

Assignment 4:
 Course Value: 100 points

Deadline: May 3, 2011

Recursive least squares training of non-linear discrete time models.

Obtain input – output data of the radial inverted pendulum free response. This will be demonstrated in lab. Consider the first half of your data set to be ‘training’ data. Use Recursive Least squares applied to the training data to determine the best second-order discrete time RBF model of the system, using delayed samples of x_1 and x_4 as inputs and current sample of x_1 as the output. The model will have the form:

$$y(k) = f(x_1(k-1), x_2(k-2), x_4(k-1), x_4(k-2))$$

Where the RBF will learn the non-linear function $f(\cdot)$.

RLS training can be illustrated as below:

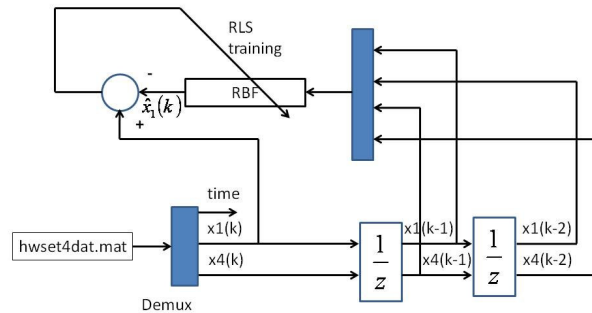
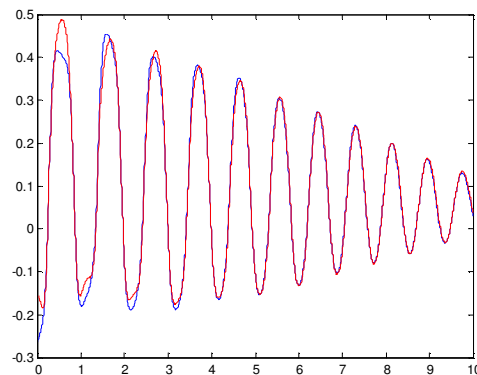


Figure 1: RBF network training

The basis functions will be four-dimensional due to the four inputs. This creates a ‘dimensionality curse’ since the total number of basis functions will be the number along each dimension raised to the fourth power, I.E. n^4 . Try using two basis functions along each direction (total = $2^4=16$), then modify your code to use four functions along each direction (total = $4^4=256$).

Validate your training by running the trained network as a one step ahead predictor on the training data set. This means repeating the simulation shown above without the training feedback (holding the RBF output weights constant). Compare your prediction with the actual data. A sample plot appears to the right.



Finally, demonstrate the network ability to predict x_1 many steps ahead. That is, modify Figure 1 such that the RBF uses past samples of $\hat{x}_1(k)$ rather than the actual data. In this case, x_4 is treated as an input.

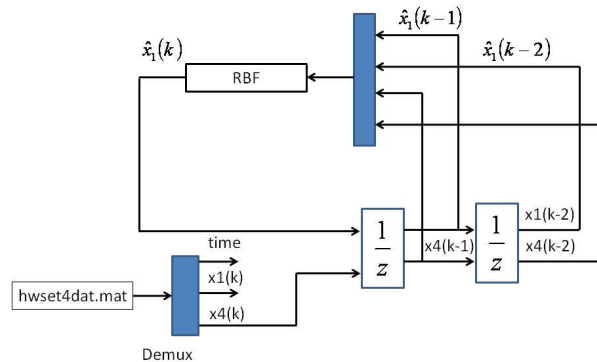
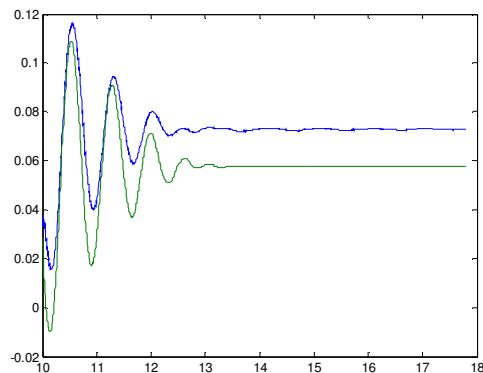


Figure 2: Many step ahead prediction

Make a comparison plot between the actual data and your many step ahead prediction. A sample plot is shown below.



Hints: Set up your code such that the center and variance of basis functions are dependent upon the maximum amplitudes of the signals. Make it easy to adjust the variance based on the distance between adjacent functions in each direction. For instance, I use the code:

```
C(1,1) = 8*min(diff(M(1,:))); % (5/3)^2
```

The ϕ vector (see RLS notes nomenclature) will be a column of length $4^4=256$ for the large network case. To contrive this, you will need to loop through all 256 combinations of basis function means (varying the mean along each of the four dimensions). Rather than using nested loops, I suggest a single loop that converts the loop counter binary, then uses 'quaternary' encoding to choose each combination. Here is some sample code:

```
for n = 0:255 % for four rbf per dim
    ns = dec2bin(n,8);
    n1234 = bin2dec(ns([1 2; 3 4; 5 6; 7 8]))+1;
```

'n1234' is a vector of four numbers that will cycle thru all 256 possible combinations of the indices 1:4