RBF Summary

1. What is a Radial Basis Function? A radial basis function (RBF) is a general model to build a mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$.

2. Choices that need to be made when using an RBF:

   (a) The transfer function, $\phi(r)$ that will be used. Some choices: $\phi(r) = r$, $\phi(r) = \mathrm{e}^{-r^2/\sigma^2}$, $\phi(r) = r^3$, etc.

   (b) The model parameters for an RBF (given a data set, these are the parameters we need to determine) are:

   - The centers, $\mathbf{c}_i$ which are also in $\mathbb{R}^n$.
   - The number of centers, k (independent of n or m)
     The decision about the number and location of the centers can be made in many ways. Some common choices:
     - The centers are randomly chosen from the data.
     - The centers are chosen as the centroids of data clusters.
     - The centers are chosen automatically using Orthogonal Least Squares (this is Matlab's default).
   - The weights (or coefficients) ω_i, $i = 1, 2, \ldots, k$.

3. Once the centers have been chosen, we solve for the coefficients using a least squares solution. That is:

   - Given p data points in $\mathbb{R}^n$ and k centers in $\mathbb{R}^n$, form the $p \times k$ Euclidean Distance Matrix (EDM) A, so that

     $$A_{ij} = \mathrm{dist}(\mathbf{x}^{(i)}, \mathbf{c}^{(j)}) = \|\mathbf{x}^{(i)} - \mathbf{c}^{(j)}\|$$

   - Form the transfer matrix Φ: $\Phi = \phi(A)$. If biases are desired (and they usually are), add a column of ones to the end of Φ, so that it is $p \times (k+1)$.
   - We solve the matrix equation:

     $$\Phi \cdot W = Y$$

     where W which has size $(k+1) \times m$ contains the weights of the linear combination, and Y (which has size $p \times m$) contains our desired outputs.

4. To test the function on new domain points (we have now fixed the centers and the weight matrix W):

   - Form the EDM. If we have $\hat{p}$ new data points, this will be $\hat{p} \times k$.
   - Form the transfer matrix Φ, which will be $\hat{p} \times (k+1)$.
   - Perform the matrix product ΦW, and this will produce our new output.

5. Why use an RBF?

   (a) The number of parameters needed to define the model does NOT depend on the dimension of the input (n); therefore using an RBF is said to "avoid the curse of dimensionality" (this refers to the explosion in the number of parameters usually needed when the dimension of the domain gets big).

   (b) Once the centers have been chosen, this is a linear modeling problem. That is, to find the weights, we are solving a linear equation. Therefore, an RBF is much faster than methods that involve nonlinear optimization.

   (c) There are nice connections to statistics if we use the Gaussian transfer function, although we have not discussed them.

6. Matlab and the RBF. There are two ways of producing an RBF model in Matlab- one is to do it explicitly, the other is to use Matlab's built-in routines using the "Neural Network Toolbox".

   Below, we assume X, Y are matrices that hold our desired domain and range values. Assume that X, Y are $p \times n$ and $p \times m$, respectively so that we present the data row-wise.

   - Explicit computations: Here we assume that the k centers have been chosen and are in a matrix C that is $k \times n$. Recall that we wrote our own *edm.m* function:

     ```
     A=edm(X,C);             %A is p x k
     Phi=exp(-A.^2./0.1);    %Put in the transfer function here
     Phi=[Phi, ones(p,1));   %Remember to add the ones
     W=pinv(Phi)*Y;          %Or we could use the SVD of Phi
     ```

     To get new output using new domain data in a matrix P,

     ```
     A=edm(P,C);
     Phi=exp(-A^2./0.1);
     Phi=[Phi, ones(p,1));   %Remember to add the ones
     newY=Phi*W;
     ```

   - Using Matlab's functions (Uses Orthogonal Least Squares to determine the number and placement of centers, also forces us to use Gaussian transfer functions).

     ```
     eg=0.1  %Set the error goal
             %used in determining the number of centers
     sp=0.5  %Set the spread of the Gaussians
     net=newrb(X,Y,eg,sp);
     ```

     In this case, `net` is a data structure that contains the weights and some other information. To use the network on new data P,

     ```
     newY=sim(net,P)
     ```