When deciding how to design an artificial neural network there are several decisions one could make when creating the implementation. Some choices which have been already made for this network are:

* The type of ANN
* Subject to learn
* How clean the data is
* Number of inputs
* Number of outputs

One potential decision that has already been made is the type of ANN. In this case we are to implement a simple back propagation network. Another is what the neural network will be made to learn about. In this case we have been provided two functions that the network will need to learn to approximate. Another decision is (which may not necessarily be a realistic choice in a real world scenario) is how clean the learning data is. In this case we will implement one network that has ideal data as an input, and another that has noise incorporated into the learning. When designing a network, one aspect that needs to be defined for implementation is picking the number of inputs and outputs. Depending on what you are attempting to model there are many choices that could potentially be made. Such as whether or not to include certain aspects of a system as inputs to the network. In this case we were provided an already defined function with a given number of inputs and outputs, so there are no decisions to be made in this regard.

Even with these portions of the network already decided before beginning, there is still more to decide on to complete the network, such as:

* Network Depth (number of hidden layers)
* Network Width (number of neurons in each layer)
* Bias Neuron
* Momentum
* Threshold
* Maximum Iterations

One is to decide how deep the network is. Could it translate directly from inputs to outputs? Since the given functions are not linear then the answer is no. Instead the network will need one or more hidden layers. Another decision is how wide the network is, or how many neurons to have in each network. Adding more will allow the network to learn better, but it will slow down the process with added computation time. Another choice is whether or not to include a bias neuron for each layer, as this will allow the decision boundary for a given neuron to deviate from the origin. Another is whether or not to include a momentum parameter, and if so then deciding on a value for it. Then there is the value of the Threshold, or at what point the error is sufficiently small as to stop training the network. In a similar vein there's the value of the maximum number of iterations to complete if the threshold value is never met.

Tuning these value will come down to experimentation because it can be hard to predict exactly what effect they will have. I will choose moderate default values to start with for all the networks, and then tweak them to try to get a better performing network.

Network Depth: I have chosen one hidden layer for the one input function, and two hidden layers for the two input function. No particular reason, just seems like a good place to start.

Network Width: I have chosen 25 neurons for the first hidden layer. This seems like a lot compared to the typical examples I’ve seen, and I chose this since the network is implemented in C++ and runs pretty fast, so this number is computed reasonably quickly, and it seems like it should help it learn better.

Bias Neuron:

Momentum:

Threshold:

I chose to include all of these since they seem relatively standard for neural networks and I just decided to keep the values that were used in the sample implementation code I have. Bias +1, momentum -1, and threshold 0.0005f.

Maximum iterations:

Hopefully the network will work so well that this will be unnecessary, but otherwise I will just adjust this to finish after 10 or 20 seconds if it hasn’t met the threshold yet. I’ll start this out at 10000 or so but it’ll be dependent on the actual results I’m seeing.