Chapter 6

Analyzing the Overall Performance of the Designed Neuro Immune Network

Naturally acquired immunity is that immunity acquired upon exposure to a specific pathogen particularly in the course of a disease. In Chapter one, we described the generation of memory cells as a result of the specific-response of B-cells humoral response. Humoral immunity depends first on the ability of B-lymphocytes to recognize specific antigens and second on their ability to initiate responses that protect the body against it. In most of the cases the response is the production of antibodies that will deactivate the antigen and lead to destruction of the infected organisms.

B cells produce only a single kind of antibody that has the ability to bind to only a single type of epitope. These antibodies are displayed on the surface of B cells, and interact with antigens when necessary. The binding of an antigen to one of these surface B cells stimulates those cells either to start producing antibody or to differentiate into cells that produce the antibody. These antigens are found on the surface of pathogens, or displayed by other immune system cells such as macrophages. Be-Cells differentiate into plasma cells, which can immediately produce antibody molecules, and memory-cells that are more long-term-stable B cells that can differentiate into plasma cells. When an Ag attacks a host, the immune system produces two responses: the primary response, and the second response. Upon initial exposure to an antigen that a body has never been exposed to, the primary response starts and there is a 10 to 17 day lag before the peak in antibody concentration. The secondary response starts when an antibody is exposed to a previously seen antigen and there is only a 2 to 7 day lag before the peak in antibody concentration. This is why your body is able to rapidly defeat many of the pathogens to which you have been previously exposed [10].

1. Testing the Acquired Immunity

In our designed immune network the acquired immunity is represented by the pre-trained AIM. In the previous chapter we concentrated on teaching the AIM network how to recognize the training set to attain acceptable performance, and keep the generality feature as well [48]. It should be prepared to deal with new situations, or at least be able to give a good starting point as a rapid immune memory response. The AIM cannot deal with every new case, but the generalization feature can give a good estimate for the initial immune response. Later, this new performance can be incorporated into the existing AIM, and increase the number of training examples. If we reflect this idea into the robot workspace, we will find that it which is very hard to feed the AIM with every single point in the 3D space a the AIN is learning. Figure6.1 illustrates the test of the pre-trained AIM to reach a target point at (0,0,150).

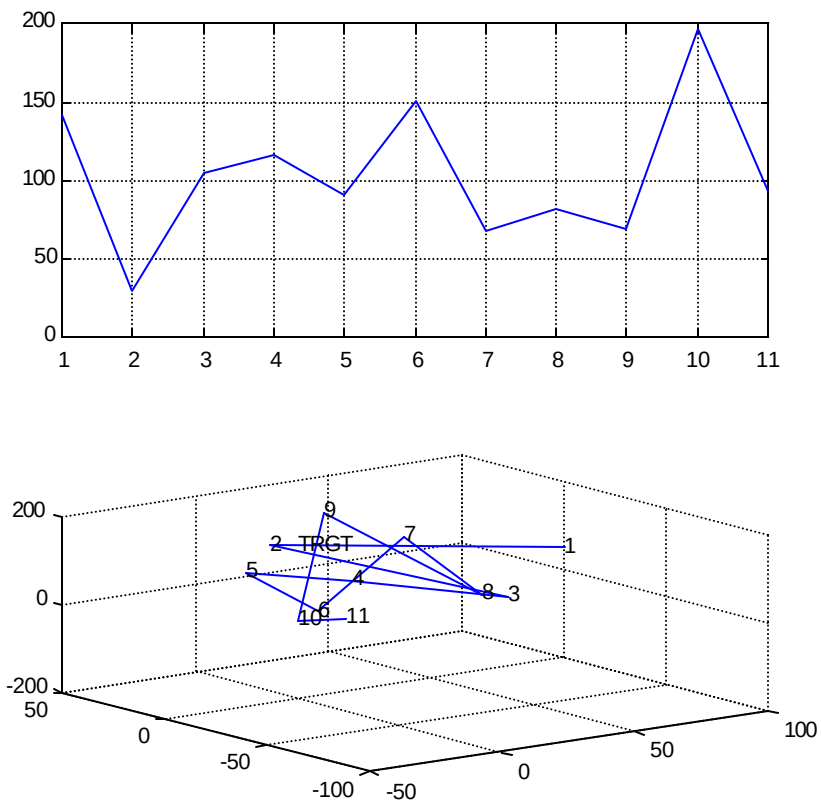


Figure6.1 AIM Response to Reach a Target Point at (0,0,150)

The first graph of Figure6.1 shows the distance of the end-effector from the target point as the AIM network is used solely to reach the target. From the biological point of view, this is not the actual behavior of the immune system. The benefit of the acquired immunity is to speed up the initial response, and to give the immune systems useful information about the invading pathogen. This is a logical behavior, because memory cells normally decay as time passes, and as a result it cannot be used as only the tool to defeat the pathogen.

2. The Combined AIM-AIN Performance

If we analyze the first graph, will find a large drop in the distance value from 141.4 to 29.8, which is a good start up point for the immune response. If we compare the one-step distance response with the number of steps that has resulted from using the AIN alone, we will find that it took approximately 11 steps to approach this distance. This is a very good response, but in the next iteration in the AIM solution we will be facing a completely new staring point. This means different values for $\pm\Delta X$, $\pm\Delta Y$, and $\pm\Delta Z$ will be fed to the AIM network to obtain the next response. This is an unpredictable response for the AIM network, because we do not know if the new input is close to training set examples or not.

We propose a solution for this problem that senses the point at which the AIM response is starting to malfunction, and then switch control to the AIN to finish the simplified problem. The performance of the AIM is evaluated each iteration, and if the end-effector starts to move away from the target, then this means that it is the turn for the AIN to learn how to deal with this new situation, see Figure6.2. Even though the control is with the AIN, the history of AIM processing is still involved, and its response is counted. After the AIN learns how to reach the desired position, the total situation response is the summation of the AIM and the AIN response. This total response is added to the training file, and a re-training process takes place again to add this situation to the immune system expertise.

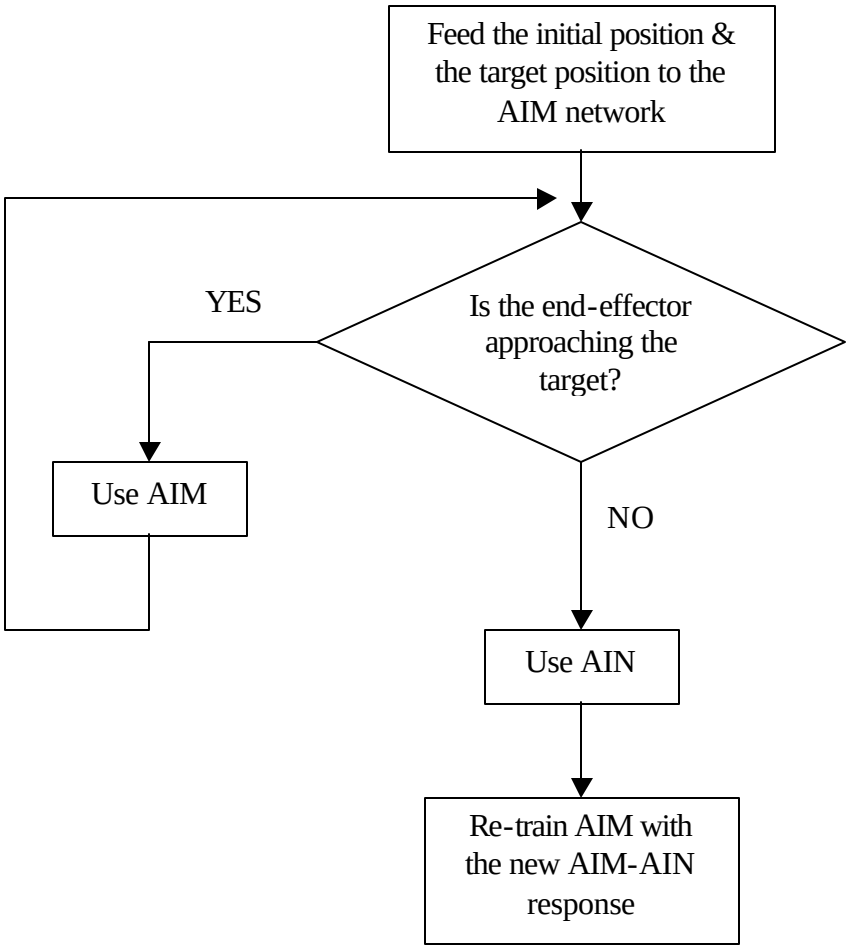


Figure6.2 The Neuro-Immune Treatment with New Situations

We modified our AIM simulation program to include AIN processing if necessary. In Figure6.1 we see that the values of the distance from the end-effector has improved at step 2, but it returned to have an abrupt increase at step 3. This is where the AIN learning will take place again after the AIM has reached its ultimate knowledge about this new encountered situation. The AIN network starts to deal with this situation, but from a better starting point that has a distance equal to 30 from the target point. Figure6.3 illustrates the change that is made by calling the AIN after reaching the ultimate performance of the AIM network. Figure6.4 gives a clear view with different range limits, so that we can notice the fine-tuning done by the AIN performance, see also Figure6.5 (a,b).

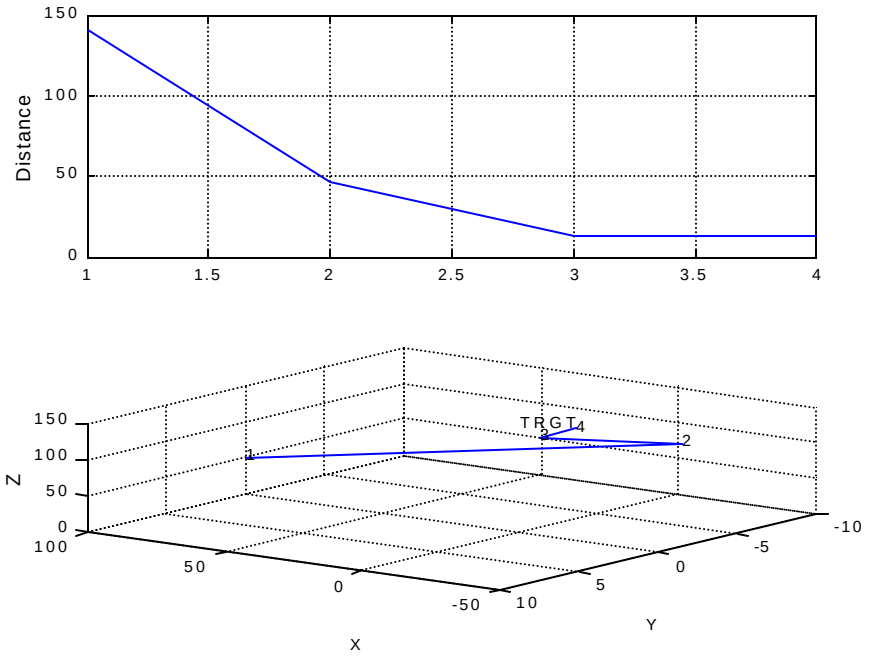


Fig6.3 AIM-AIN Performance

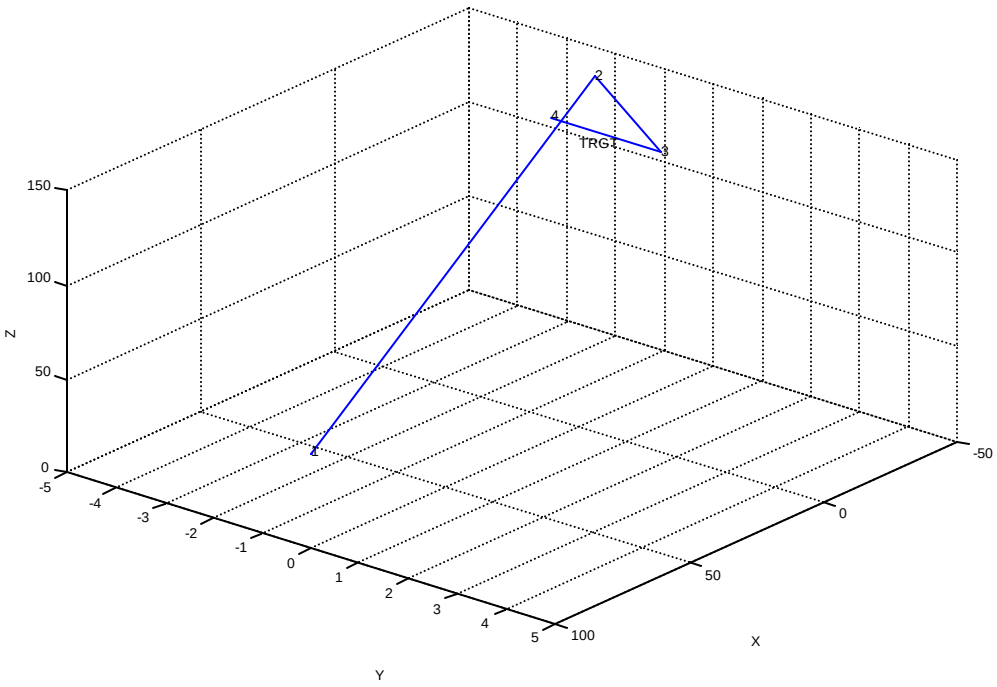
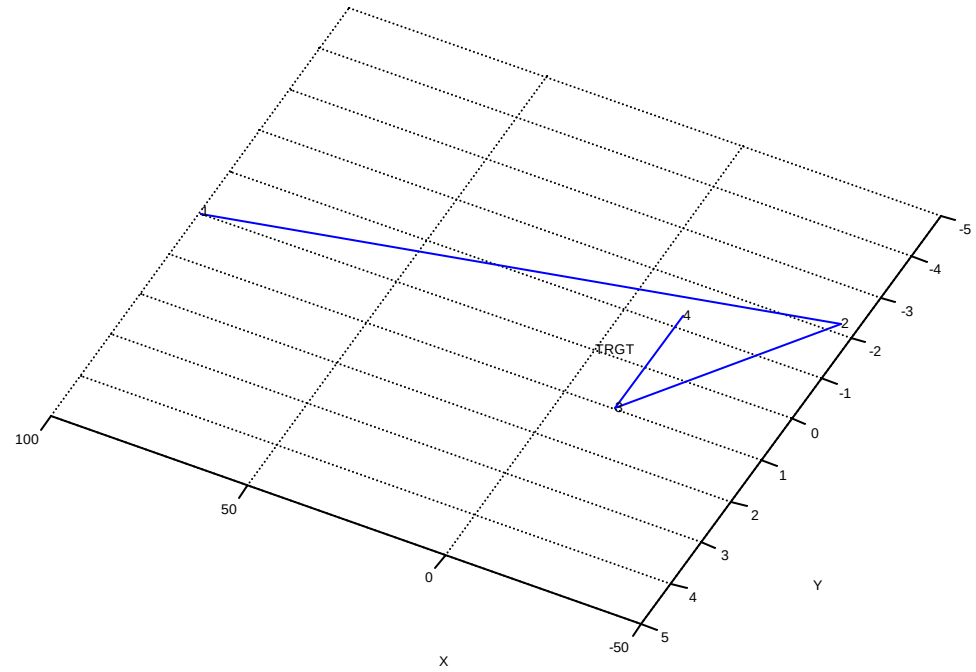
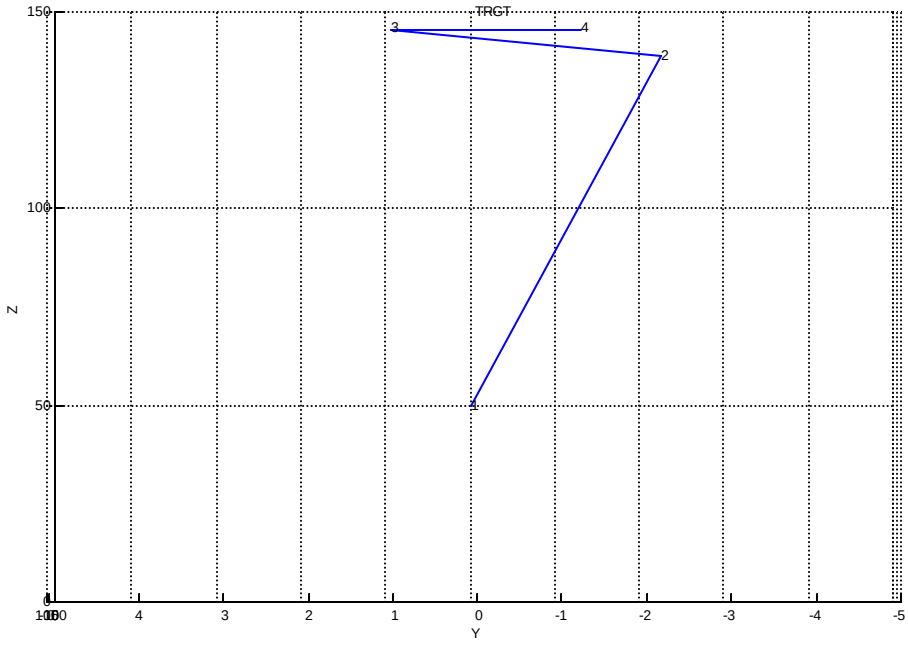


Fig6.4 End-effector Motion for the AIM-AIN Performance



(a)



(b)

Fig6.5 (a) XY Plane Projection (b) ZY Plane Projection
End-Effector Motion for the AIM-AIN Performance

The new AIM-AIN performance needs to be inserted into the training file knowledge base for future processing. A pre-training session should be performed to incorporate this new change into the old AIM network. We need to re-train each single network involved in the AIM training process with this new training example. By doing a simple one-pass analysis process over the last AIN training, we can obtain the necessary value for the new re-training process. The outcome of the re-training sessions will differ from one net to another inside the AIM network, and a regression analysis phase will be needed to judge the usefulness of these re-trained networks.

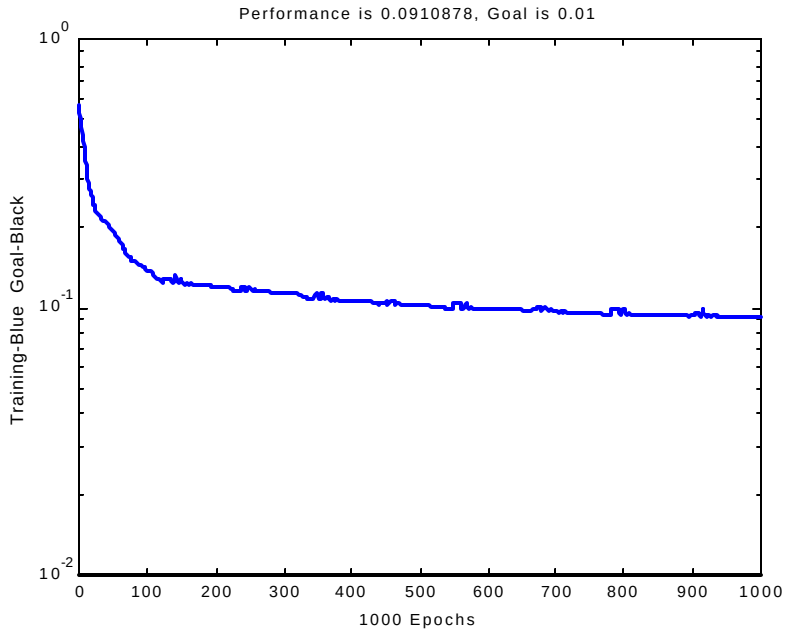


Figure6.6 (a) NN#1 Performance Analysis Before Adding The New Training Example

Figure6.6 (a) represents the performance of NN#1 before the new example is fed into the training set. Compared to Figure6.7 (a) we find that the performance has changed from 0.09 to 0.12 which is a bad compared to the desired goal. R1 has changed from 0.81 to 0.63, and R2 changed from 0.81 to 0.83. So, adding the previous example to the training set, and re-training it did not improve the performance; see Figure6.6 (b, c) and Figure6.7 (b, c).

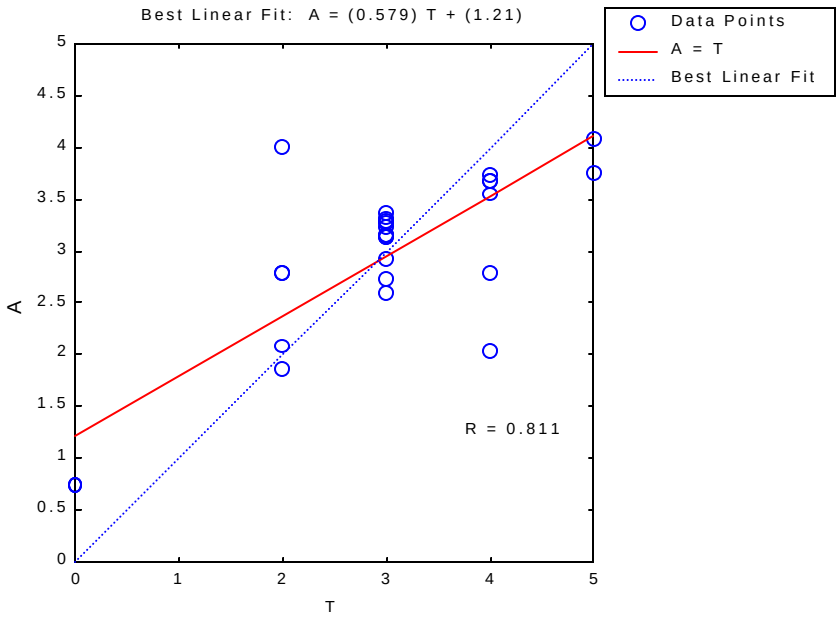


Figure6.6 (b) Regression Analysis for the First Target Value of NN#1

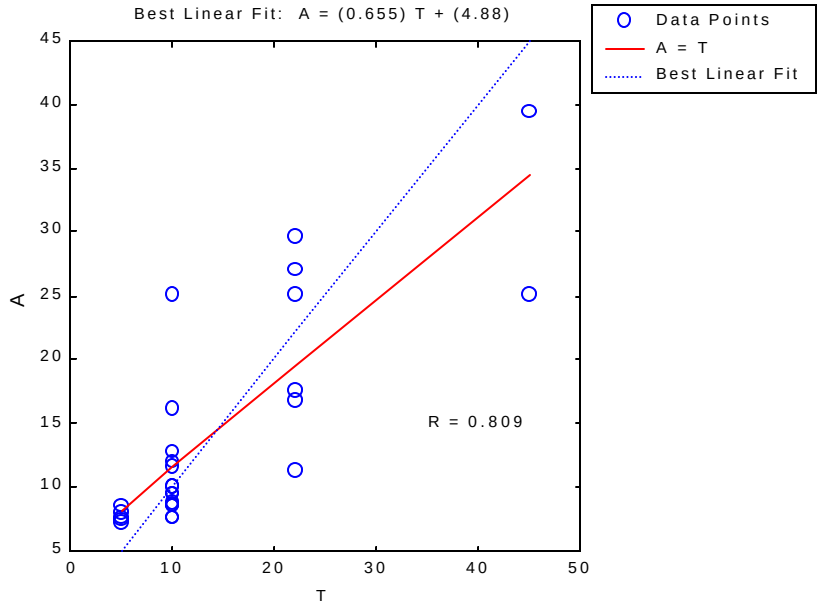


Figure6.6 (c) Regression Analysis for the Second Target Value of NN#1

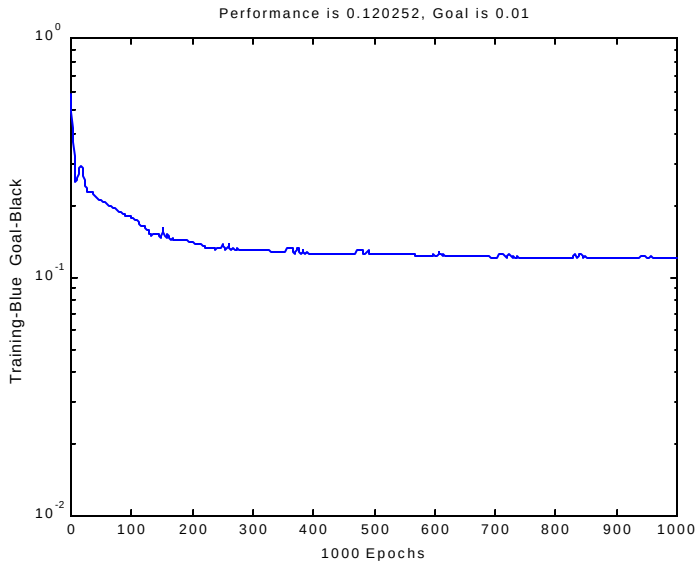


Figure6.7 (a) NN#1 Performance Analysis After Adding the New Training Example

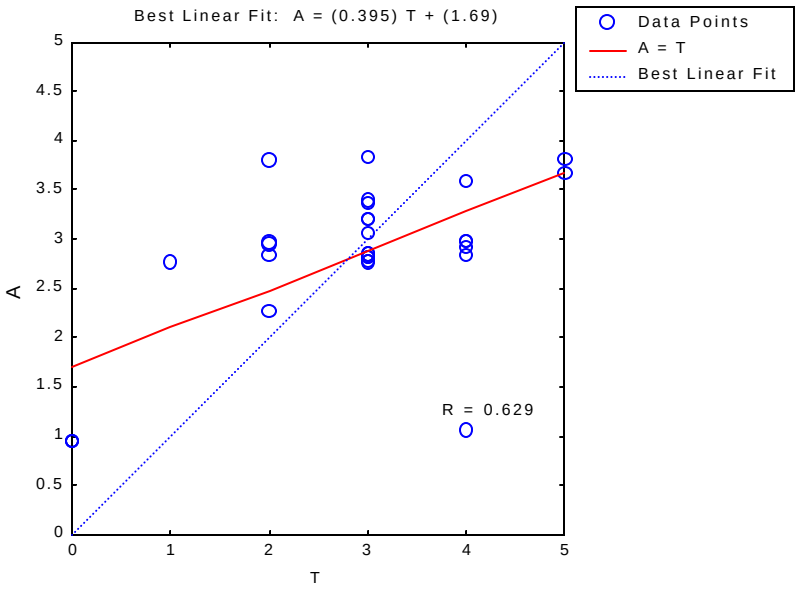


Figure6.7 (b) Regression Analysis for the First Target Value of NN#1

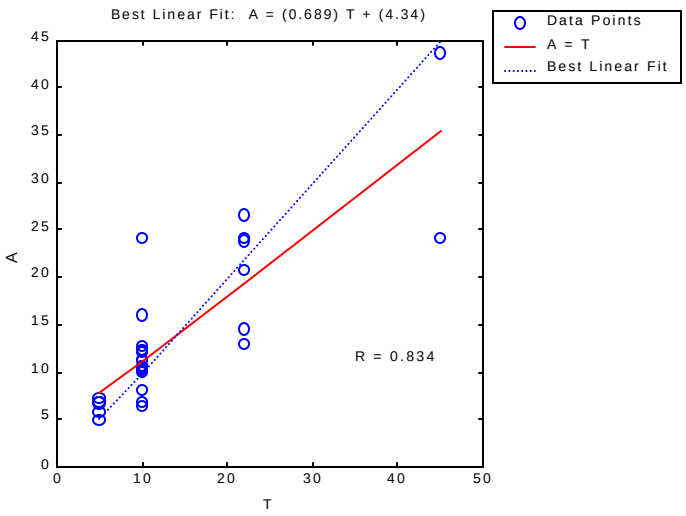


Figure6.7 (c) Regression Analysis for the Second Target Value of NN#1

Figure6.8 (a) represents the performance of NN#2 before the new example is fed into the training set. Compared to Figure6.9 (a) we find that the performance has changed from 0.072 to 0.067, which is a slight progress toward to the desired goal. R1 has changed from 0.77 to 0.71, and R2 changed from 0.83 to 0.85. So, adding the previous example to the training set, and re-training it makes a slight improvement toward the performance; see Figure6.8 (b, c) and Figure6.9 (b, c).

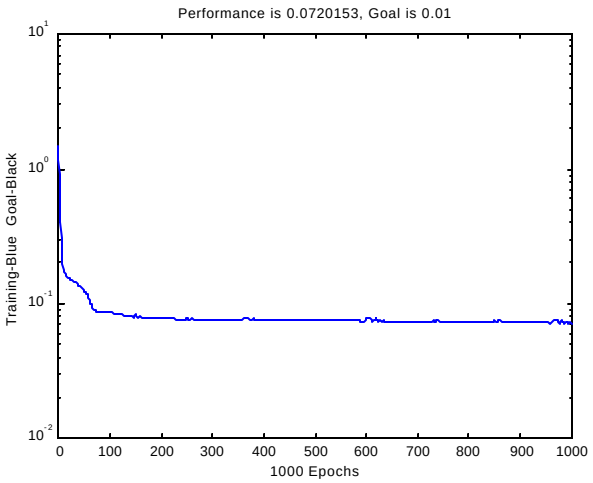


Figure6.8 (a) NN#2 Performance Analysis Before Adding the New Training Example

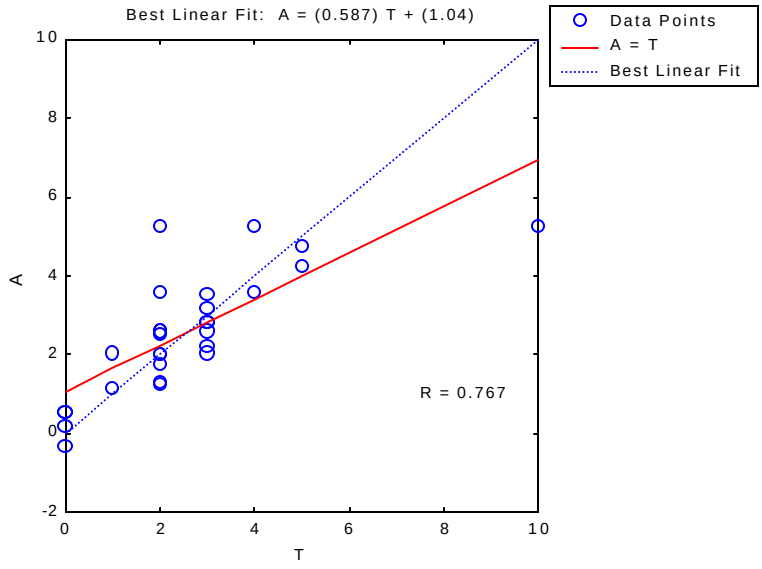


Figure6.8 (b) Regression Analysis for the First Target Value of NN#2

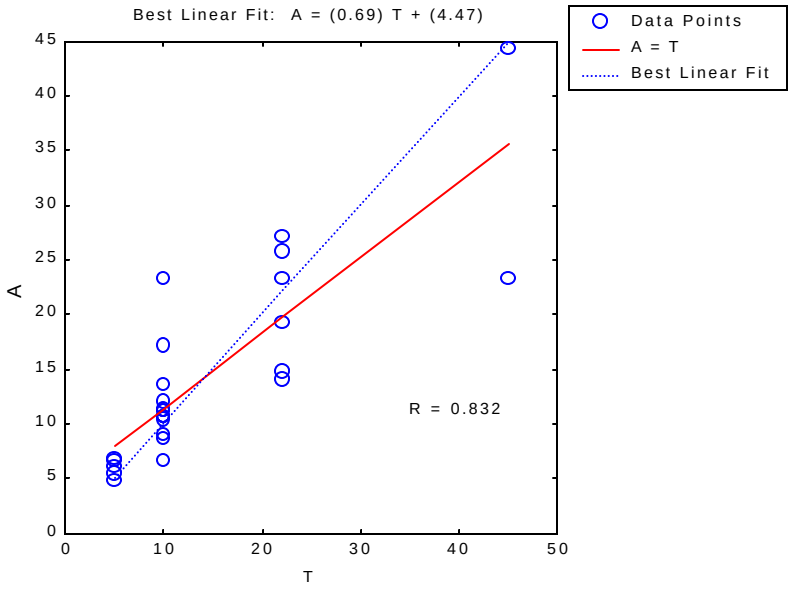


Figure6.8 (c) Regression Analysis for the Second Target Value of NN#2

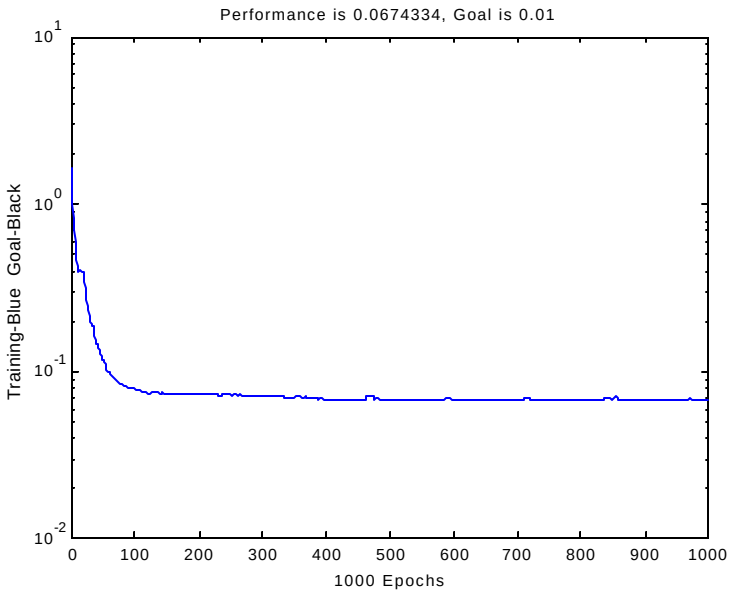


Figure6.9 (a) NN#2 Performance Analysis After Adding the New Training Example

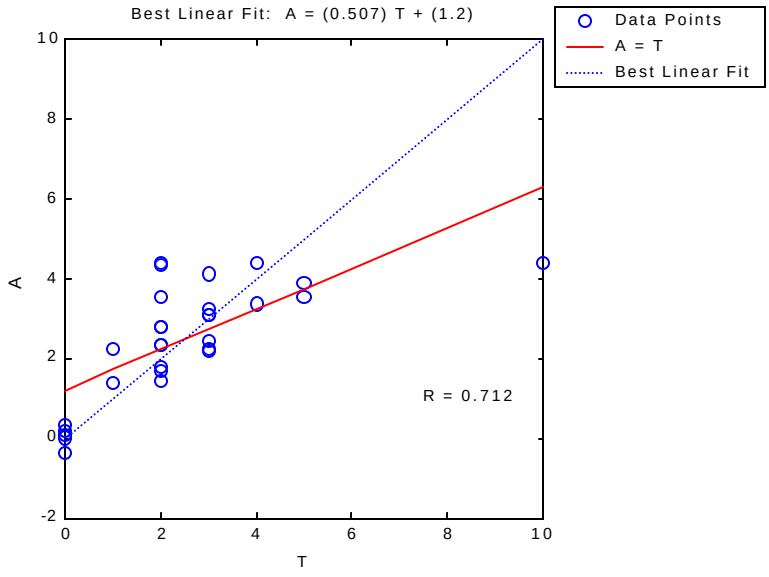


Figure6.9 (b) Regression Analysis for the First Target Value of NN#2

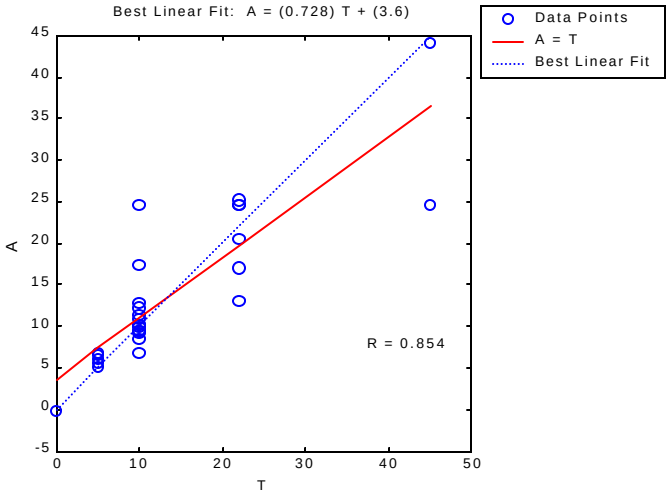


Figure6.9 (c) Regression Analysis for the Second Target Value of NN#2

Figure6.10 (a) represents the performance of NN#3 before the new example is fed into the training set. Compared to Figure6.11 (a) we find that the performance has changed from 0.07 to 0.05 which is a good compared to the desired goal. R1 has changed from 0.84 to 0.9, and R2 changed from 0.83 to 0.85. So, adding the previous example to the training set, and re-training it improves the performance; see Figure6.10 (b, c) and Figure6.11 (b, c).

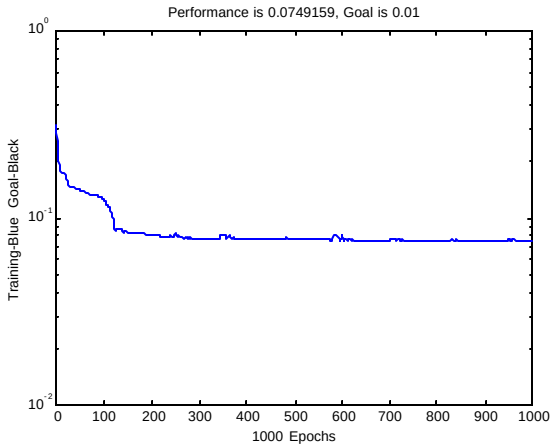


Figure6.10 (a) NN#3 Performance Analysis Before Adding the New Training Example

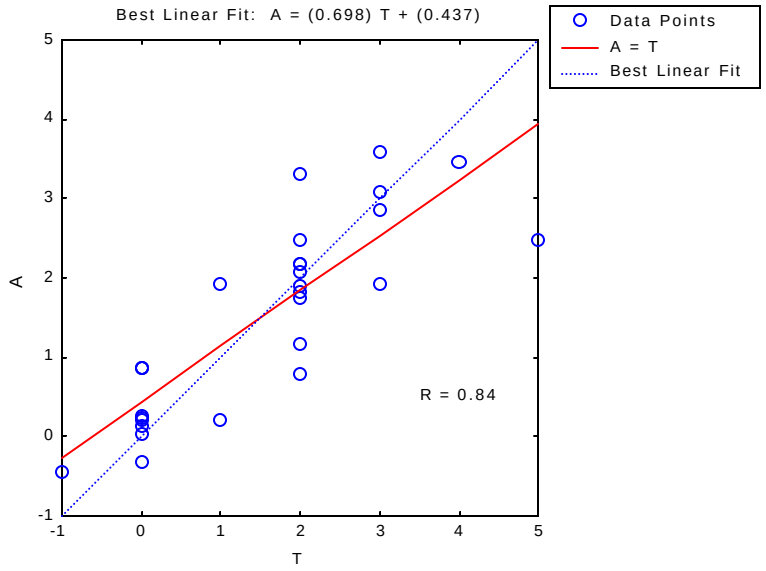


Figure6.10 (b) Regression Analysis for the First Target Value of NN#3

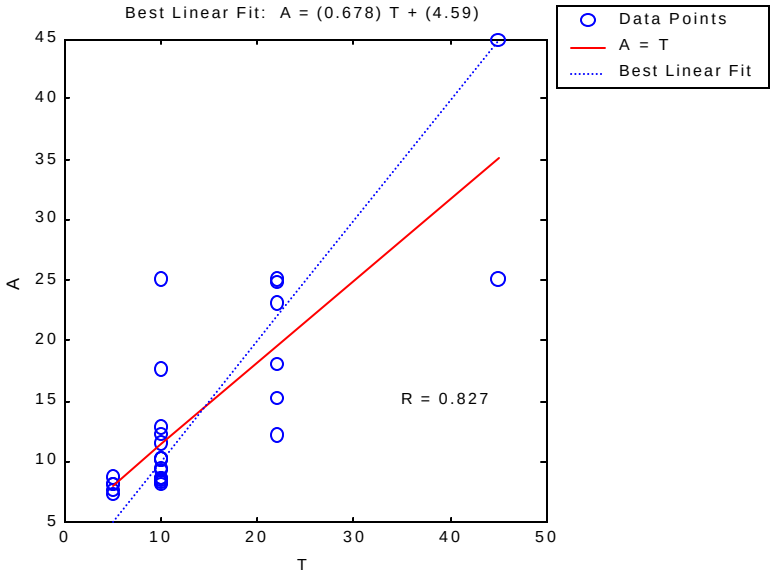


Figure6.10 (c) Regression Analysis for the Second Target Value of NN#3

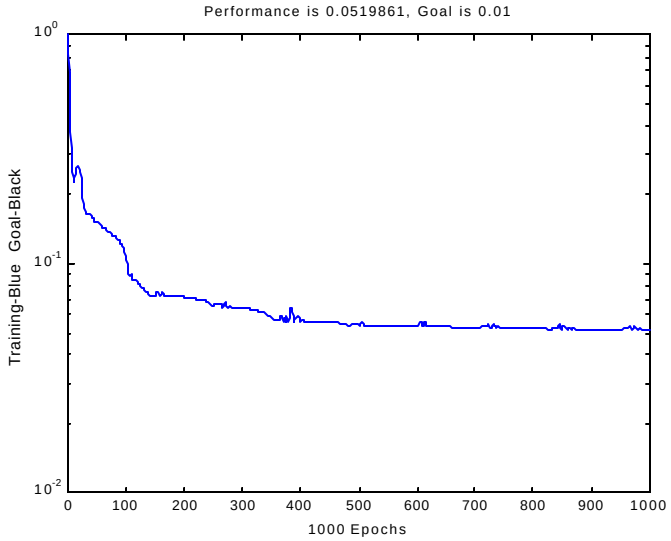


Figure6.11 (a) NN#3 Performance Analysis After Adding the New Training Example

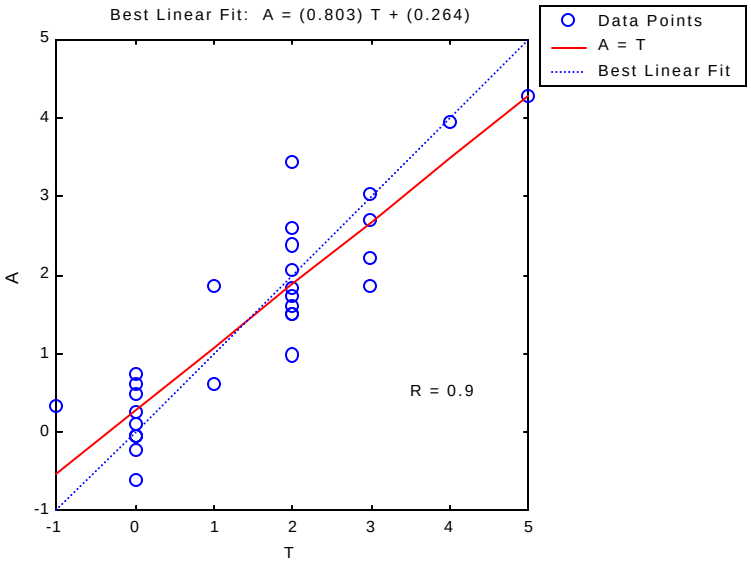


Figure6.11 (b) Regression Analysis for the First Target Value of NN#3

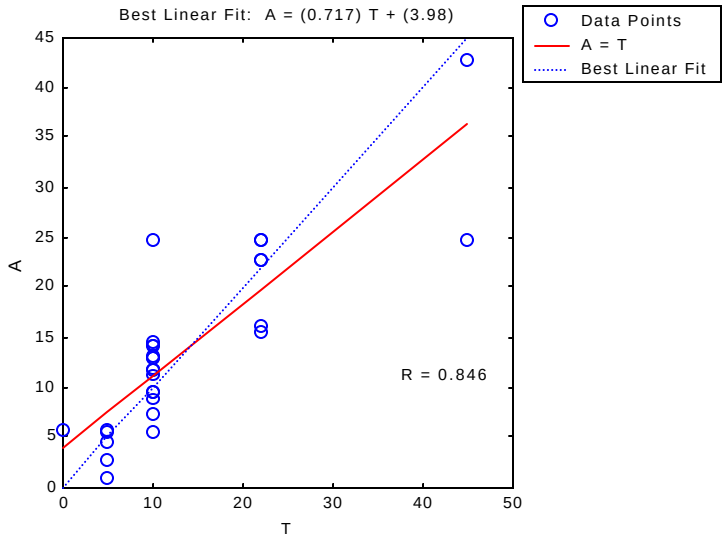


Figure6.11 (c) Regression Analysis for the Second Target Value of NN#3

3. AIM-AIN Performance Validation

The number of steps needed to reach the target point decrease after using the combined AIM-AIN, Neuro-Immune Network (NIN). These results are comparable to the biological immune system, where the memory is used to speed up the response of the host to the invading antigen. Another important point is that, the NIN training program has generated a new training point that is not fed by the user, but generated automatically in the learning process. The previous results show that a single example added to the training set cannot guarantee improvement in the overall performance of the NIN.

In the last example, the new point is generated at a new distance of 29.7, which will generate new delta-difference values for this new position. Changing the target point for the NIN continuously will enlarge the training file. These values will be used to enhance the training process, and could lead to an increase in the NIN performance. We need more than example to enhance the NIN performance, and make it ready to face new a target-points' problem, see Figure6.12. We will show another example to validate the NIN performance, and then try another example on the accumulative training process as well.

Another Example From the Training Set

Target point at (71.9846 60.4023 84.2020)

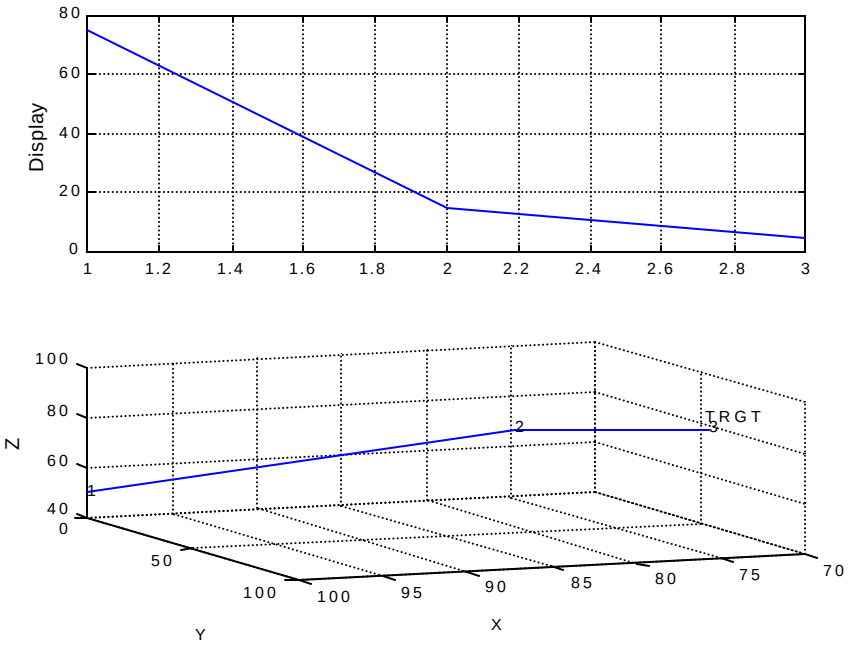


Figure6.12 End-Effector Distance from the Initial Point to the Target Point

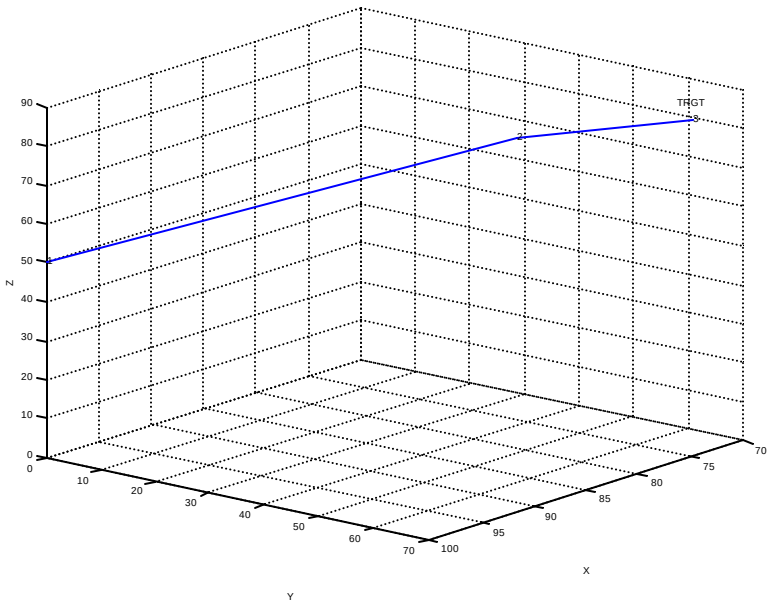


Figure6.13 (a) End-Effector Motion

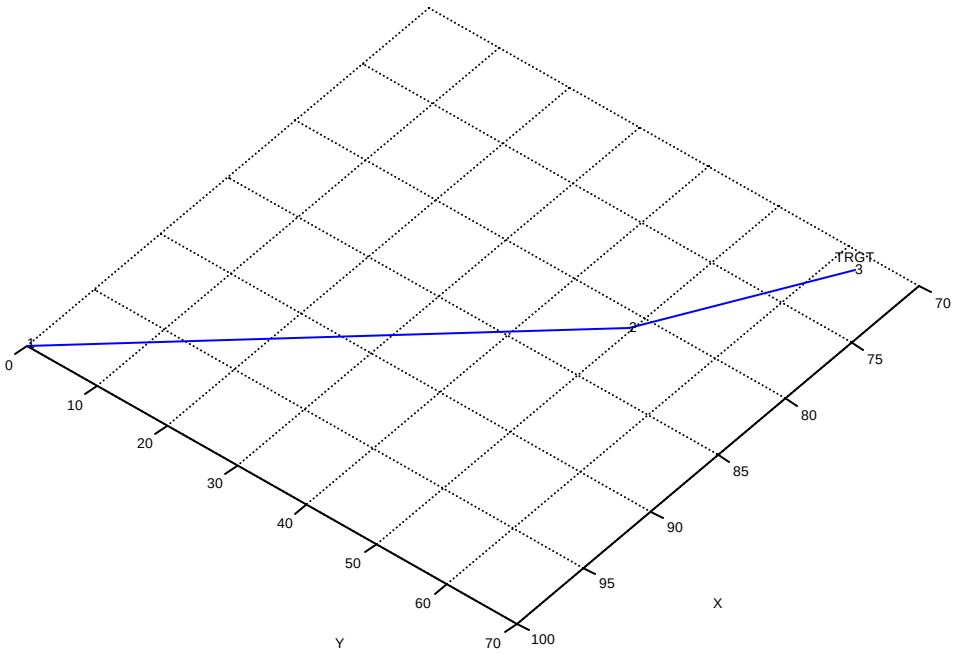


Figure6.13 (b) XY Projection

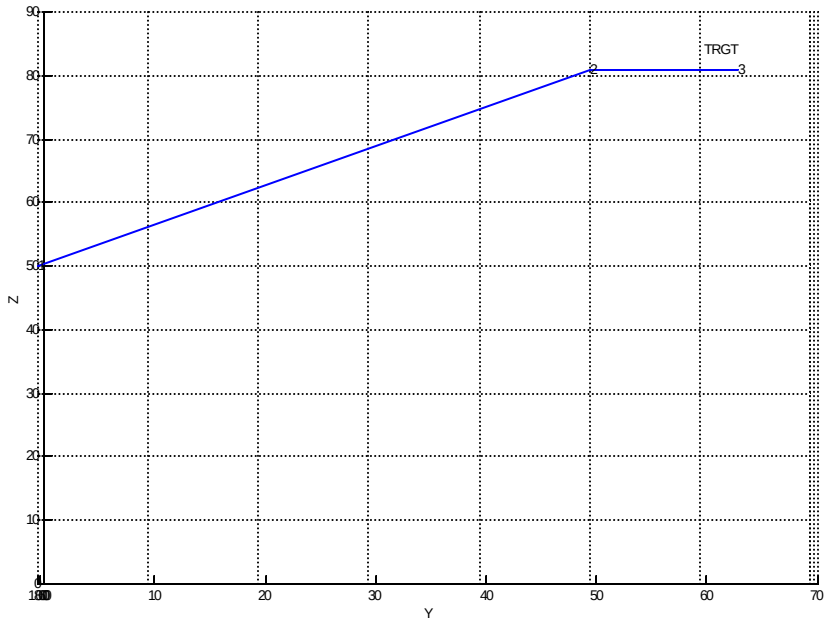


Figure6.13 (c) ZY Projection

If we compare the performance of NIN to the previous AIN, we find that using the using the AIM give a jump-start from an approximate distance of 75 to 18 from the target point: see Figure6.14 (a). It requires 12 movements to reach the target point, and approximately 11 steps to come close to a distance of 18; see Figure6.14 (b). We reach a conclusion that using the AIM enhances the performance of the artificial immune system, but it cannot finish the whole process by itself. As we see in Figure 6.14 (a, b, and c), the AIN took control again after step 2 of the NIN performance. The rest of these figures illustrate the results of the training process, and also the performance after adding the new resulted example to the AIM knowledge base.

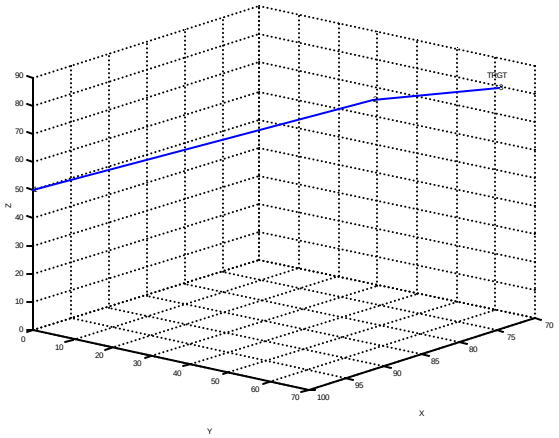


Figure6.14 (a) End-Effector Motion After Using NIN

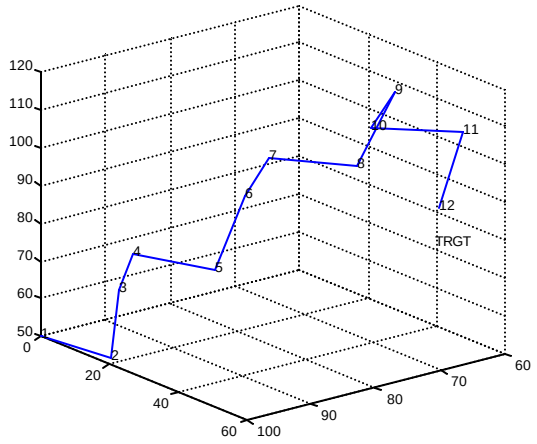


Figure6.14 (b) End-Effector Motion Before Using NIN

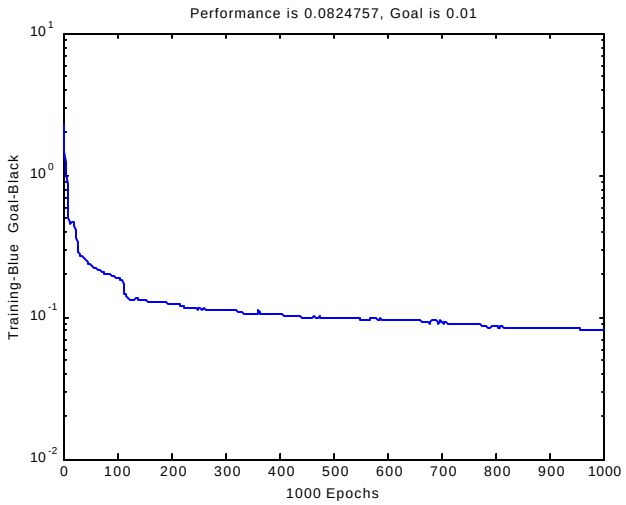


Figure6.15 (a) NN#1 Performance Analysis Before Adding the New Training Example

Figure6.15 (a) represents the performance of NN#1 before the new example is fed into the training set. Compared to Figure6.16 (a) we find that the performance has changed from 0.082 to 0.079 which is a slight change compared to the desired goal. R1 has changed from 0.82 to 0.84, and R2 changed from 0.83 to 0.82. So, adding the previous example to the training set, and re-training maintains the performance; see Figure6.15 (b, c) and Figure6.16 (b, c).

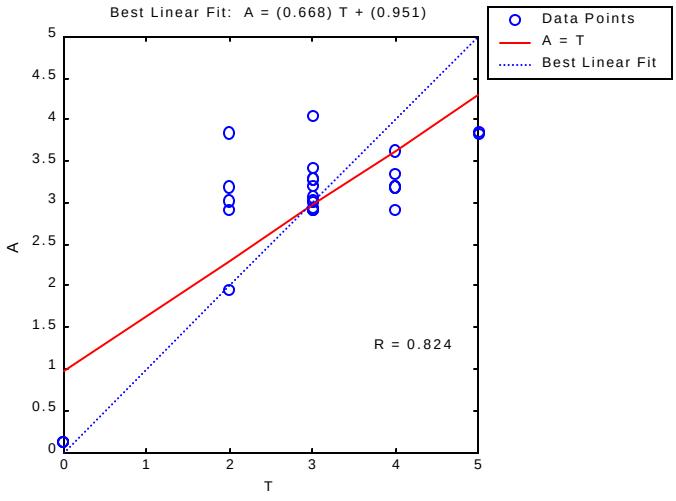


Figure6.15 (b) Regression Analysis for the First Target Value of NN#1

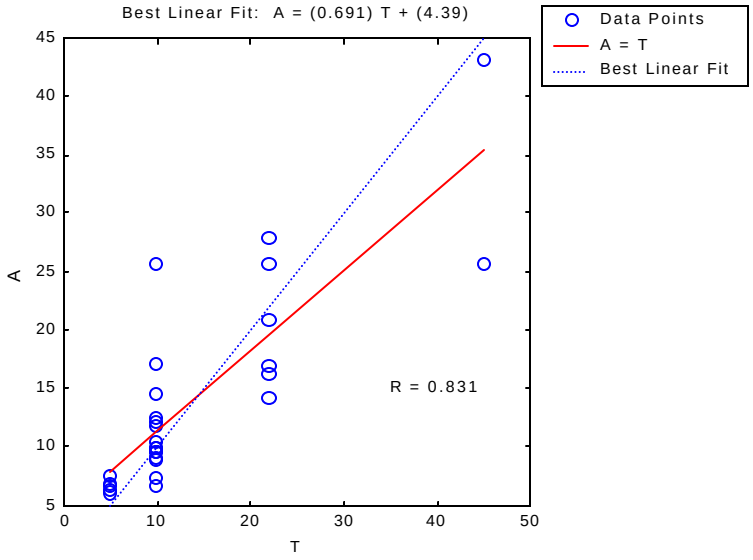


Figure6.15 (c) Regression Analysis for the Second Target Value of NN#1

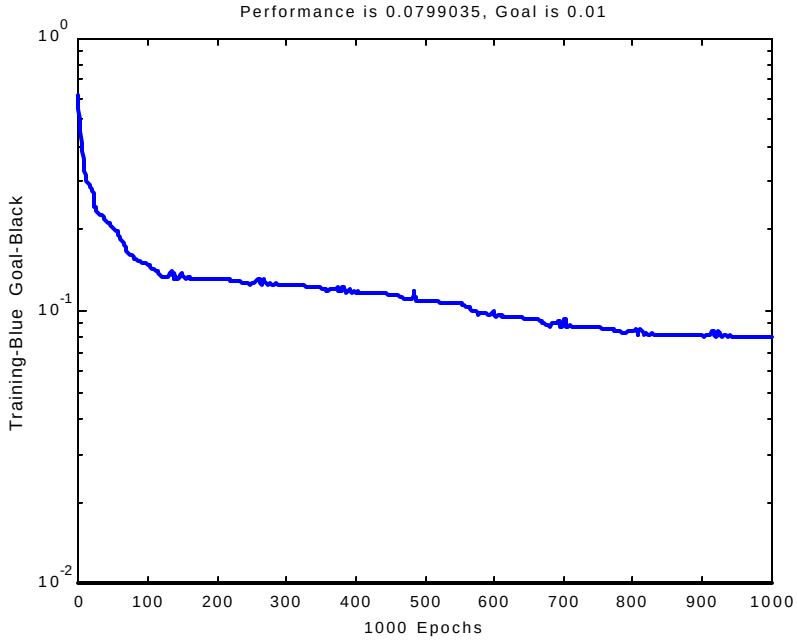


Figure6.16 (a) NN#1 Performance Analysis After Adding the New Training Example

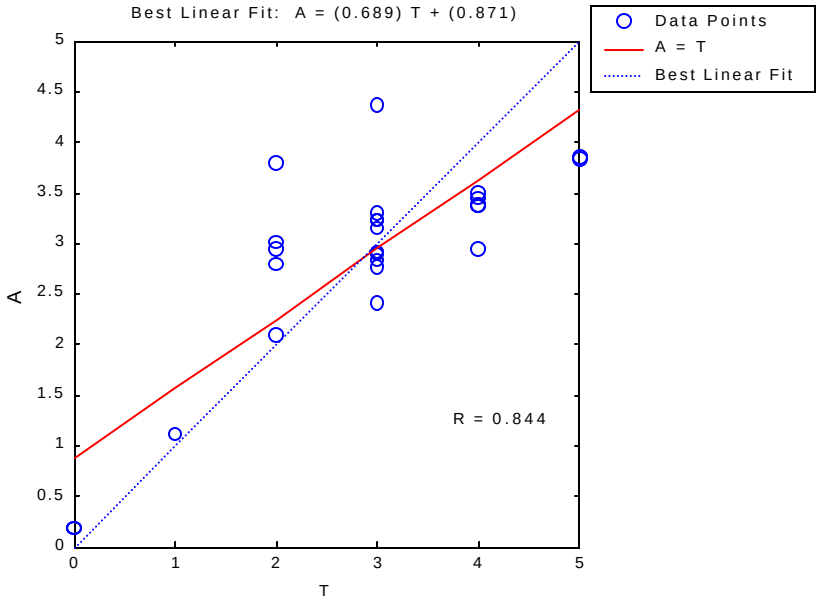


Figure6.16 (b) Regression Analysis for the First Target Value of NN#1

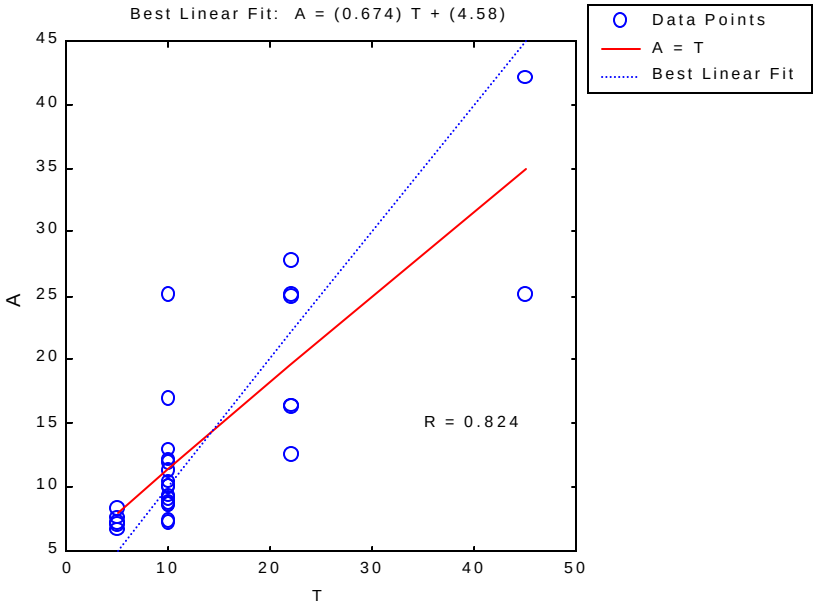


Figure6.16 (c) Regression Analysis for the Second Target Value of NN#1

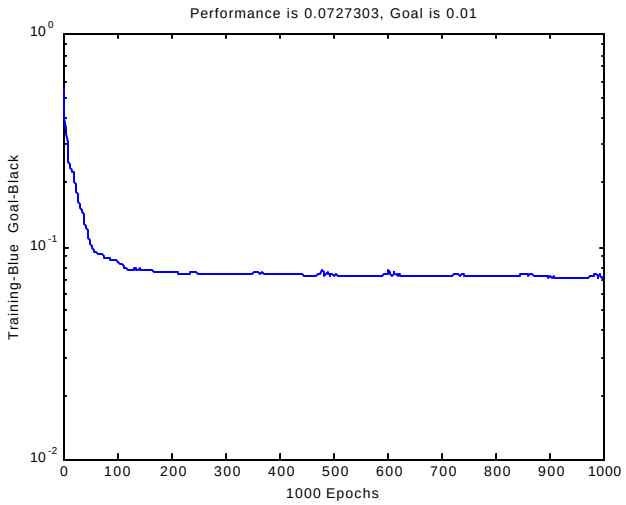


Figure6.17 (a) NN#2 Performance Analysis Before Adding the New Training Example

Figure6.17 (a) represents the performance of NN#2 before the new example is fed into the training set. Compared to Figure6.18 (a) we find that the performance has changed from 0.073 to 0.06 which is a good compared to the desired goal. R1 has changed from 0.75 to 0.78, and R2 is approximately unchanged. So, adding the previous example to the training set, and re-training it improves the performance; see Figure6.17 (b, c) and Figure6.18 (b, c).

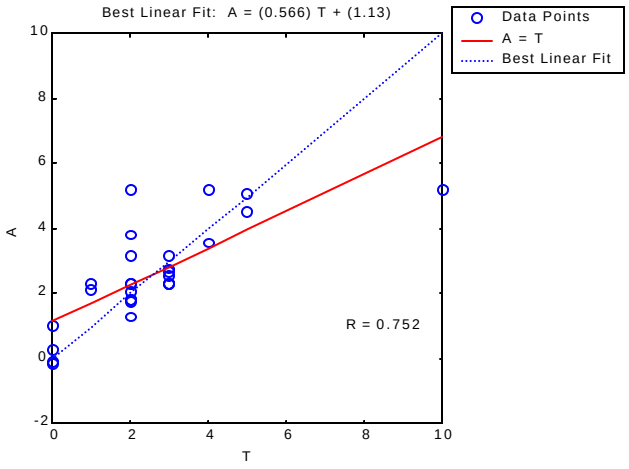


Figure6.17 (b) Regression Analysis for the First Target Value of NN#2

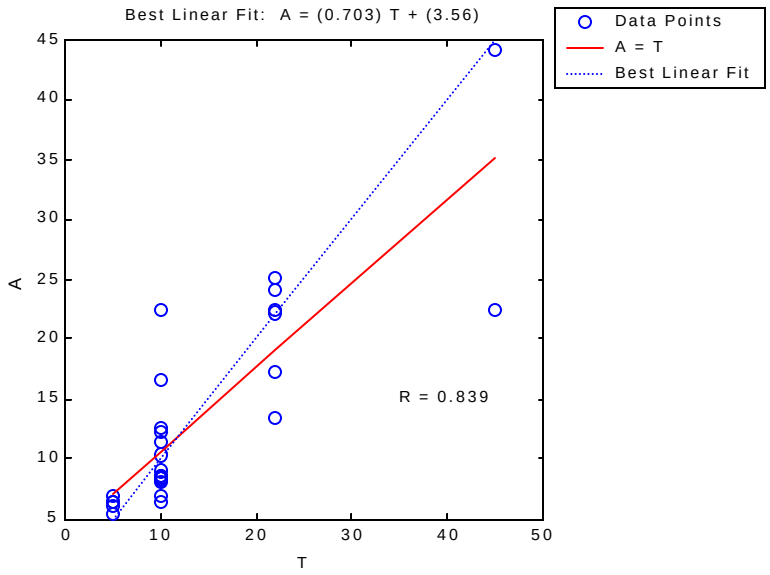


Figure6.17 (c) Regression Analysis for the Second Target Value of NN#2

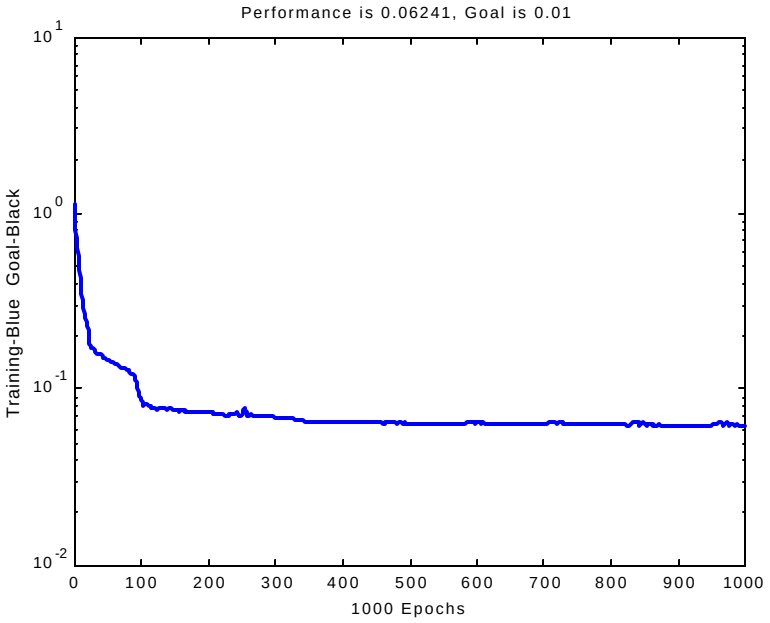


Figure6.18 (a) NN#2 Performance Analysis After Adding the New Training Example

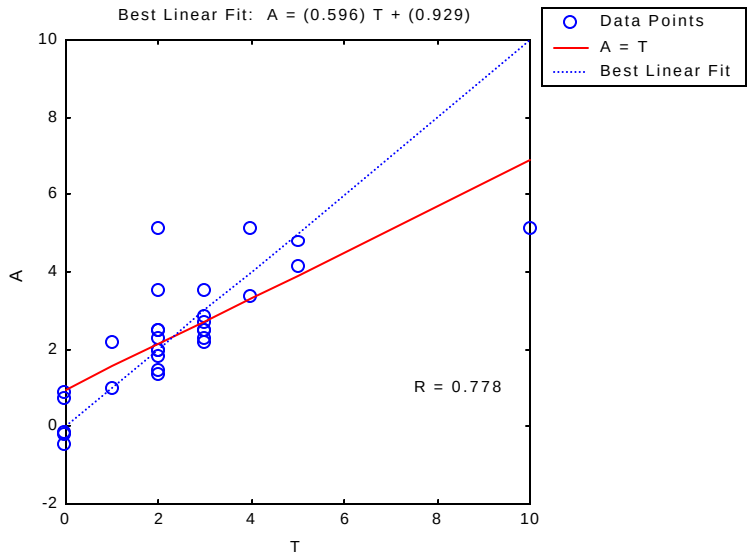


Figure6.18 (b) Regression Analysis for the First Target Value of NN#2

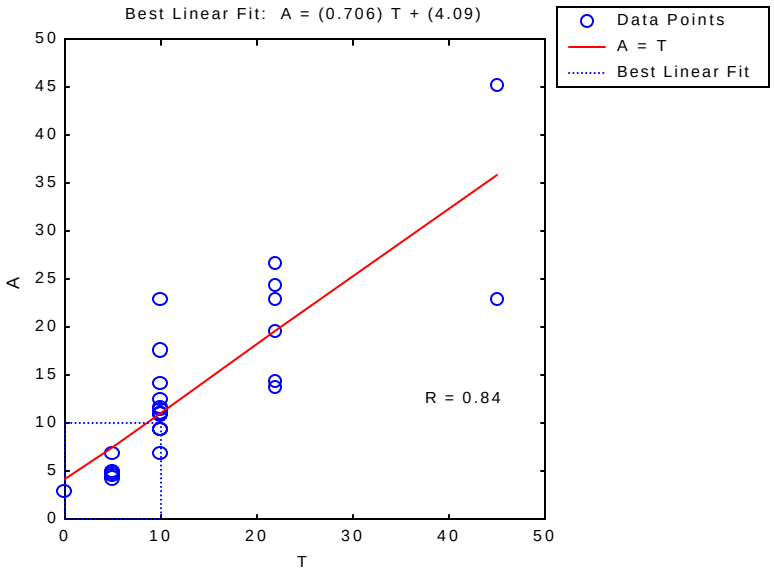


Figure6.18 (c) Regression Analysis for the Second Target Value of NN#2

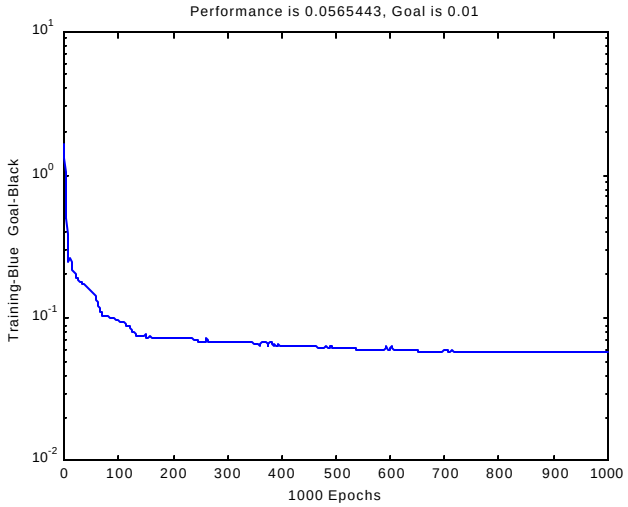


Figure6.19 (a) NN#3 Performance Analysis Before Adding the New Training Example

Figure6.19 (a) represents the performance of NN#3 before the new example is fed into the training set. Compared to Figure6.20 (a) we find that the performance has changed from 0.057 to 0.06, which approximately has the same value. R1 has changed from 0.92 to 0.87, and R2 is approximately unchanged. So, adding the previous example to the training set, and re-training it did not improve the performance; see Figure6.19 (b, c) and Figure6.20 (b, c).

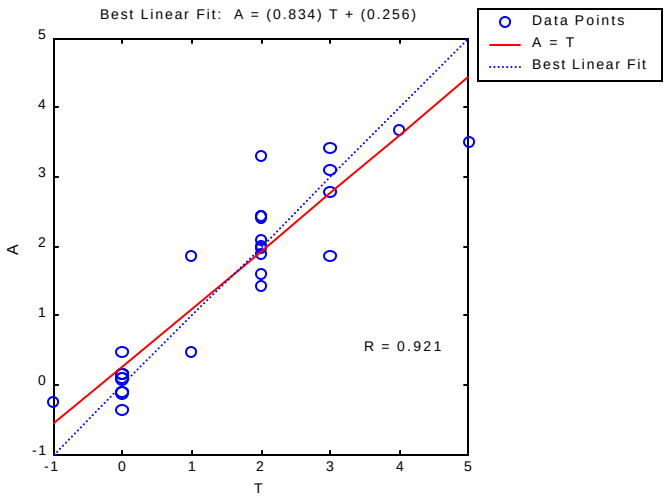


Figure6.19 (b) Regression Analysis for the First Target Value of NN#3

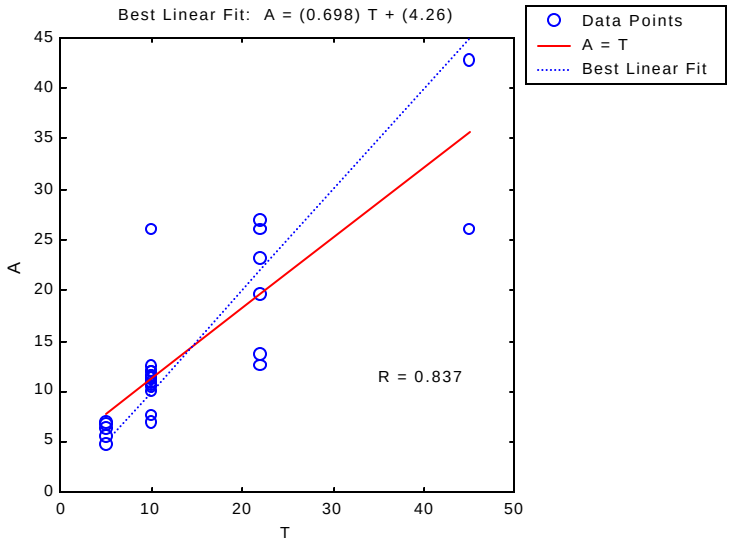


Figure6.19 (c) Regression Analysis for the Second Target Value of NN#3

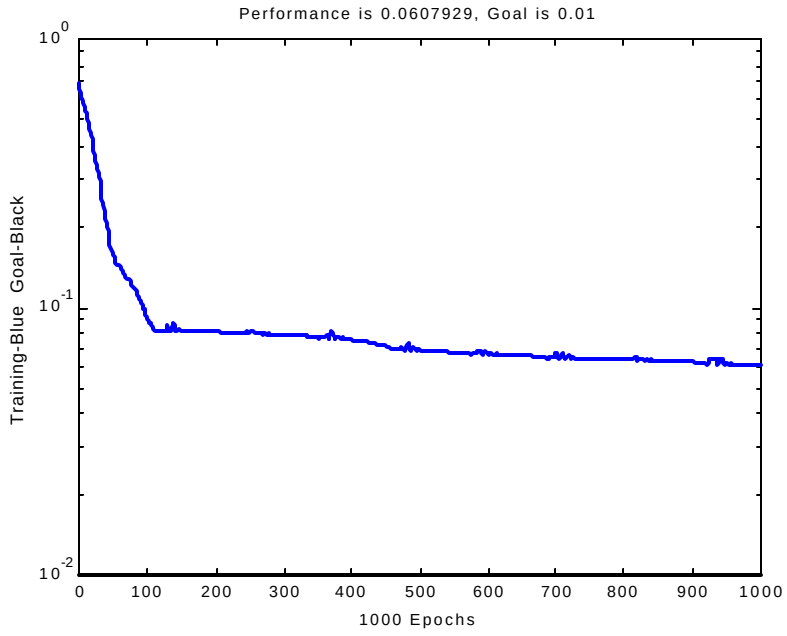


Figure6.20 (a) NN#3 Performance Analysis After Adding the New Training Example

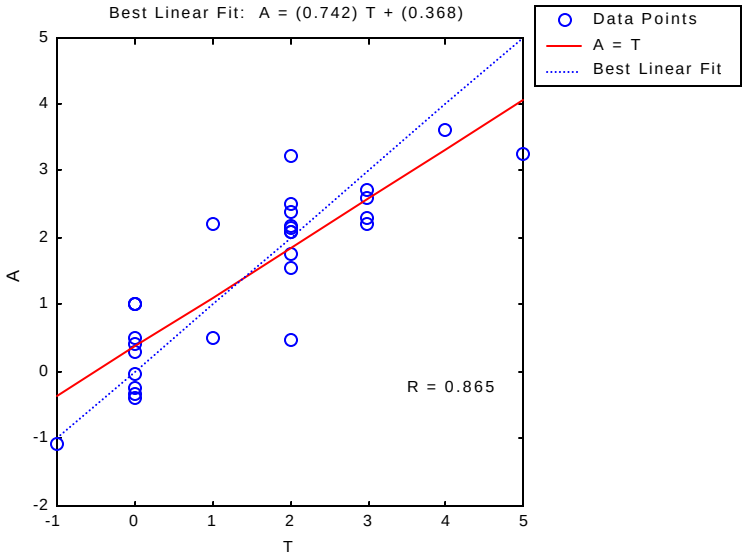


Figure6.20 (b) Regression Analysis for the First Target Value of NN#3

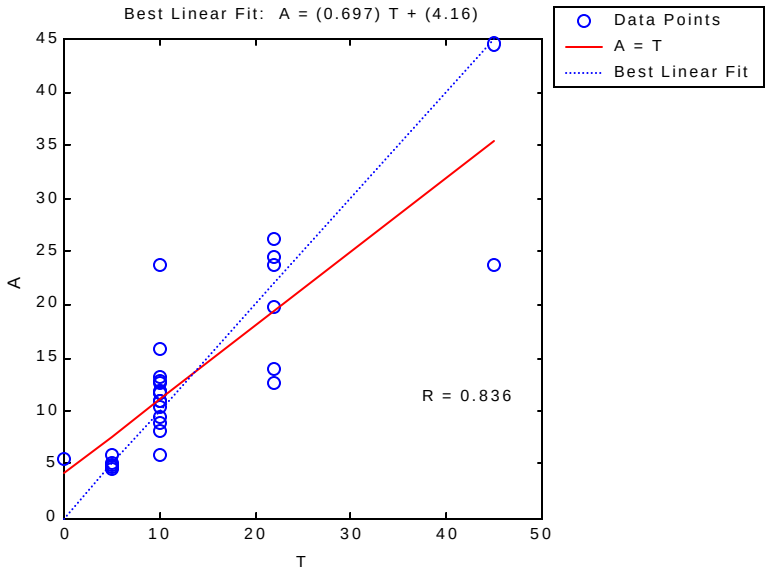


Figure6.20 (c) Regression Analysis for the Second Target Value of NN#3

4. Testing the Generality Feature

Another Example Not Included in the Training Set
Target Point at (50,50,100)

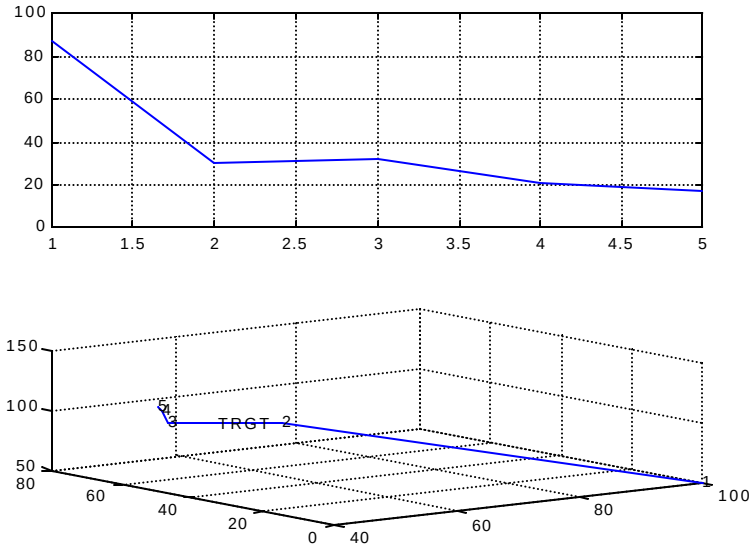


Figure6.21 (a) End-Effector Distance from the Initial Point to the Target Point

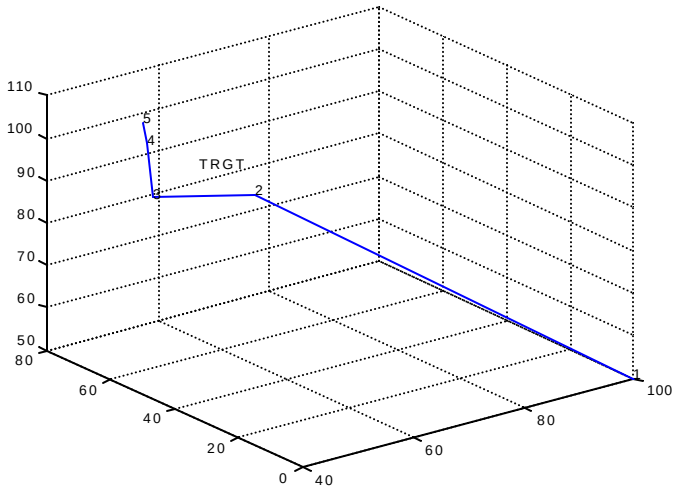


Figure6.21 (b) End-Effector Motion

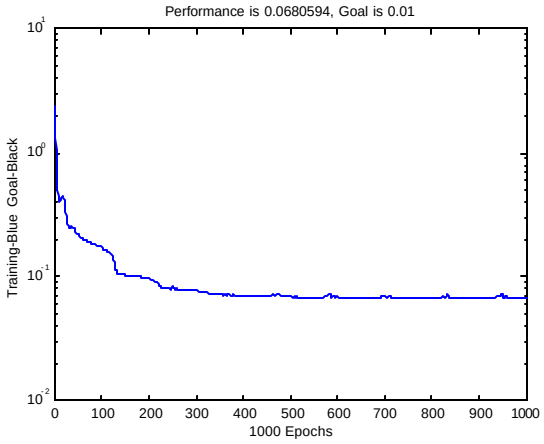


Figure6.22 (a) NN#1 Performance Analysis Before Adding the New Training Example

Using the AIM gives a jump-start from an approximate distance of 85 to 30 from the target point. As we see in Figure6.21 (a,b) the AIN took control again after step 2 of the NIN performance. Figure6.22 (a) represents the performance of NN#1 before the new example is fed into the training set. Compared to Figure6.23 (a) we find that the performance has changed from 0.068 to 0.081 which is a bad compared to the desired goal. R1 has changed from 0.87 to 0.83, and R2 changed from 0.85 to 0.83. So, adding the previous example to the training set, and re-training it did not improve the performance; see Figure6.22 (b, c) and Figure6.23 (b, c).

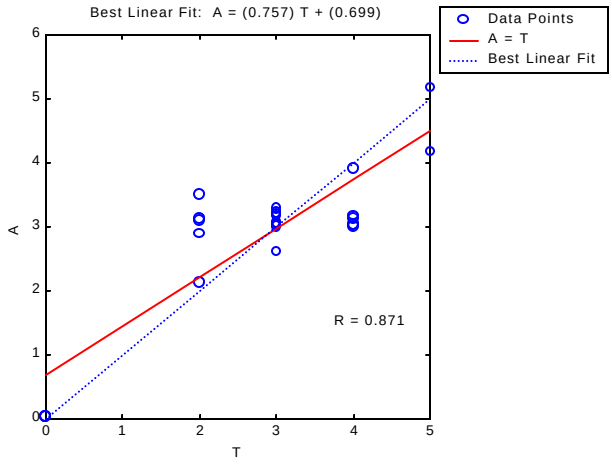


Figure6.22 (b) Regression Analysis for the First Target Value of NN#1

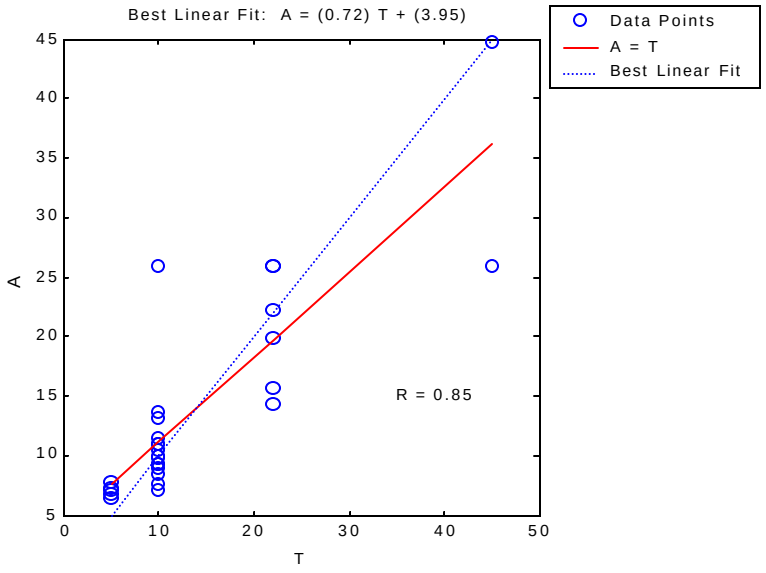


Figure6.22 (c) Regression Analysis for the Second Target Value of NN#1

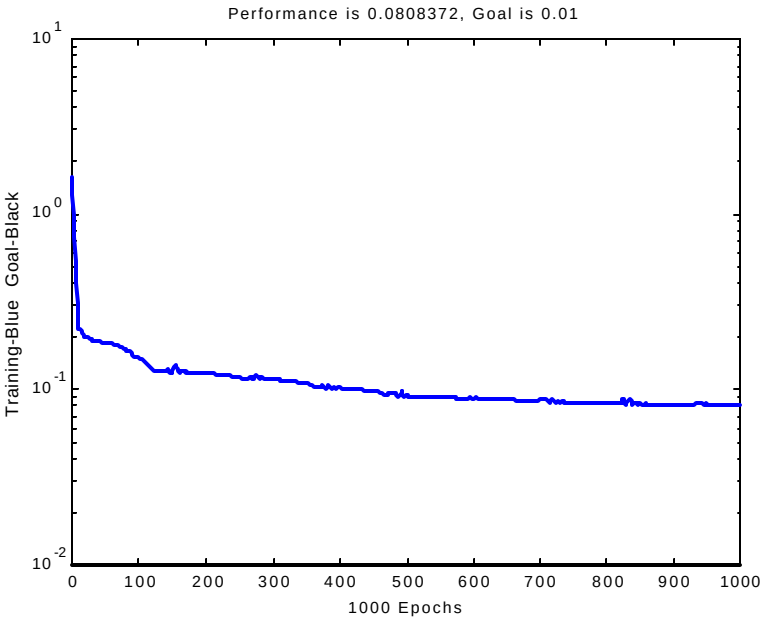


Figure6.23 (a) NN#1 Performance Analysis After Adding the New Training Example

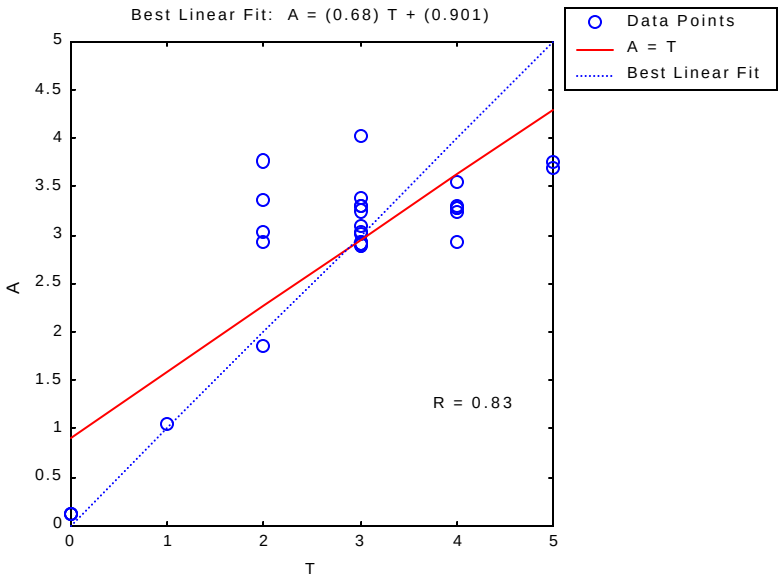


Figure6.23 (b) Regression Analysis for the First Target Value of NN#1

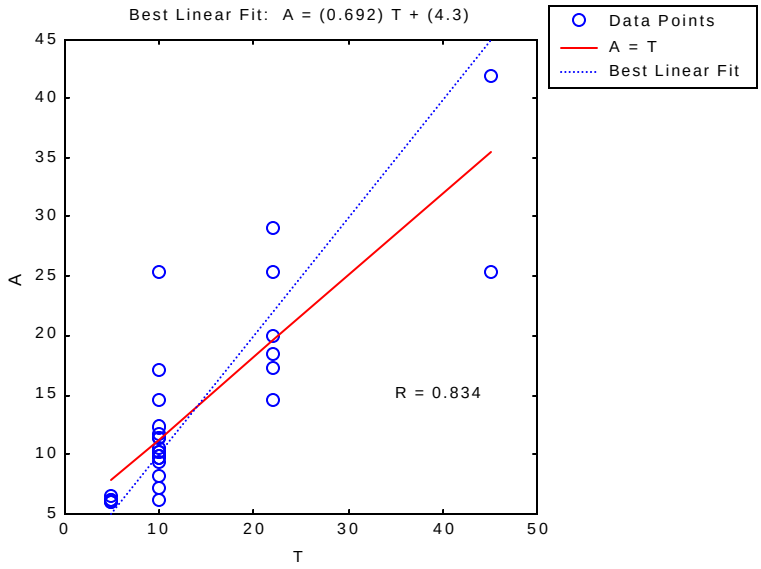


Figure6.23 (c) Regression Analysis for the Second Target Value of NN#1

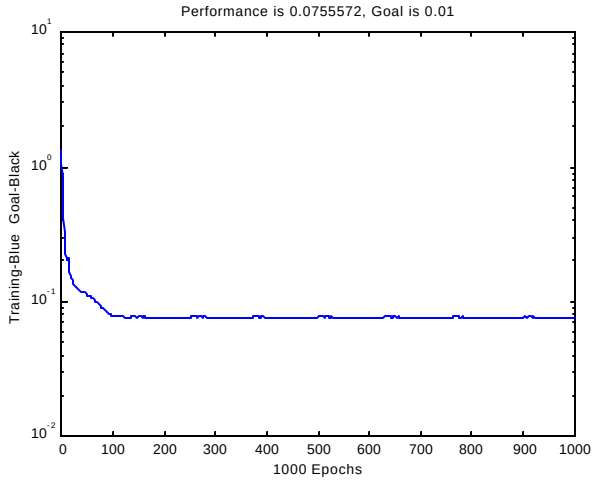


Figure6.24 (a) NN#2 Performance Analysis Before Adding the New Training Example

Figure6.24 (a) represents the performance of NN#3 before the new example is fed into the training set. Compared to Figure6.25 (a) we find that the performance is approximately unchanged which is a good progress toward the desired goal. R1 has changed from 0.73 to 0.7, and R2 changed is approximately unchanged. So, adding the previous example to the training set, and re-training it slightly improves the performance; see Figure6.24 (b, c) and Figure6.25 (b, c).

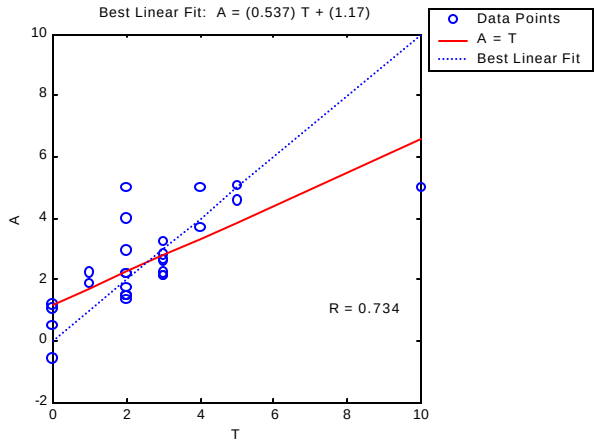


Figure6.24 (b) Regression Analysis for the First Target Value of NN#2

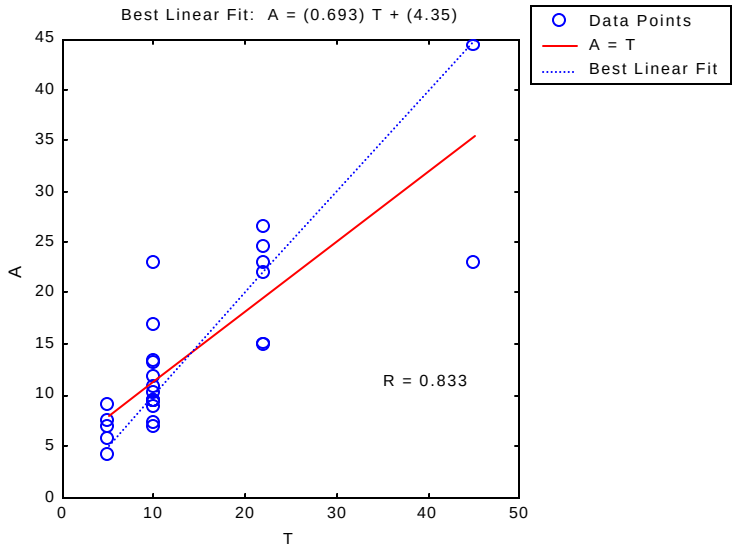


Figure6.24 (c) Regression Analysis for the Second Target Value of NN#2

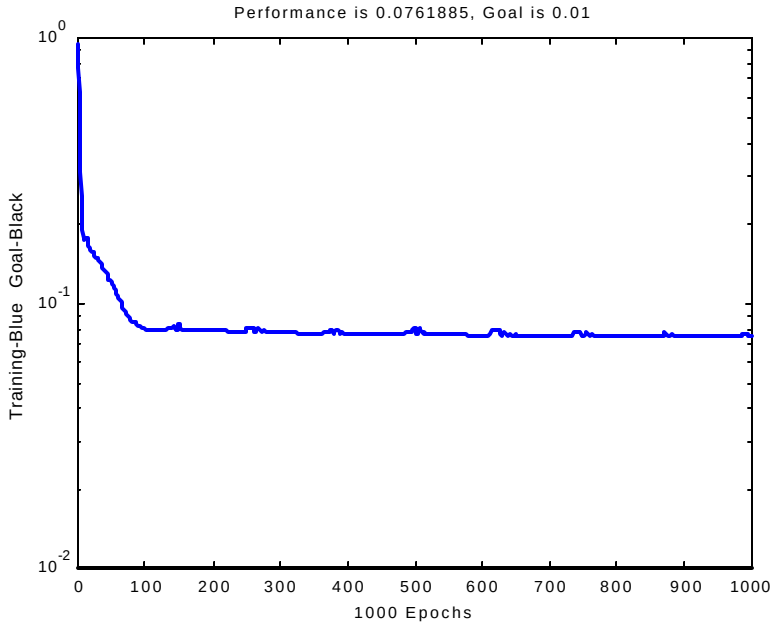


Figure6.25 (a) NN#2 Performance Analysis After Adding the New Training Example

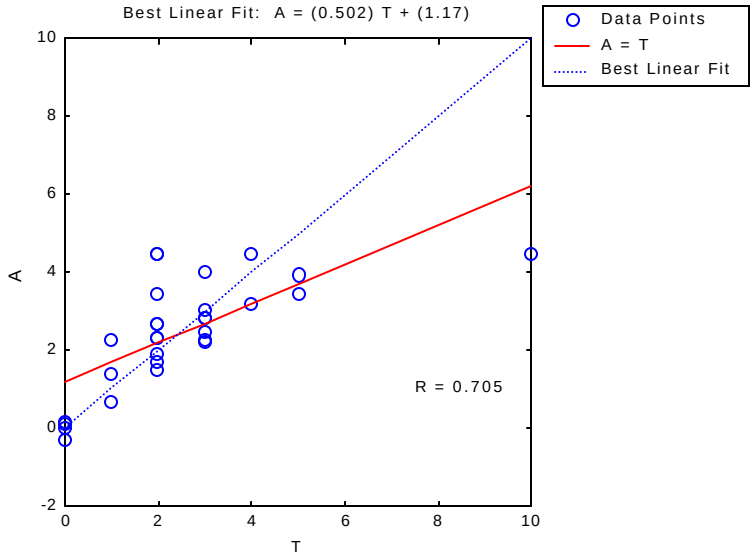


Figure6.25 (b) Regression Analysis for the First Target Value of NN#2

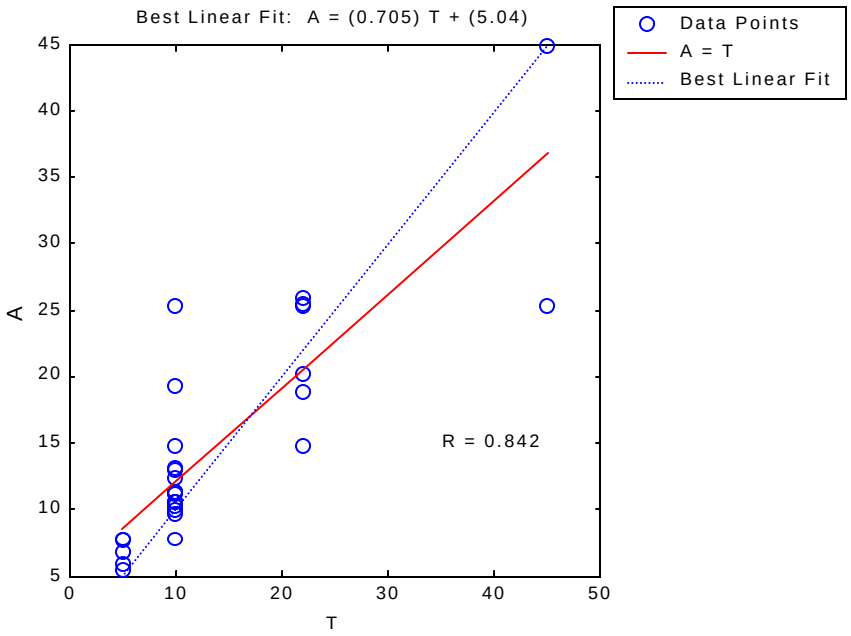


Figure6.25 (c) Regression Analysis for the Second Target Value of NN#2

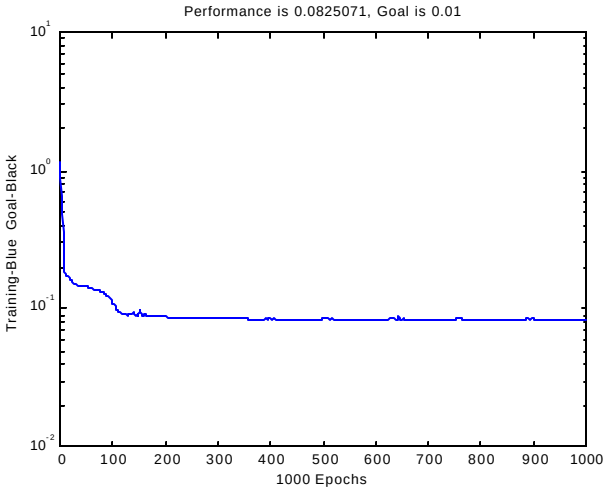


Figure6.26 (a) NN#3 Performance Analysis Before Adding the New Training Example

Figure6.26 (a) represents the performance of NN#3 before the new example is fed into the training set. Compared to Figure6.27 (a) we find that the performance has changed from 0.083 to 0.067, which is good compared to the desired goal. R1 has changed from 0.8 to 0.87, and R2 changed is approximately unchanged. So, adding the previous example to the training set, and re-training it improves the performance; see Figure6.26 (b, c) and Figure6.27 (b, c).

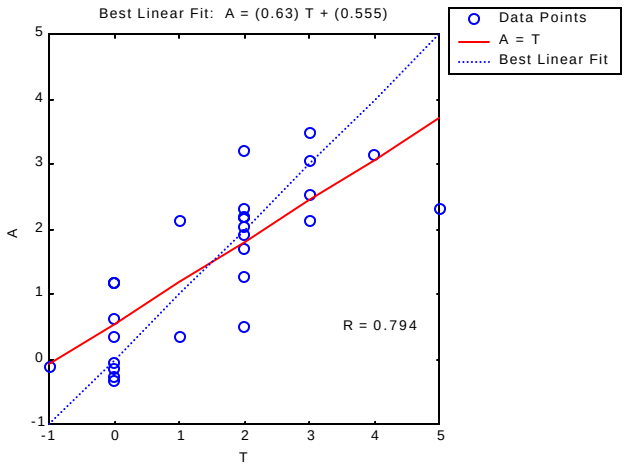


Figure6.26 (b) Regression Analysis for the First Target Value of NN#3

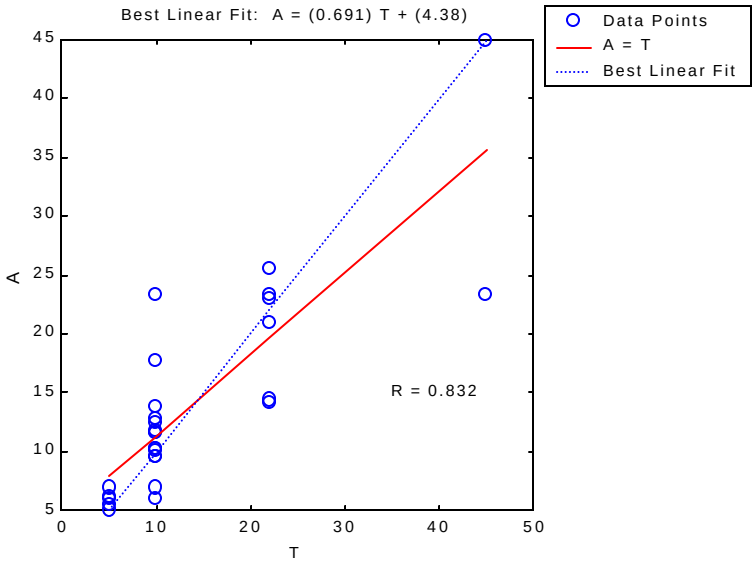


Figure6.26 (c) Regression Analysis for the Second Target Value of NN#3

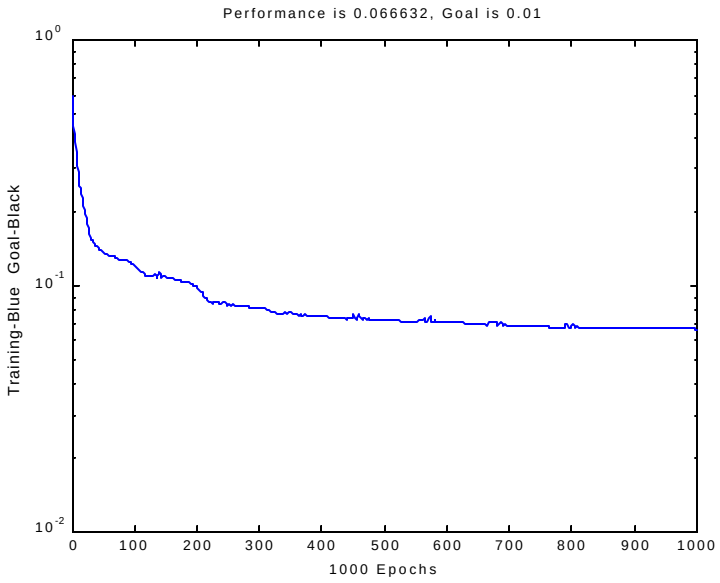


Figure6.27 (a) NN#3 Performance Analysis After Adding the New Training Example

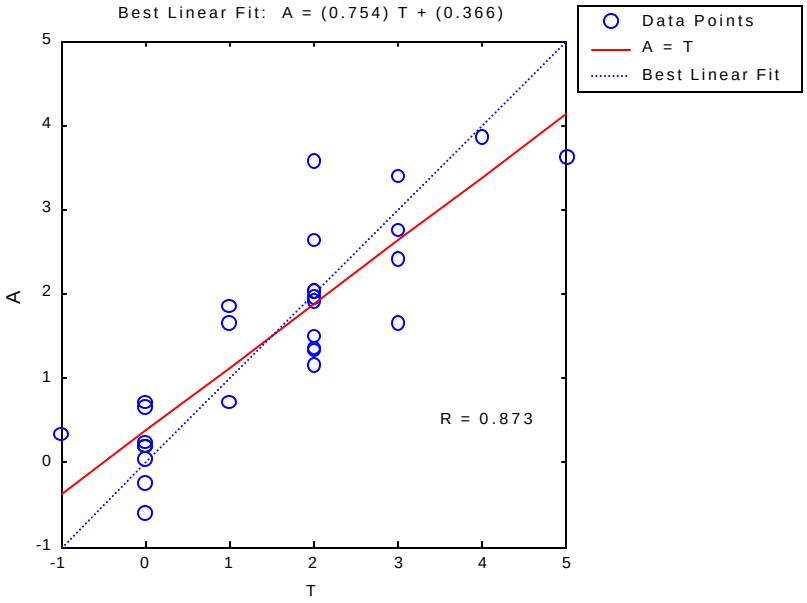


Figure6.27 (b) Regression Analysis for the First Target Value of NN#3

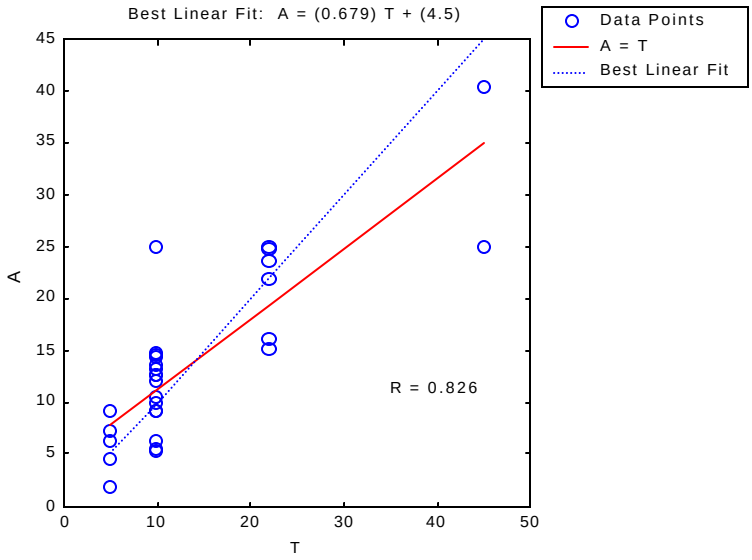


Figure6.27 (c) Regression Analysis for the Second Target Value of NN#3

5. Using Incremental Training

In the previous examples we applied different input cases to test the performance of the NIN. We notice that adding a single example to the AIM knowledge base does not always guarantee performance enhancement. We have to have enough number of examples to cover most of the new cases that might be encountered in the simulation process. The following example uses incremental training to increase the number of learning cases. In all of the previous examples, we used the new example to retrain the AIM Network without changing the original training set. A new algorithm is designed such that the cases resulting from the AIN learning modifies both the training set and the weight matrix. The algorithm stops the automatic generation of training examples, and also the overall training process when the desired goal is reached. We use the following rule to control the incremental processing:

$$\sum R_{ij} = g \ , \ \begin{matrix} i = 1,...,3 \\ j = 1,2 \end{matrix}$$

The optimal performance is achieved when g=6, i.e. $\sum R_{ij} = 6$. In the example, we set the value of g to 5.4 such that there is an average R-value of 0.9.

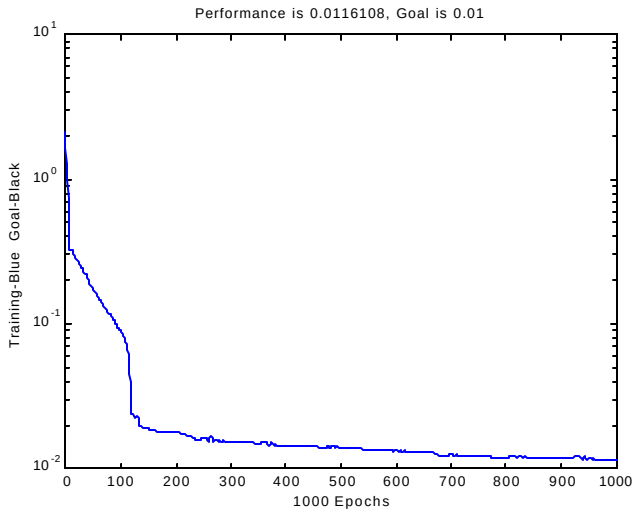


Figure6.28 (a) NN#1 Performance Analysis After Adding the New Training Examples

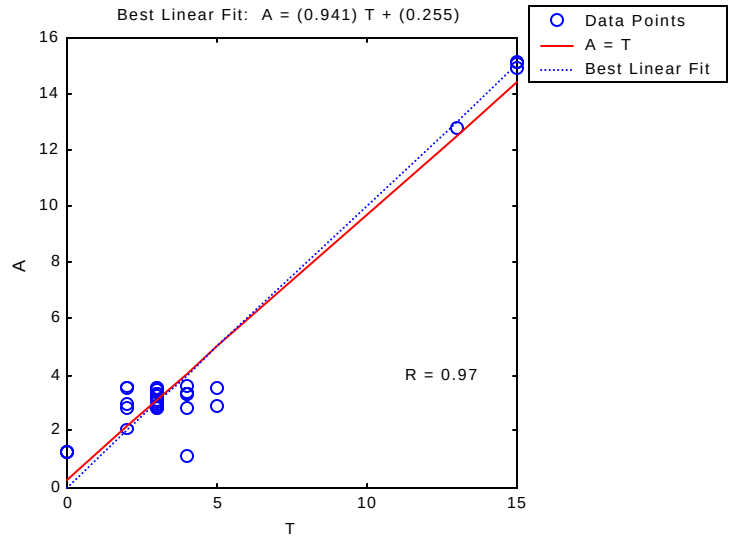


Figure6.28 (b) Regression Analysis for the First Target Value of NN#1

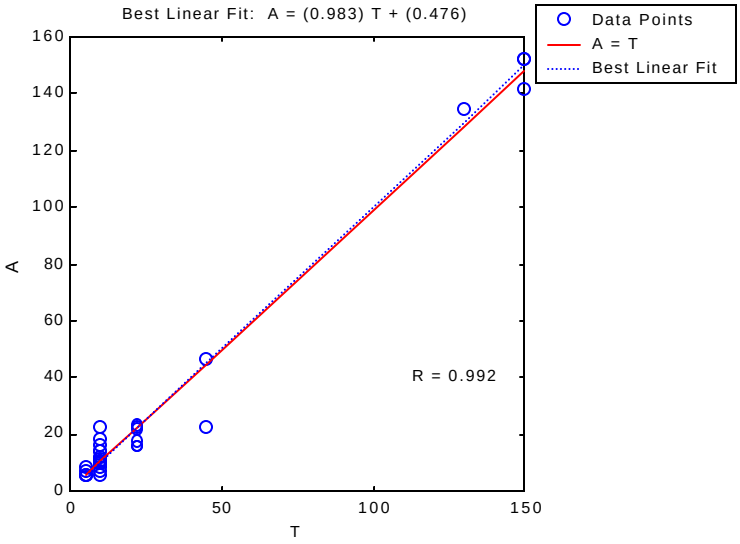


Figure6.28 (c) Regression Analysis for the Second Target Value of NN#1

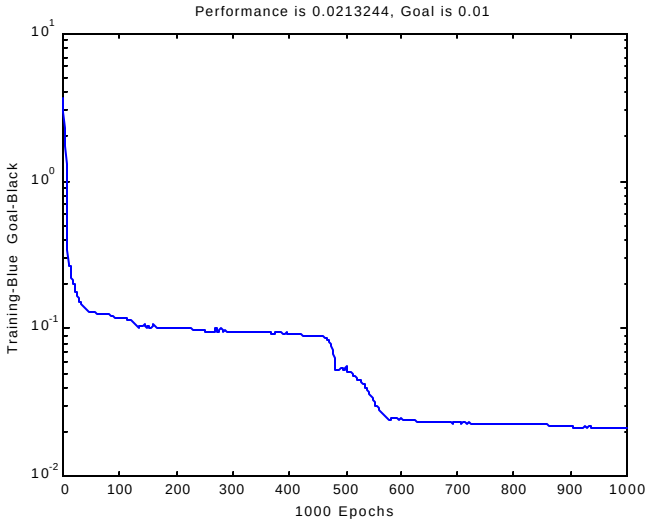


Figure6.29 (a) NN#2 Performance Analysis After Adding the New Training Examples

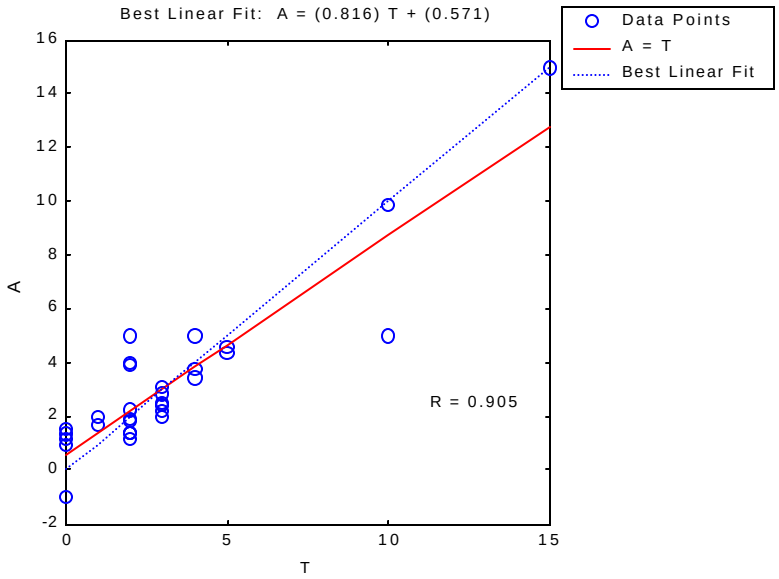


Figure6.29 (b) Regression Analysis for the First Target Value of NN#2

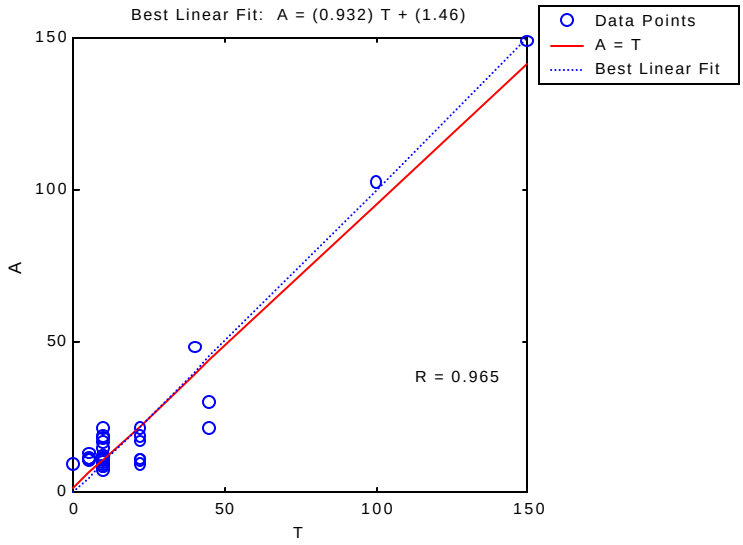


Figure6.29 (c) Regression Analysis for the Second Target Value of NN#2

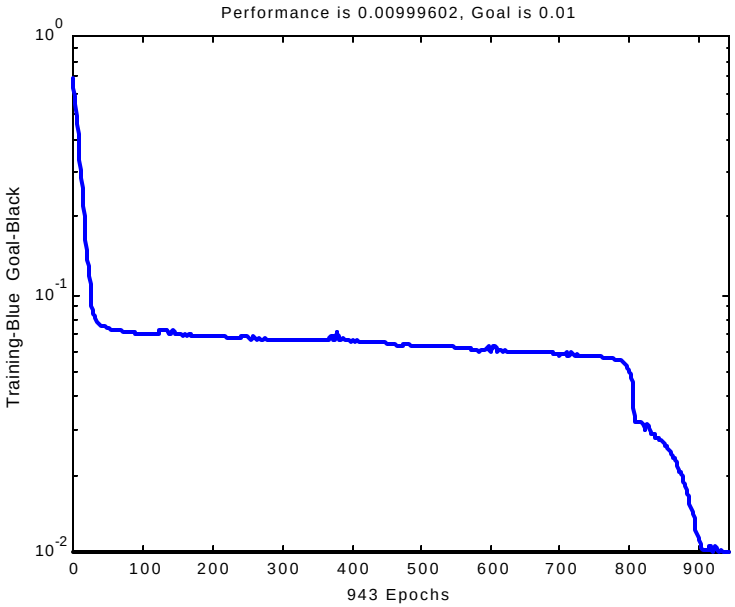


Figure6.30 (a) NN#3 Performance Analysis After Adding the New Training Examples

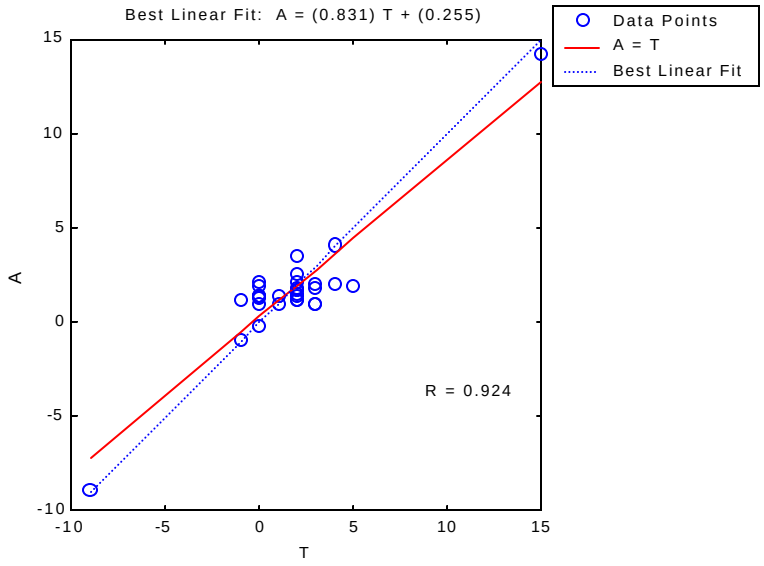


Figure6.30 (b) Regression Analysis for the First Target Value of NN#3

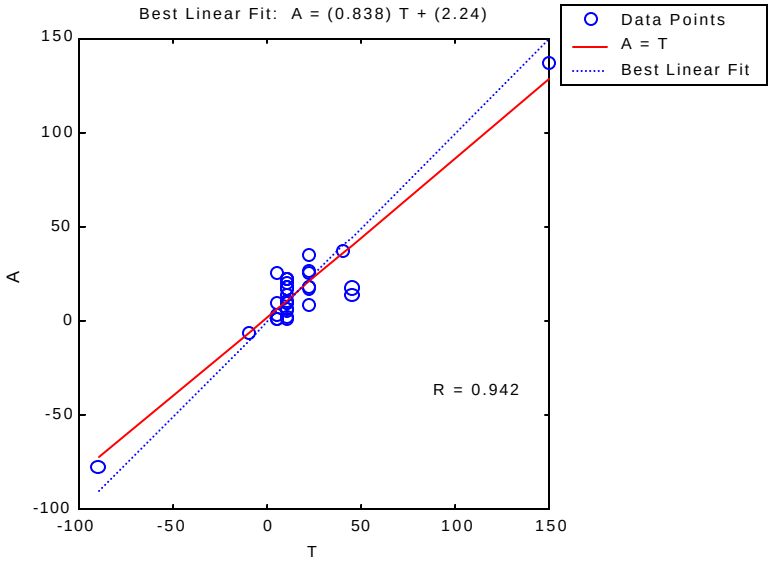


Figure6.30 (c) Regression Analysis for the Second Target Value of NN#3

Figure6.28 shows a remarkable decrease in NN#1 error level from 0.108 to 0.011, an increase in R1-value from 0.68 to 0.97, and an increase in R2-value from 0.85 to 0.99. Figure29 shows a decrease in NN#2 error level from 0.076 to 0.021, an increase in R1-value from 0.7 to 0.91, and an increase in R2-value from 0.85 to 0.97. Figure30 shows a decrease in NN#3 error level from 0.068 to 0.001, which is the desired error level (convergence met). R1-value for NN#3 increase from 0.86 to 0.92, and R2-value increase from 0.84 to 0.94 and as a result NN#3 has the best performance over the other nets in the NIN.

After running the previous simulation example, four training input examples have been added to the training set. The incremental training algorithm not only enlarges the training set, but also improves the overall efficiency of the NIN. So, by increasing the value of g to approach six, we can improve the performance of the NIN, and still have new automatically generated cases as well.

New Additional input ±Dx, ±Dy, and ±Dz values

-101.2535	-99.5634	-107.4818	-60.3178
-0.9297	99.9379	19.4384	57.8187
-21.0686	-53.3347	34.7490	-23.9679

New Additional desired output values for NN#1

15	15	13	15
150	150	130	150

New Additional desired output values for NN#2

4	0	10	15
40	0	100	150

New Additional desired output values for NN#3

4	-9	-1	15
40	-90	-10	150