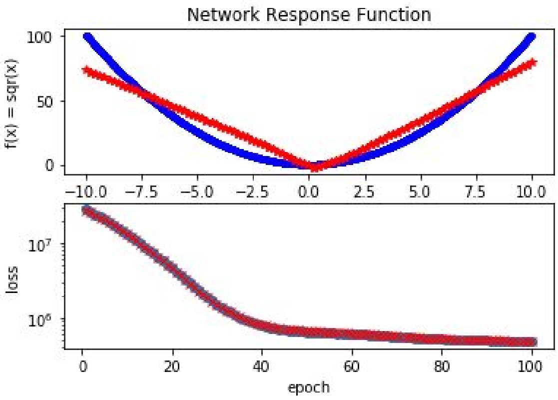
For this assignment, we adapted the given MLP and added a second hidden layer to the script with 50 hidden nodes. Then, we made use of a node's dropout with 50% probability, to evaluate the difference that occurred to the cost function. Also, we examined the effect of batch training, namely divided the dataset into smaller parts and finally we merged the two methods in order to evaluate the combined performance.

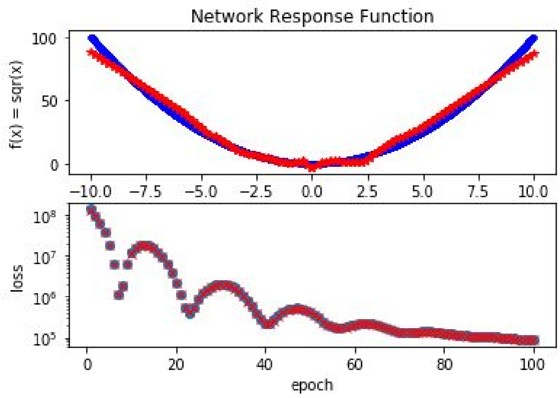
1. After comparing the two methods, with 1 and 2 hidden layers respectively (and for the same number of training epochs equal to 100), it is obvious that the MLP with 2 layers is much more accurate, but does not give the ideal solution. If we increase the epochs to 1000, we observe lower cost and more accuracy. Also, we can see that the loss graph is linear for the first 40 epochs with 1 hidden layer, while with 2 layers it has many ups and downs.

1 hidden layer




```
Loss function for the final epoch =  479062.8  (test data)
Pseudo chi^2 = loss/Ndata          = 47.906281 (test data)
Training phase finished
```

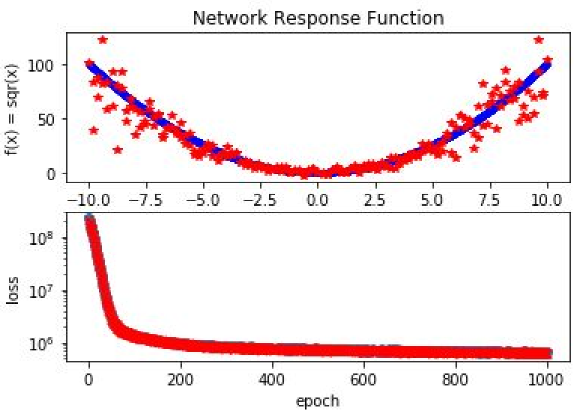
2 hidden layers




```
Loss function for the final epoch =  86315.53  (test data)
Pseudo chi^2 = loss/Ndata          = 8.631553 (test data)
Training phase finished
```

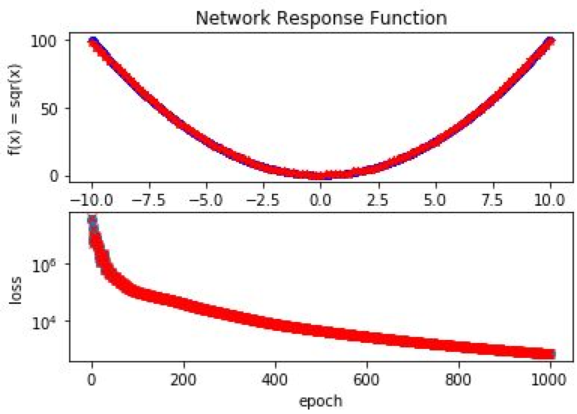
2. We compare the cost evolution with and without the use of dropout for a keep probability of 50% and can notice that, although the loss seems to become more robust with the use of dropout, it is clearly less accurate than the one without dropout. Also, the loss value reaches a stable value, which is 2 orders of magnitude higher with the use of dropout than without. This result seems rational concerning the small amount of data used in this MLP.(training epochs set to 1000)

with use of dropout(2 hidden layers)




```
Loss function for the final epoch =  635763.25  (test data)
Pseudo chi^2 = loss/Ndata          = 63.576325 (test data)
Training phase finished
```

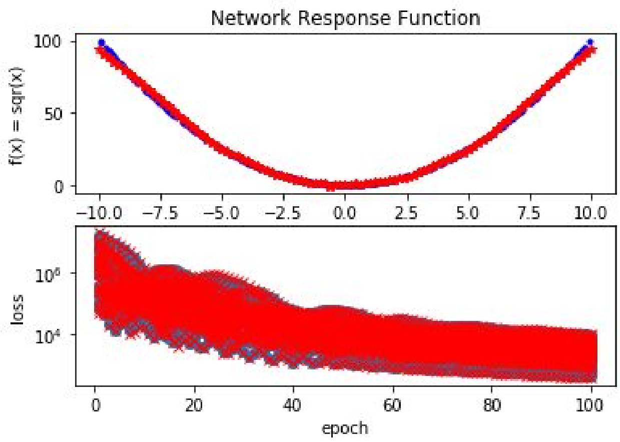
without use of dropout(2 hidden layers)




```
Loss function for the final epoch =  679.62463  (test data)
Pseudo chi^2 = loss/Ndata          = 0.067962 (test data)
Training phase finished
```

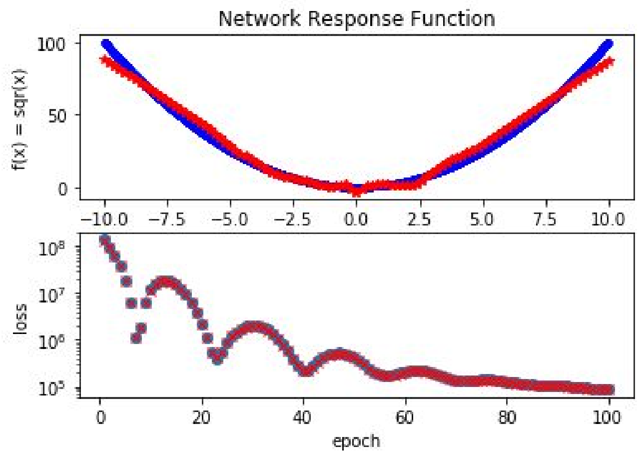
3. The concept of batch training is to divide the given dataset into smaller groups of training samples. We determine the optimal batch size, hence the batch number. So each time, one of the 30 batches runs and then follows the next one, until all of them are trained. The training epochs for each batch is set to 100. What we concluded after running the MLP is that every time that a batch is trained, the loss graph decreases even more. By using different batch sizes we can notice that, the smaller the batch size is , the lower loss value and the more accurate graph we get .

batches




```
Loss function for the final epoch =  345.31873  (test data)
Pseudo chi^2 = loss/Ndata          = 0.034532 (test data)
Training phase finished
```

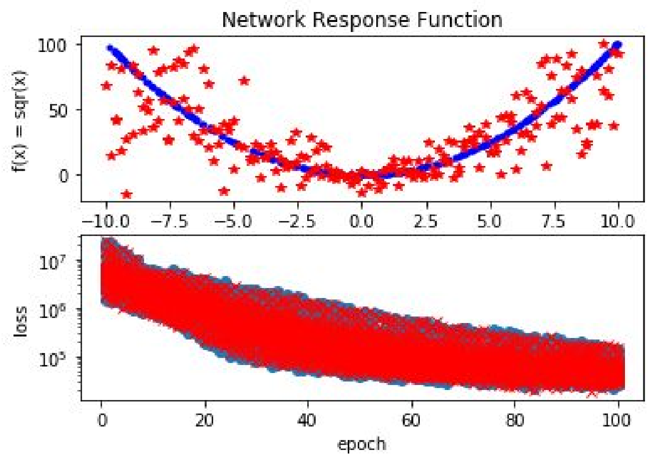
no batches




```
Loss function for the final epoch =  86315.53  (test data)
Pseudo chi^2 = loss/Ndata          = 8.631553 (test data)
Training phase finished
```

The thick red graph in the loss plot is a combination of the separate graphs for each batch. The highest value line belongs to the first trained batch and the lowest one to the last batch. This illustrates how much the loss of the MLP improved till the end of the training.

4. For this last scenario, we merge the batch training with the dropout to explore the potential of using both methods. We used dropout with keep probability of 50% and split the dataset into 30 batches. By comparing the different plots and the numerical results, we can notice that the combined performance of using both methods is worse than the one with the use of batch training and similar with the one that we make no use of batches.




```
Loss function for the final epoch =  86454.37  (test data)
Pseudo chi^2 = loss/Ndata          = 8.645437 (test data)
Training phase finished
```