Ex 1: ① generate Gaussian data $X \in \mathbb{R}^{2\times 63}$ (isotropic Gaussian, meaning $E(xx^T) = I$, $Ex = 0$)

② linear transform x: $y = Ax + t$, where $A = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$, $t = \begin{bmatrix} 8 \\ 1 \end{bmatrix}$

You're welcome to use your own A matrix (but make sure it's non-singular)

③ Given merely y, we are going to recover A, t.

since $Ex = 0$, $Ey = E(Ax + t) = E(Ax) + t = A \cdot Ex + t = t$.

hence by averaging y, we get an estimate of t (the translation vector)

④ center the data: $\hat{y} \leftarrow y -$ mean(y, 2) * ones(1, size(y,2));

i.e., substract the empirical average of y from each column

⑤ compute the empirical covariance matrix: $C = \hat{y} * \hat{y}^T / \text{size}(y,2)$;

By the law of large numbers, $C \mapsto E(yy^T) = E((Ax)(Ax)^T) = A\,E(xx^T)\,A^T = AA^T$

⑥ eigen-decompose $C = U \Sigma U^T$, where U represents the rotation, the diagonal of Σ represents the scaling (along the new coordinate system determined by U)

⑦ for your interest, compare the C matrix we computed (which is an empirical estimate of AA^T) with AA^T, are they close?

⑧ plot the results as indicated in the web. Note that the eigenvalue represents the variance along the direction determined by the corresponding eigenvector. It is NOT meant to be the length of some ellipsoid inscribing the data.

Ex 4: (similar to Ex 1) ① Form the matrix M, each of its column represents a (vectorized) image

② center the data: $M_Z \leftarrow M -$ mean(M, 2) * ones(1, size(M,2));

③ Form the covariance: $C = M_Z' * M_Z / \text{size}(M,2)$;

④ compute the "largest" K eigen pairs of C, by using [Y, D] = eigs(C, K);

⑤ compute the Basis images: $B = M_Z * Y$; where Y are the eigenvectors (is B orthogonal? orthonormal?)

⑥ the coefficients for our sample images are Y';

⑦ each column of B is a basis image, reshape it into a matrix and image show it.

⑧ show the curve for each row of Y', these are going to be interpolated

Ex 2: general idea: we want a succinct representation of a sequence of similar objects.

Our model: each instance/object is a linear combination of some primitive/basis objects.

$$\boxed{\text{Object}_K = \text{Primitive}_1 * a_{1,K} + \text{Primitive}_2 * a_{2,K} + \cdots + \text{Primitive}_m * a_{m,K}}$$

therefore to describe all objects, we need only the Primitives and the coefficients A_K

note that Primitives are shared by all objects, while each object has its own unique coefficient A_K.

the frequency is controlled by a, the bigger a is, the higher the frequency is.

let $B(:,1) = \sin(ax) \Rightarrow ax = \arcsin(B(:,1)) \Rightarrow \sin(2ax) = ?$

effect of changing b: first take an educated guess, then experimentally verify.

Ex 3: almost the same as Ex 2, except that you are asked to find the basis images through Matlab's "gradient" function.

note that we only have three basis images in this exercise: I, I_x, I_y

and each coefficient vector has three entries: $1, dx, dy$, where dx, dy represents the shift along x, y direction respectively. ($dx=1, dy=1$ means shift down and right one pixel)

Ex 5: recall each image is written as a linear combination of some basis images

$$[I_1, I_2, \cdots, I_n] = [B_1, B_2, \cdots, B_m] \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1,n} \\ A_{21} & A_{22} & \cdots & A_{2,n} \\ \vdots & \vdots & & \vdots \\ A_{m,1} & A_{m,2} & \cdots & A_{m,n} \end{bmatrix}$$

for each mat file, there's an X vector

to interpolate new images, we need only interpolate the coefficients, i.e., each row of A

take the first row as an example:

$$\left.\begin{matrix} x_1 & x_2 & \cdots & x_n \\ A_{11} & A_{12} & \cdots & A_{1n} \end{matrix}\right\} \Rightarrow \text{fitting a polynomial with suitable degree}$$

then on a more densely sampled X vector $x_1 \; x_{1.5} \; x_2 \; x_{2.5} \cdots x_n$

$\tilde{A}_{11} \; \tilde{A}_{1.51} \cdots \tilde{A}_{n,1} \leftarrow$ given by your interpolated polynomial

repeat the procedure for each row of A, eventually we will get a new coefficient matrix $\tilde{A}$, which has more columns than A. Note that each column represents a different image, hence we have successfully interpolated new images. Draw them