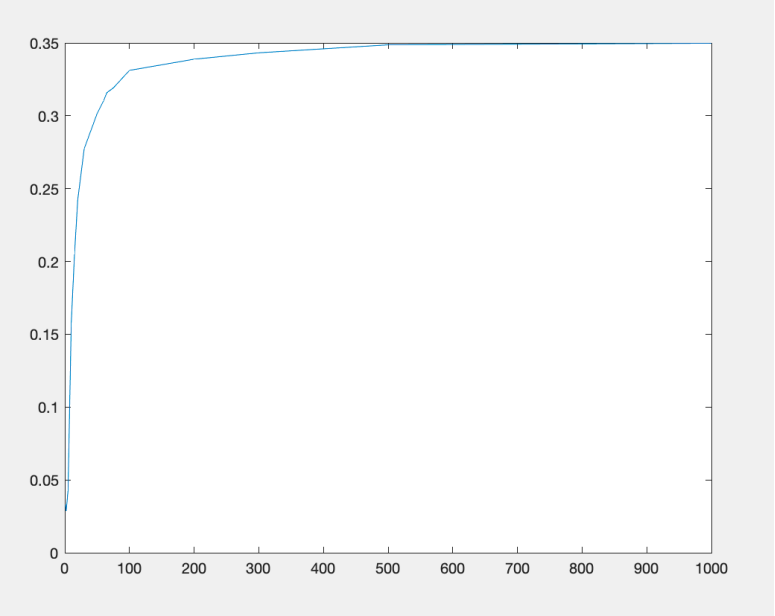
Yale Dataset

All eigenvalues included-

We see the maximum accuracy of only about 0.35.

This is as the eigenvalues of the top 3 vectors are very large. Hence they overshadow the rest of the eigenvalues, which actually contains a lot of details about the faces of the people.

Thus we see that our accuracy is not very good in this case.



With top 3 eigenvalues excluded-

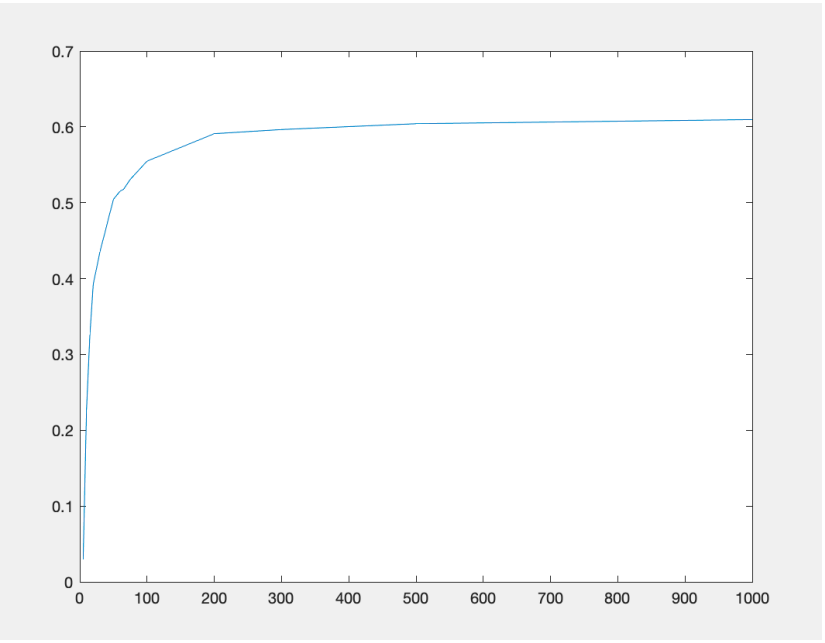
Let's exclude only our top 3 eigenvalues. They are around 10^10, as the intensities of the images are very high, (possibly images taken with the flash of the camera on).

The eigenvalues after that are considerably smaller.

Hence the top 3 eigenvalues, which mainly consist of high intensity profiles, overshadow all the rest.

Hence, by considering the top 3, we are overshadowing all the intricate details of all our remaining eigenvectors.

Thus by neglecting the top 3 eigenvectors, we get a considerably better accuracy of recognition.



Thus we see we reach a max accuracy of slightly greater than 0.6.

Note: in my code for this question, the code for the ORL dataset is at the beginning. All the code for the Yale dataset is below that, and is commented out. Similarly, uncomment only those sections that are required for that particular task from the matlab file.

Code-

```
% INITIALIZATION

image_size = 192*168;
train_image_array = zeros(image_size,38,40);
test_image_array = zeros(image_size,38,24);
K = [5 10 15 20 30 50 60 65 75 100 200 300 500 1000];
final_rr = zeros([length(K),1]);

%FILE READING
directory_name = dir('CroppedYale/');
is_directory = [directory_name.isdir]  & ~strcmp({directory_name.name},'.') &
~strcmp({directory_name.name},'..');
person_directory = directory_name(is_directory);

for i= 1:length(person_directory)
    image_files = dir(['CroppedYale/' person_directory(i).name '/*.pgm']);
    for j=1:length(image_files)
        image = double(imread(['CroppedYale/' person_directory(i).name '/' image_files(j).name]));
        if (j>40)
            test_image_array(:,i,j-40) = image(:);
        else
            train_image_array(:,i,j) = image(:);
        end
    end
end

train_image_array = reshape(train_image_array,image_size,[]);
test_image_array = reshape(test_image_array,image_size,[]);

%MEAN
mean_images = mean(train_image_array(:,:),'all');
train_image_array= train_image_array-mean_images;

%SVD
[U,S,V] = svd(train_image_array,"econ");


%EIGENVALUES
%[W,D] = eigs(transpose(train_image_array)*train_image_array,192);
%U = train_image_array*W;
%U = U./sum(U);


%TESTING
for i= 1:length(K)
    U_k = U(:,4:K(i));
```

```
    rr =0;
    projected_train_image_array = transpose(U_k)*train_image_array;
    for j= 1:38
        for k = 1:24
            test_image = test_image_array(:,38*(k-1)+j)-mean_images;
            projected_test_vector = transpose(U_k)*test_image(:);
            distance = vecnorm(projected_train_image_array-projected_test_vector,2,1);
            [M,I] = min(distance);
            if mod(I-1,38)+1 == j
               rr = rr+1;
            end
        end
    end
   final_rr(i) = rr/(38*24);
end


plot(K,final_rr)
```