1. (a)

I did this homework with Luning Zhao.

We work on our homework seperately, but we discuss when meeting problems.

Homework is fine.

(b) I certify that all solutions are entirely in my words. and that I have not looked at another student's solutions. I have credited all external sources in this write up.

2. (a) because $(X+\mathcal{E}_X)w = y+\mathcal{E}_y$, so $y+\mathcal{E}_y$ is a linear combination of columns in $X+\mathcal{E}_X$.

so $\text{rank}([X+\mathcal{E}_X,\ y+\mathcal{E}_y]) = \text{rank}(X+\mathcal{E}_X) = d$

(b) $[\mathcal{E}_X, \mathcal{E}_y] = [X+\mathcal{E}_X,\ y+\mathcal{E}_y] - [X, y]$

$$= \begin{bmatrix} U_{xx} & u_{xy} \\ u_{xy}^T & u_{yy} \end{bmatrix} \left(\begin{bmatrix} \Sigma_d & \\ & 0 \end{bmatrix} - \begin{bmatrix} \Sigma_d & \\ & \sigma_{d+1} \end{bmatrix} \right) \begin{bmatrix} V_{xx} & v_{xy} \\ v_{xy}^T & v_{yy} \end{bmatrix}^T$$

$$= - \begin{bmatrix} U_{xx} & u_{xy} \\ u_{xy}^T & u_{yy} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & \sigma_{d+1} \end{bmatrix} \begin{bmatrix} V_{xx} & v_{xy} \\ v_{xy}^T & v_{yy} \end{bmatrix}^T$$

$$= - \begin{bmatrix} 0 & u_{xy}\sigma_{d+1} \\ 0 & u_{yy}\sigma_{d+1} \end{bmatrix} \begin{bmatrix} V_{xx} & v_{xy} \\ v_{xy}^T & v_{yy} \end{bmatrix}^T$$

$$= -\sigma_{d+1} \begin{bmatrix} u_{xy} \\ u_{yy} \end{bmatrix} \begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix}^T$$

(c) $[\mathcal{E}_X, \mathcal{E}_y] = -\sigma_{d+1} \begin{bmatrix} u_{xy} \\ u_{yy} \end{bmatrix} \begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix}^T$

We find that $-[X, y] \begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix} \begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix}^T$

$$= - \underbrace{\begin{bmatrix} U_{xx} & u_{xy} \\ u_{xy}^T & u_{yy} \end{bmatrix}}_{n \times n} \underbrace{\begin{bmatrix} \Sigma_d & 0 \\ 0 & \sigma_{d+1} \end{bmatrix}}_{n \times d+1} \underbrace{\begin{bmatrix} V_{xx} & v_{xy} \\ v_{xy}^T & v_{yy} \end{bmatrix}^T}_{d+1 \times n} \underbrace{\begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix}}_{n \times 1} \underbrace{\begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix}^T}_{1 \times n}$$

$$\underbrace{\begin{bmatrix} \Sigma_d & 0 \\ 0 & \sigma_{d+1} \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}}_{\begin{bmatrix} 0 \\ \vdots \\ \sigma_{d+1} \end{bmatrix}}$$

$$\underbrace{\begin{bmatrix} U_{xx} & u_{xy} \\ u_{xy}^T & u_{yy} \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ \sigma_{d+1} \end{bmatrix}}_{\begin{bmatrix} u_{xy}\sigma_{d+1} \\ u_{yy}\sigma_{d+1} \end{bmatrix}}$$

$$= -\sigma_{d+1} \begin{bmatrix} u_{xy} \\ u_{yy} \end{bmatrix} \begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix}^T = [\mathcal{E}_X, \mathcal{E}_y]$$

So $[x+\varepsilon_x, y+\varepsilon_y] = [x, y] + [\varepsilon_x, \varepsilon_y]$

$$= [x, y] - [x, y]\begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix}\begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix}^T$$

If we right multiply by $\begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix}$ $\Rightarrow$ $[x, y]\begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix} - [x, y]\begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix}\underbrace{\begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix}^T\begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix}}_{1}$

$$= [x, y]\begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix} - [x, y]\begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix} = 0$$

So $[x+\varepsilon_x, y+\varepsilon_y]\begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix} = 0$

$\Rightarrow$ $\begin{bmatrix} w \\ -1 \end{bmatrix} = \begin{bmatrix} v_{xy} \\ v_{yy} \end{bmatrix}$ $\Rightarrow$ $w = -\dfrac{v_{xy}}{v_{yy}}$

(d) $\begin{bmatrix} w \\ -1 \end{bmatrix}$ is a right-singular vector of $[x, y]$.

so $[x, y]^T [x, y]\, \vec{V}_{d+1} = V \Sigma \cancel{U^T U} \Sigma V^T \vec{V}_{d+1}$

$$= V \begin{bmatrix} \Sigma_d^2 & \\ & \sigma_{d+1}^2 \end{bmatrix} V^T \vec{V}_{d+1}$$

$$= \sigma_{d+1}^2 \vec{V}_{d+1}$$

We can see $\vec{V}_{d+1}$ is a eigenvector of $[x, y]^T [x, y]$, with the eigenvalue of σ_{d+1}^2

so $\begin{bmatrix} X^TX & X^Ty \\ y^TX & y^Ty \end{bmatrix}\begin{bmatrix} w \\ -1 \end{bmatrix} = \sigma_{d+1}^2 \begin{bmatrix} w \\ -1 \end{bmatrix}$

$\Rightarrow$ $X^TX w - X^Ty = \sigma_{d+1}^2 w$.

$(X^TX - \sigma_{d+1}^2 I) w = X^Ty$

2 (e)

the estimated value for γ

```
Coefficients:
[[ 202.31845431  162.41956802  149.07075034]
 [ -27.66555164  -17.88905339  -12.92356688]
 [  -5.15203925   -4.51375871   -4.24262639]
 [  -1.08629293    0.42947012    1.15475569]
 [  -3.14053107   -3.70269907   -3.74382934]
 [  23.67671768   23.15698002   21.94638397]
 [  -3.82167171    0.57606634    1.81637483]
 [   4.7346737     1.4677692    -1.12253649]
 [  -9.72739616   -5.75691108   -4.8395598 ]]
```

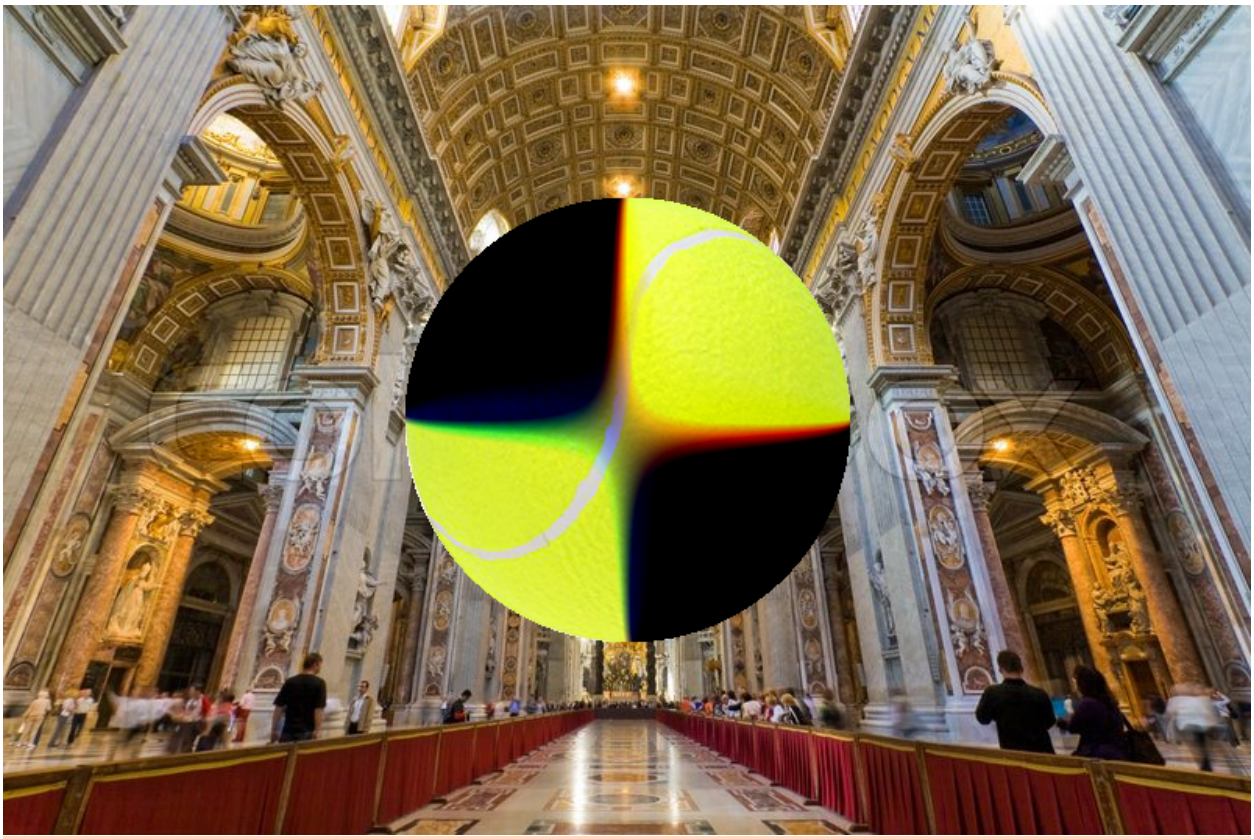
visualization of approximation

code:

```
# Fill in this function to compute the basis functions
# This function is used in renderSphere()
def computeBasis(ns):
    # Returns the first 9 spherical harmonic basis functions

    ################################################
    # TODO: Compute the first 9 basis functions
    ################################################
    B = np.ones((len(ns),9)) # This line is here just to fill space

    x = ns[:,0]
    y = ns[:,1]
    z = ns[:,2]

    B.T[0] = 1
    B.T[1] = y
    B.T[2] = x
    B.T[3] = z
    B.T[4] = x*y
    B.T[5] = y*z
    B.T[6] = 3*z*z - 1
    B.T[7] = x*z
    B.T[8] = x*x - y*y

if __name__ == '__main__':

    data,tennis,target = loadImages()
    ns, vs = extractNormals(data)
    B = computeBasis(ns)

    # reduce the number of samples because computing the SVD on
    # the entire data set takes too long
    Bp = B[::50]
    vsp = vs[::50]

    ################################################
    # TODO: Solve for the coefficients using least squares
    # or total least squares here
    ################################################
    #coeff = np.zeros((9,3))
    #coeff[0,:] = 255

    # use OLS to calculate coeff
    coeff = np.linalg.inv((Bp.T @ Bp)) @ Bp.T @ vsp

    img = relightSphere(tennis,coeff)

    output = compositeImages(img,target)

    print('Coefficients:\n'+str(coeff))

    plt.figure(1)
    plt.imshow(output)
    plt.show()

    imsave('output.png',output)
```

2 (f)

```
Coefficients:
[[  2.13318421e+02   1.70780299e+02   1.57126297e+02]
 [ -3.23046362e+01  -2.02975310e+01  -1.45516114e+01]
 [ -4.31689131e+00  -3.80778081e+00  -4.83616306e+00]
 [ -4.89811386e+00  -3.37684058e+00  -1.14207091e+00]
 [ -7.05901066e+03  -7.39934207e+03  -4.26448732e+03]
 [ -3.05378224e+02  -1.56329401e+02   3.50285345e+02]
 [ -9.76079364e+00  -5.33182216e+00  -1.55699782e+00]
 [  7.30792588e+02   3.52130316e+02  -6.11683200e+02]
 [ -9.08887079e+00  -3.84309477e+00  -4.16456437e+00]]
```



```
## use TLS to calculate coeff
A = np.concatenate((Bp.T,vsp.T)).T
u,s,vt = np.linalg.svd(A, full_matrices=False)
vt = vt.T
vtx = vt.T[-3:].T[:-3]
vty = vt.T[-3:].T[-3:]
coeff = vtx @ np.linalg.inv(vty)
```

2 (g) I scale f(n) by 1/384.

```
Coefficients:
[[ 209.38212459  169.03666402  155.36677288]
 [ -30.26805402  -20.30443706  -15.20472049]
 [  -5.753416     -5.07881542   -4.78144904]
 [  -1.05630713    0.46377951    1.19195587]
 [  -7.90569522   -8.20316831   -8.05137623]
 [  54.96251667   52.62398401   50.09265545]
 [  -3.8491927     0.55663535    1.80236903]
 [   7.32655583    3.83064183    1.07500107]
 [ -10.90665749   -6.8522162    -5.87526417]]
```

3. (a) $XX^T = U\Sigma V^T V\Sigma U^T = U\Sigma\Sigma U^T$

$X^TX = V\Sigma U^T U\Sigma V^T = V\Sigma\Sigma V^T$

(b) $\psi_{PCA}(\vec{x}) = (\vec{v}_1^T\vec{x}, \cdots, \vec{v}_k^T\vec{x})^T$

$$= \begin{bmatrix} \vec{v}_1^T \\ \vdots \\ \vec{v}_k^T \end{bmatrix} \cdot \vec{x}$$

$$= V_k^T\vec{x}$$

$\Rightarrow \psi_{PCA}(\vec{x}_i)^T \psi_{PCA}(\vec{x}_j) = \vec{x}_i^T V_k V_k^T \vec{x}_j$

$$V_k V_k^T = [\vec{v}_1, \cdots, \vec{v}_k] \begin{bmatrix} \vec{v}_1^T \\ \vdots \\ \vec{v}_k^T \end{bmatrix} = \vec{v}_1\vec{v}_1^T + \cdots + \vec{v}_k\vec{v}_k^T$$

$$V I^k V^T = [\vec{v}_1, \cdots, \vec{v}_d] \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 0 \end{bmatrix} \begin{bmatrix} \vec{v}_1 \\ \vdots \\ \vec{v}_d \end{bmatrix} = \vec{v}_1\vec{v}_1^T + \cdots + \vec{v}_k\vec{v}_k^T$$

so we can prove that $V_k V_k^T = V I^k V^T$

(c) $\sum_{i=1}^{n}\sum_{j=1}^{n} (\vec{x}_i^T\vec{x}_j)^2 = \sum_i\sum_j (XX^T)_{ij}^2 = \|XX^T\|^2 = \left\| U \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_d^2 \end{bmatrix} U^T \right\|^2 = \sum_{i=1}^{d} \sigma_i^2$

$\sum_i\sum_j |\vec{x}_i^T\vec{x}_j - \psi_{PCA}(\vec{x}_i)^T \psi_{PCA}(\vec{x}_j)|^2$

$= \sum_i\sum_j |(XX^T)_{ij} - \vec{x}_i^T V I^k V^T \vec{x}_j|^2$

$= \sum_i\sum_j |(XX^T)_{ij} - X V I^k V^T X^T|^2$

$= \|X V (I - I^k) V^T X^T\|_F^2$

$= \|U\Sigma V^T V (I - I^k) V^T V\Sigma U^T\|_F^2$

$= \|U\Sigma (I - I^k)\Sigma U^T\|_F^2$

$= \left\| U \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_d \end{bmatrix} \begin{bmatrix} 0 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_d \end{bmatrix} U^T \right\|_F^2 = \sum_{i=k+1}^{d} \sigma_i^2.$

$\Rightarrow \frac{1}{\sum\sum (\vec{x}_i^T\vec{x}_j)^2} \sum\sum |(\vec{x}_i^T\vec{x}_j - (\psi_{PCA}(\vec{x}_i)^T \psi_{PCA}(\vec{x}_j)|^2 \le \varepsilon$

(d) $|\psi(\vec{x}_i)^T \psi(\vec{x}_j) - \vec{x}_i^T \vec{x}_j|$

$= \frac{1}{2} \left| (\psi(\vec{x}_i) - \psi(\vec{x}_j))^T (\psi(\vec{x}_i) - \psi(\vec{x}_j)) - \psi(\vec{x}_i)^T \psi(\vec{x}_i) - \psi(\vec{x}_j)^T \psi(\vec{x}_j) \right.$

$\left. - (\vec{x}_i - \vec{x}_j)^T (\vec{x}_i - \vec{x}_j) + \vec{x}_i^T \vec{x}_i + \vec{x}_j^T \vec{x}_j \right|$

$= \frac{1}{2} \left| \|\psi(\vec{x}_i) - \psi(\vec{x}_j)\|^2 - \|\psi(\vec{x}_i)\|^2 - \|\psi(\vec{x}_j)\|^2 - \|\vec{x}_i - \vec{x}_j\|^2 + \|\vec{x}_i\|^2 + \|\vec{x}_j\|^2 \right|$

$\leq \frac{\varepsilon}{2} \left| \|\vec{x}_i - \vec{x}_j\|^2 - \|\vec{x}_i\|^2 - \|\vec{x}_j\|^2 \right|$

$= \frac{\varepsilon}{2} \|(\vec{x}_i - \vec{x}_j)^T (\vec{x}_i - \vec{x}_j) - \vec{x}_i^T \vec{x}_i - \vec{x}_j^T \vec{x}_j\|$

$= \varepsilon \|\vec{x}_i^T \vec{x}_j\|^2$

$\leq \varepsilon \cdot \|\vec{x}_i\| \|\vec{x}_j\|$

Let $c = \max_{ij} \|\vec{x}_i\|^2 \|\vec{x}_j\|^2$, then $|\psi(\vec{x}_i)^T \psi(\vec{x}_i) - (\vec{x}_i^T \vec{x}_j)| \leq \varepsilon$

(e) $\|\psi_J(\vec{u})\|^2 = \frac{1}{k} \|J\vec{u}\|^2$ $\quad J = \begin{bmatrix} \vec{J}_1^T \\ \vdots \\ \vec{J}_k^T \end{bmatrix}$ $\quad \|\vec{u}\|^2 = u_1^2 + \cdots + u_d^2$

$= \frac{1}{k} \left\| \begin{bmatrix} \vec{J}_1^T \vec{u} \\ \vdots \\ \vec{J}_k^T \vec{u} \end{bmatrix} \right\|^2$

$= \frac{1}{k} (\|\vec{J}_1^T \vec{u}\|^2 + \cdots + \|\vec{J}_k^T \vec{u}\|^2)$

$= \frac{1}{k} \sum_{i=1}^{k} \frac{\|\vec{J}_i^T \vec{u}\|^2}{\|\vec{u}\|^2}$

$= \frac{1}{k} \sum_{i=1}^{k} \left(\frac{|\vec{J}_i^T \vec{u}|}{\sqrt{\|\vec{u}\|^2}} \right)^2$

Since $\vec{J}_i^T \vec{u} = J_{i1} u_1 + \cdots + J_{id} u_d \sim N(0, \sqrt{u_1^2 + \cdots + u_d^2})$

$\|\psi_J(\vec{u})\|^2 = \frac{1}{k} \sum_{i}^{k} Z_i^2$ where $Z_i \sim N(0,1)$

(f) define event $A_{ij} = \left\{ \frac{\|\psi_3(\vec{x}_i) - \psi_3(\vec{x}_j)\|^2}{\|\vec{x}_i - \vec{x}_j\|^2} \in (1-\varepsilon, 1+\varepsilon) \right\}$

$$P[\cap_{ij} A_{ij}] \geq 1 - \sum_{ij} P[A_{ij}^c]$$

$$= 1 - \sum_{ij} P\left\{ \frac{\|\psi_3(\vec{x}_i - \vec{x}_j)\|^2}{\|\vec{x}_i - \vec{x}_j\|^2} \notin (1-\varepsilon, 1+\varepsilon) \right\}$$

$$= 1 - \sum_{ij} P\left\{ \left| \frac{1}{k} \sum z_i^2 \right| \notin (1-\varepsilon, 1+\varepsilon) \right\}$$

$$\geq 1 - \sum_{ij} 2e^{-\frac{k\varepsilon^2}{8}}$$

$$= 1 - 2N^2 e^{-\frac{k\varepsilon^2}{8}} \geq 1 - \delta.$$

$$2N^2 e^{-\frac{k\varepsilon^2}{8}} \leq \delta.$$

$$k \geq \frac{16}{\varepsilon^2} \log \frac{N}{\delta}$$

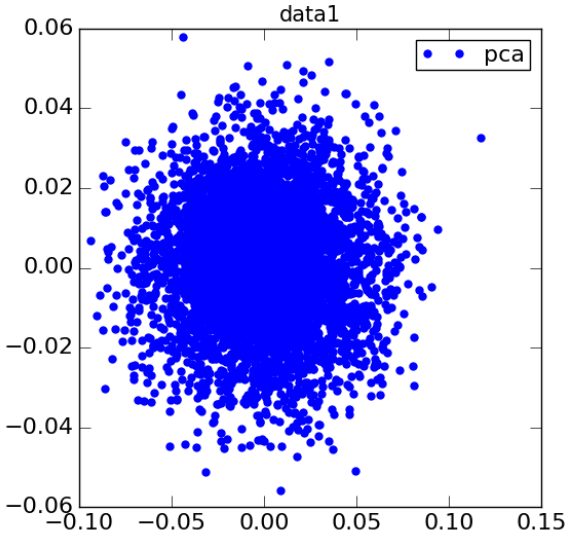
(g) $P\left[\frac{\|\psi_3(\vec{u}) - \psi_3(\vec{v})\|^2}{\|\vec{u} - \vec{v}\|^2} \in (1-\varepsilon, 1+\varepsilon) \right] \geq 1 - \delta$ if $k \geq \frac{16}{\varepsilon^2} \log \frac{N}{\delta}$

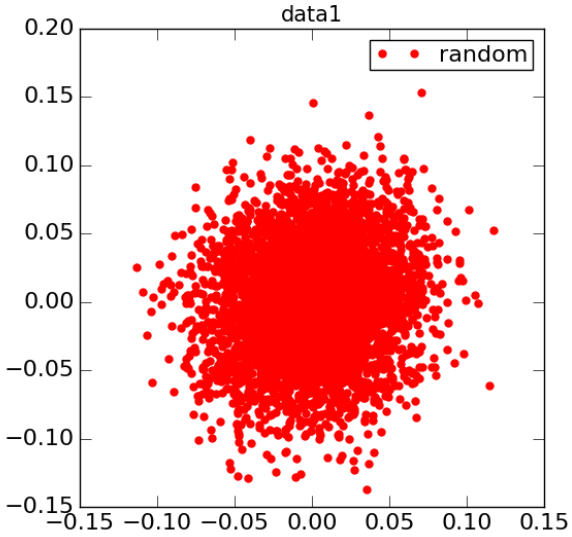
$\Rightarrow$ $P[\|\psi_3(\vec{u}) - \psi_3(\vec{v})\|^2 \geq (1-\varepsilon)\|\vec{u} - \vec{v}\|^2] \geq 1 - \delta$ ~

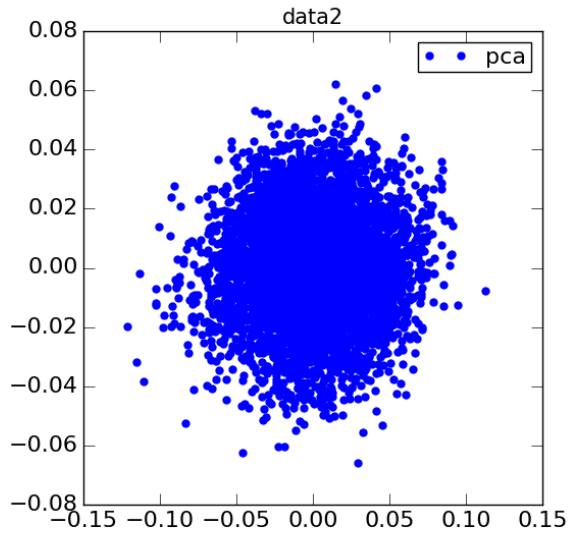
$\Rightarrow$ $P[\|\psi_3(\vec{u}) - \psi_3(\vec{v})\|^2 \geq (1-\varepsilon)\gamma] \geq 1 - \delta$ ~.

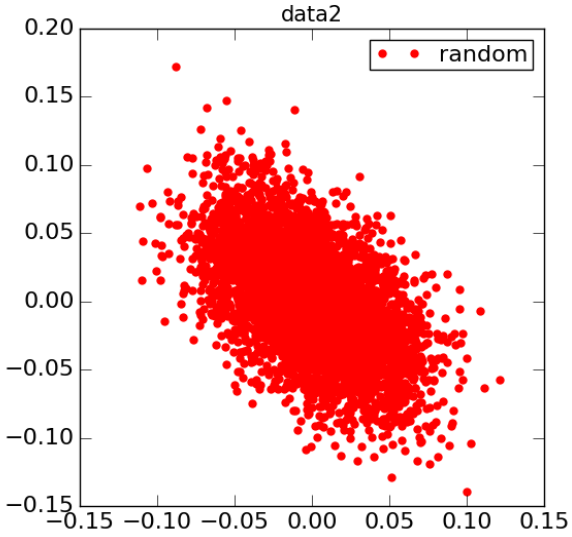
So with high probability, $\min \|\psi_3(\vec{u}) - \psi_3(\vec{v})\|^2 \geq (1-\varepsilon)\gamma$ if $k \geq \frac{c}{\varepsilon^2} \log(m+n)$

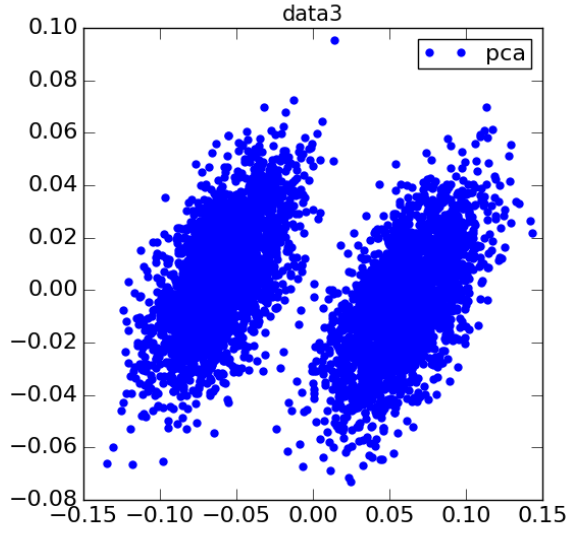
3 (h)

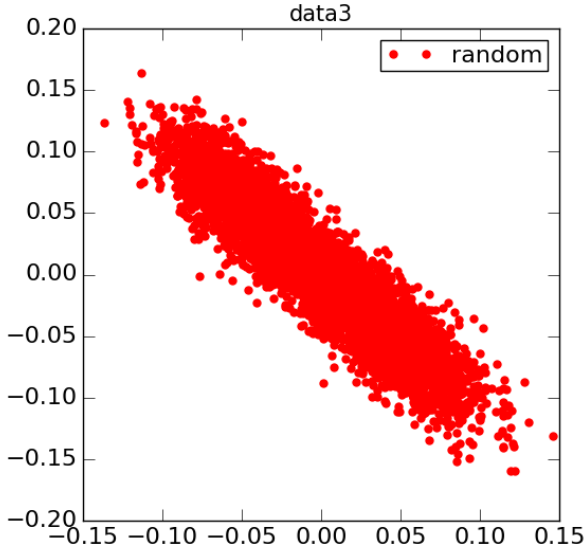

data1
pca
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.10
−0.05
0.00
0.05
0.10
0.15


data1
random
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15


data2
pca
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15


data2
random
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15


data3
pca
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15

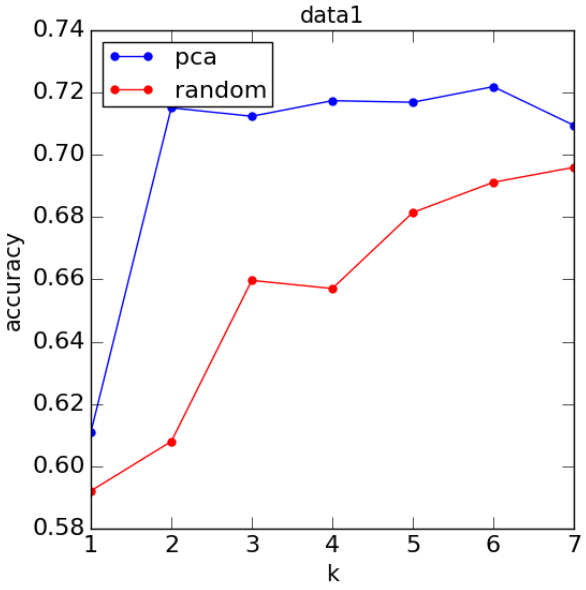

data3
random
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15

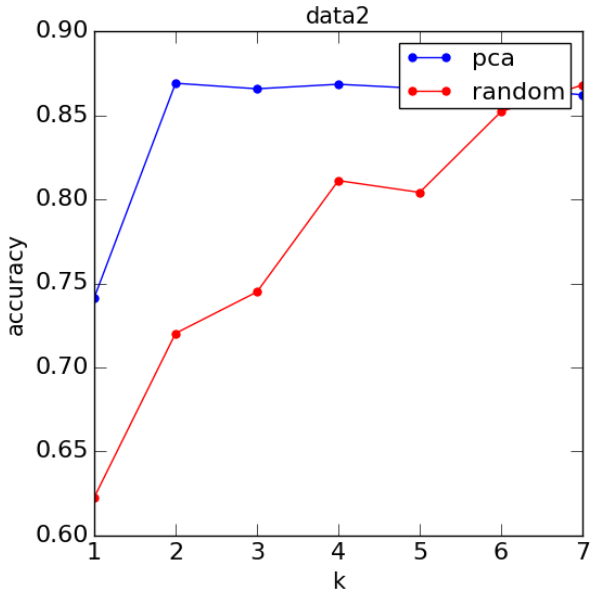
Yes, there is a difference between PCA and random projections.

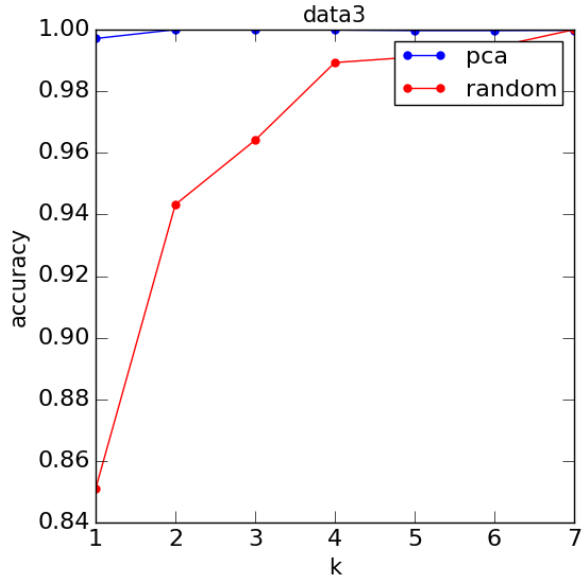
From data1 to data3, the difference between PCA and random projection is more and more obvious.

```
# Using PCA and Random Projection for:
# Visualizing the datasets
X_pca = pca_proj(X, 2)
plt.clf()
plt.plot(X_pca.T[0], X_pca.T[1], 'bo', mec='none', label='pca')
plt.legend()
plt.title('data3')
plt.savefig('pca_3.png', format='png')

X_rand = random_proj(X, 2)
plt.clf()
plt.plot(X_rand.T[0], X_rand.T[1], 'ro', mec='none', label='random')
plt.legend()
plt.title('data3')
plt.savefig('rand_3.png', format='png')
```

3 (i)



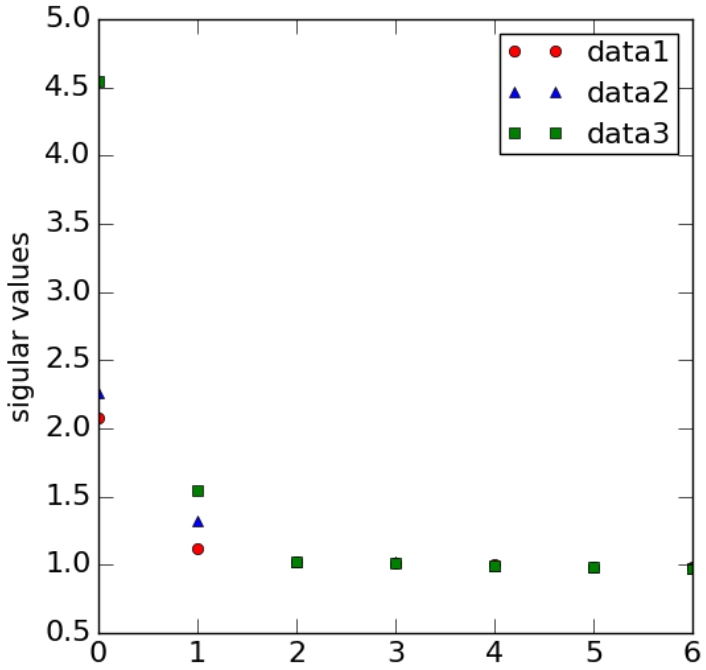




The accuracy increases with k for both pea and random projection, and they converge when k gets close to d. In data3, when data is more correlated, the accuracy is higher.

```
# Computing the accuracies over different datasets.
pca_acc = np.zeros((d, n_trials))
rand_acc = np.zeros((d, n_trials))
for k in range(d):
    for i in range(n_trials):
        pca_acc[k][i] = pca_proj_accuracy(X, y, k+1)
        rand_acc[k][i] = rand_proj_accuracy_split(X, y, k+1)
plt.clf()
# Don't forget to average the accuracy for multiple
# random projections to get a smooth curve.
plt.plot(np.arange(d)+1, np.mean(pca_acc, axis=1), 'bo-', mec='none', label='pca')
plt.plot(np.arange(d)+1, np.mean(rand_acc, axis=1), 'ro-', mec='none', label='random')
plt.legend(loc='best')
plt.title('data1')
plt.xlabel('k')
plt.ylabel('accuracy')
plt.savefig('acc_k_1.png', format='png')
```

3 (j)



The fist few singular values are much larger compared to the rest of them in data3, but not in data1, meaning data3 is dominant in the first few dimensions. That's the reason why pca/ random works better for data3.

```
# And computing the SVD of the feature matrix
data = np.load('data1.npz')
X = data['X']
u,s1,v = np.linalg.svd(X)

data = np.load('data2.npz')
X = data['X']
u,s2,v = np.linalg.svd(X)

data = np.load('data3.npz')
X = data['X']
u,s3,v = np.linalg.svd(X)

plt.clf()
plt.plot(s1, 'ro-', label='data1')
plt.plot(s2, 'b^-',label='data2')
plt.plot(s3, 'gs-', label='data3')
plt.legend()
plt.ylabel('sigular values')
plt.savefig('sv.png', format='png')
```

5. Q: Let $H = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}$, where A is a square and $A = U\Sigma V^T$ is its SVD.

Let $\Sigma = \text{diag}(\sigma_1, \cdots, \sigma_n)$, $U = [\vec{u}_1, \cdots, \vec{u}_n]$, $V = [\vec{v}_1 \cdots, \vec{v}_n]$

Prove that the eigenvalues of H are $\pm \sigma_i$, with corresponding eigenvectors $\frac{1}{\sqrt{2}} \begin{bmatrix} \vec{v}_i \\ \pm \vec{u}_i \end{bmatrix}$.

A: $H \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} \vec{v}_i \\ \vec{u}_i \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \vec{v}_i \\ \vec{u}_i \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} A^T \vec{u}_i \\ A \vec{v}_i \end{bmatrix}$

$A^T \vec{u}_i = V \Sigma U^T \vec{u}_i = [\vec{v}_1, \cdots, \vec{v}_n] \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix} \begin{bmatrix} \vec{u}_1^T \\ \vdots \\ \vec{u}_n^T \end{bmatrix} \vec{u}_i = \sigma_i \vec{v}_i$

Similarly, $A \vec{v}_i = \sigma_i \vec{u}_i$.

$\Rightarrow H \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} \vec{v}_i \\ \vec{u}_i \end{bmatrix} = \frac{\sigma_i}{\sqrt{2}} \begin{bmatrix} \vec{v}_i \\ \vec{u}_i \end{bmatrix}$, so $\frac{1}{\sqrt{2}} \begin{bmatrix} \vec{v}_i \\ \vec{u}_i \end{bmatrix}$ is an eigenvector for H, with eval of σ_i.