Modeling Questions: SVD

1. Suppose we have p data points in $\mathbb{R}^n$.
   - The mean vector is a vector in what space?
   - When we discuss "covariance" what are we talking about- In other words, we're looking at how two sets of data interact- Which two sets of data?
2. There are three forms of the SVD: The regular SVD, the reduced SVD (given the rank), and the "economy sized" SVD from Matlab. Explain the difference.
3. From the SVD of a matrix A, what do the columns of matrices U and V represent? What do the singular values represent?
4. Suppose that matrix U has orthonormal columns and is $m \times k$. (Must $k \leq m$?) Suppose we have a matrix Z that is $m \times p$ holding p data points in $\mathbb{R}^m$. What expression represents the **coordinates of the projection** of the data in Z to $\mathbb{R}^m$? What expression represents the projection of the data to the $k-$dimensional subspace in $\mathbb{R}^m$?
5. Here we examine numerical error produced by solving $A\mathbf{x} = \mathbf{b}$.
   (a) First, define matrix A as a 20×20 Hilbert matrix (Matlab command: `A=hilb(20);`). The Hilbert matrix is a template of a matrix with some very interesting numerical properties, and we'll investigate them here.
   (b) Let $\mathbf{x}$ be a random vector in $\mathbb{R}^{20}$, and define $\mathbf{b} = A\mathbf{x}$.
   (c) Try solving $A\mathbf{x} = \mathbf{b}$ using the `inv` command, and see how close we get- The Hilbert matrix is actually invertible, but numerically we might see some issues:

   ```
   x1=inv(A)*b;
   norm(x-x1)
   ```

   (d) Compute the SVD of A. We're going to try to find the best pseudoinverse. Try creating this loop through the number of columns of U. For each index:
      i. Create the pseudoinverse using that many columns.
      ii. Compute the approximate solution.
      iii. Find the error between the approximate solution and the actual solution.
      iv. Repeat.

   At the end, plot the values of the error that you find, and locate the number of columns that is the best!
6. For this problem, write a Matlab script that will do the following:

(a) Create a random matrix of data, 800×15. Call this matrix X, and in this problem, we're thinking of this as 15 points in $\mathbb{R}^{800}$. Because of this viewpoint, we also note that the mean vector should be a vector in $\mathbb{R}^{800}$ as well.

(b) The script should mean subtract the data. That is, the mean vector is computed, then subtracted from each column.

MATLAB SIDE NOTE: New in Matlab! Matlab will allow you to subtract a vector from a matrix, but beware! This functionality (which does violate the rules from linear algebra) will also allow you to make errors that you weren't allowed to make before. If A is $m \times n$, and $\mathbf{u}$ is $m \times 1$, then $A - \mathbf{u}$ will assume that you want the subtraction column-wise. If $\mathbf{u}$ is $1 \times n$, then $A - \mathbf{u}$ will assume you want the subtraction row-wise. If $\mathbf{u}$ is neither size, then an error will be produced.

What does this mean for us? We don't have to use the `repmat` command any more!

(c) Once the data has been mean-subtracted, we want to find the best two dimensional basis for the data. Do this by using the SVD, and show the two vectors by plotting them.

For example, if the first two columns of a matrix B are large, we can visualize them by plotting them:

```
plot(1:size(B,1),B(:,1),'r-',1:size(B,1),B(:,2),'k-')
```

(There won't be any patterns- Our data was noise!)

(d) Plot the two dimensional representation of the data. These are the projection coefficients- Think about how to get them using matrix algebra.

(e) Using our two basis vectors, reconstruct the data back to $\mathbb{R}^{800}$ and see which of the 15 columns was closest to being reconstructed.

If matrix B is approximating matrix A, how would we do this?

```
errors=sum((A-B).^2);
find(errors==min(errors))
```

(Because the data was random, everyone's answer should be different)

(f) Verify that the first two columns of the matrix V can be computed by using $A^T\mathbf{u}_i = \mathbf{v}_i$ (then normalize). Similarly, verify that $A\mathbf{v}_i = \mathbf{u}_i$ (then normalize).

If the matrix was $700,000 \times 15$, note that we could just compute the matrix V (which is 15×15), then get as many columns of U as we want.