Chapter 4

Using the Immune Network to Control a Robot Arm

Most Industrial tasks, such as assembly, and packaging, rely on the manipulation and transportation of small, or large components. Robot arm manipulators can automate many of these processes and improve the cost efficiency of the operation. In order to control a robot, it must be kinematically analyzed and the result of this analysis embedded into the controller of the robot. Kinematics analysis starts with the determination of the position of each joint given the configuration of the robot, the base position and orientation, the angles of the revolute joints, and the length of the links of the robot; this is called the *forward kinematics solution* (FKS) of the robot. The FKS is always obtainable in a finite number of operations and can be determined in a closed form. The FKS is a prerequisite for obtaining the inverse kinematics solution, which is entered into the robot controller and forms the basis of the remote control of the robot. To study the reverse problem, namely, 'Given a position in the space of the robot, can the joint space be determined?' is known as obtaining the *inverse kinematics solution* (IKS) [40], [41], and [42].

1. Forward Kinematics Solution (FKS)

Figure 4.1 shows a 3 DOF manipulator, trying to place a ball in a can. In order to represent a relative kinematics relationship precisely between two adjacent links, we follow the *Denavit-Hartenberg* (D-H) notation. The relationship between the two links is described by the relative position and the orientation of the two coordinated frames attached to the two links. Figure 4.2 shows the frame assignment for each manipulator joint according to the D-H criteria [43].

The relative position of the two frames can be completely determined by the following four parameters:

a_i The length of the common normal, equal to the shortest distance between the Z_i-1 axis and the Z_i axis.

d_i The offset, the distance from the origin to the i-1 coordinate frame to the intersection point of the Z_i-1 axis and the X_i axis measured along the Z_i-1 axis.

μ_i The twist, the angle between the Z_{i-1} axis and the Z_i axis about the X_i axis in the right hind sense.

q_i The angle between the Xi-1 axis and the Xi about the Z_{i-1} axis the right hand sense.

Using the above four parameters, we can proceed to the next step, which is representing the homogeneous transformation matrix H from frame i to frame i-1. Table-1 represents the D-H table that contains the parameters needed to calculate the FKS, which is presented by T^{i}_{i-1}.

$$H = \begin{bmatrix} R^{n}_{0} & d^{n}_{0} \\ 0 & 1 \end{bmatrix} = T^{n}_{0}$$

$$T^{i}_{i-1} = Rot(Z,\grave{e}_i)Trans(0,0,d_i)Trans(a_i,0,0)Rot(x,\acute{a}_i)$$

Table 4.1 The D-H Table

Link Variable		q	a	a	d
1	q_1	q_1	90°	a_1	d_1
2	q_2	q_2	0°	a_2	d_2
3	q_3	q_3	90°	0	0

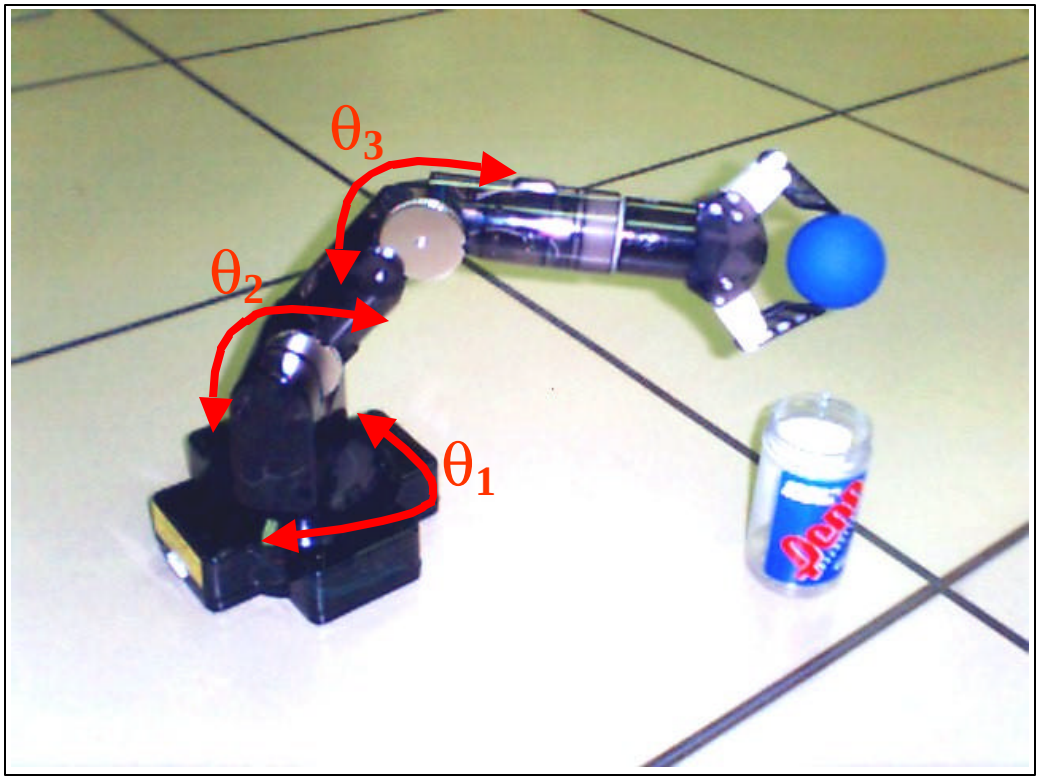


Figure4.1 Robotic Arm Manipulator

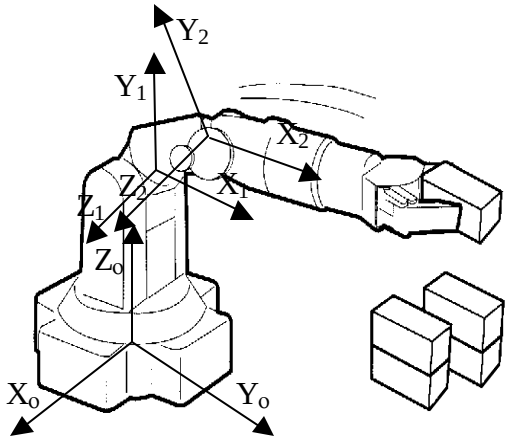


Figure4.2 Frame Assignment of the Manipulator

$$T_0^3 = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & c_1c_{23}l_3 + c_1c_2l_2 \\ s_1c_{23} & -s_1s_{23} & -c_1 & s_1c_{23}l_3 + s_1c_2l_2 \\ s_{23} & c_{23} & 0 & s_{23}l_3 + s_2l_2 + l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

By obtaining the homogeneous transformation matrix T_0^3, we are ready to calculate the possible inverse kinematics solutions where $C1$ is $\cos\theta_1$, S_1 is $\sin\theta_1$, and $C23$ is $\cos(\theta_2+\theta_3)$. Generally, in the case of IKS, all H matrix values are given, and the problem statement is that: Given a point in the robot state space, find the possible values for joint angles to reach this desired point. The problem is that there maybe more than one way to solve the IKS problem. There might be more than one value for angles that will be used to reach the desired target position. Also, there are other considerations that might need attention in case of a constrained environment.

2. The Proposed Artificial Immune Network

We designed an artificial immune network based upon the previous discussion of Jerne's biological approach. Each arm has a state that is considered to be a B-cell and the behavior of each link is considered to be the antibody. Catching an object by the manipulator end-effector, implicitly going to a certain XYZ location in the workspace of the manipulator is considered as attacking the antigen to suppress its behavior. The control parameter is chosen to be the distance between the end-effector position, and the desired target object position. This control parameter is considered to be a T-cell that stimulates B-cells and their antibodies to interact mutually in order to overcome the antigen behavior. The structure of our designed immune network is basically three cells corresponding to three joints; each cell has two antibodies attached to it. Each antibody is fully connected to all the other B-cell antibodies to allow stimulation and suppression, see Figure4.3.

Each joint has local behavior (skill), and also global behavior (with other joints). The designed AIS network is designed in a way that uses the stimulation and suppression among joints in the network to reach the desired target location. So, when a joint takes the robot end-effector away from the desired location, its antibody concentration should be decreased (suppressed), and vise versa.

Joint's Local Behavior (skills):

1- Increase q_i (with a certain step) for Example ELBOW UP.
2- Decrease q_i (with a certain step) for Example ELBOW DOWN.

Joint's Global Behavior (with other joints):

1- Suppress other B-Cells Abs (behavior) Concentrations.
2- Stimulate other B-Cells Abs (behavior) Concentrations.

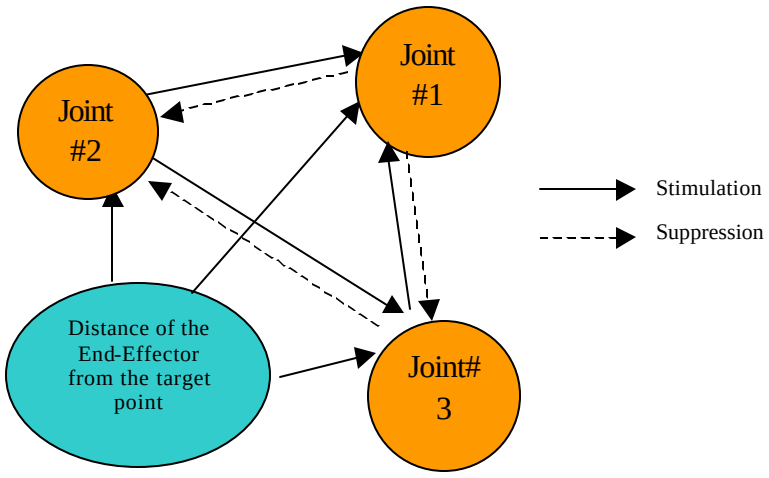


Figure4.3 The Proposed Immune Network

We developed equations for the concentrations of antibodies in each B-cell. Also, a progress factor was introduced to measure the validity and success of each antibody. Antibodies will be used whenever their concentrations exceed a certain threshold.

$$a_i = S_i \sum_{j}^{M} C_j \qquad (4.1)$$

$$P_i = (1 - S_i) \sum_{j=1}^{M} C_j \qquad (4.2)$$

$$C_{ik} = \sum_{j}^{M} a_j - \sum_{j}^{M} P_j \qquad (4.3)$$

$$C_{ik} = (2S_i - 1) \sum_{j=1}^{M} C_j \qquad (4.4)$$

Where $i = 1..N$, $j = 1..M$ and both N and M correspond to the total number of antibodies in each B-cell. K is an index number to differentiate between B-cell #1 and B-cell #2. S_i represents the success ratio for each antibody i, and it can be calculated as the ratio of the count of correct responses (actions) divided by the total number of responses taken by this antibody. The parameter C_{ik} represents the current concentration of antibody i in B-cell K, and it should exceed the concentration threshold for the antibody to respond to antigen stimulation.

The first term on the right hand side of Equation (4.3) represents the stimulation activation level spread by other antibodies in the herding immune network, and it can be calculated using Equation (4.1). The second term represents the suppression activation level spread by other antibodies in the same immune network, and it can be calculated using Equation (4.2). The third term represents the stimulation that an antibody receives from an antigen, and it controls the balance of the immune herding network. Introducing Equations (4.1), and (4.2) into Equation (4.3), we get Equation (4.4), which is the final form for calculating the concentration of an antibody.

3. Arm Manipulator Simulation

In the previous section we discussed the concentration equations that control the behavior of each link of the robot arm. In this section we inserted these equations into a Matlab program that simulates the overall behavior of the robot arm. We used the calculated Forward Kinematics Solution and the adaptive feature of the designed artificial immune network to control the bahavior of the manipulator. Figure4.4 illustrates the simple algorithm that is used in the simulation program.

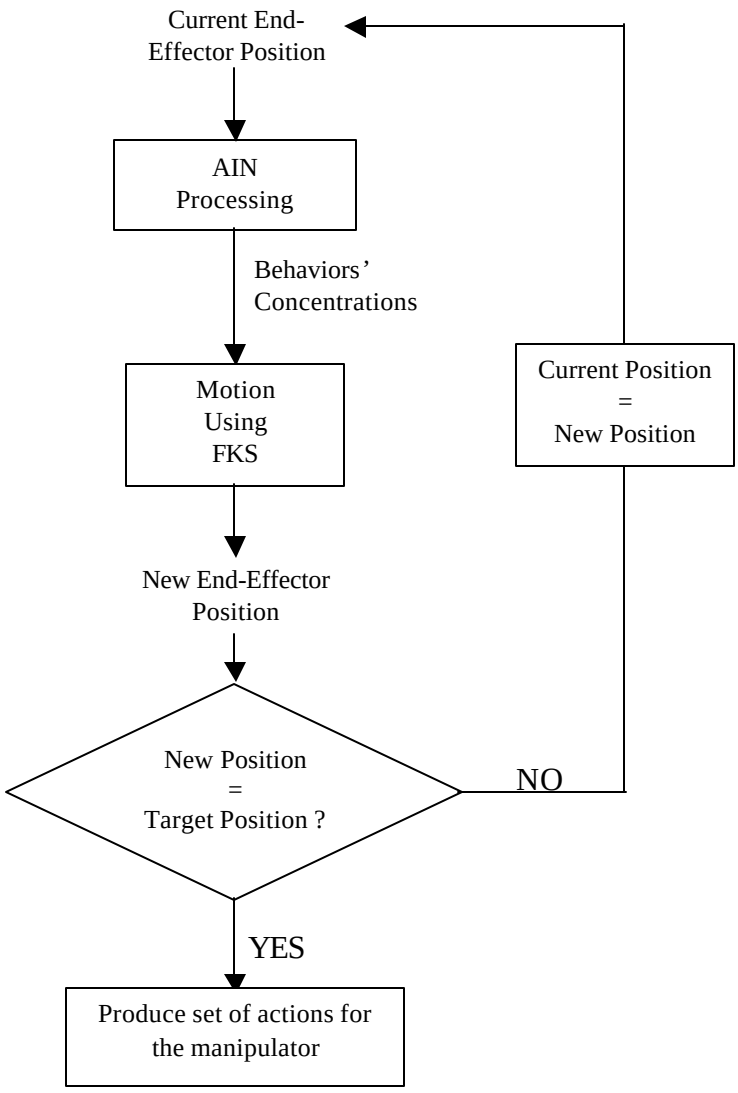


Figure4.4 Robot Arm Simulation Algorithm

We used this algorithm to analyze the performance of the Artificial Immune Network (AIN). Several tasks were studied to illustrate the learning ability of the AIN, and to investigate any modification needed for its design.

Task#1: it is required to control the motion of the robot arm while moving through the path from point (100 0 50) to (0 0 150). $\Delta\theta_i=\pi/18$, and $\xi<=15$.

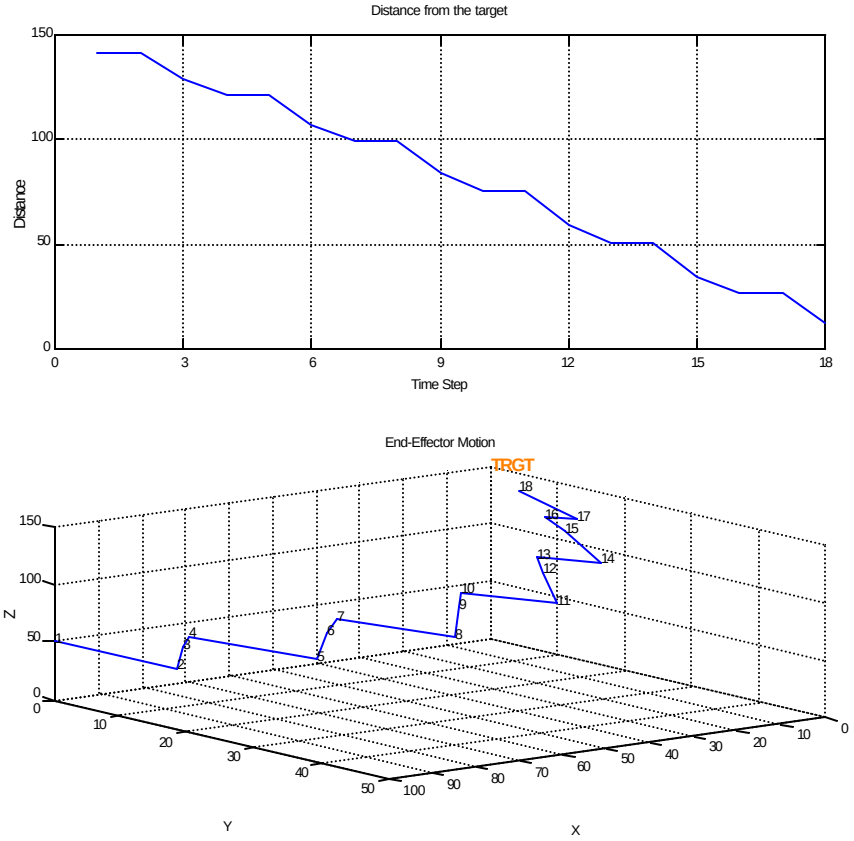


Figure4.5 Motion Analysis

Figure 4.5 summerizes the results of using the Atificial Immune Network in controling the motion of the robot arm. We can judge the performance of the immune network from motion annalysis dipicted in Figure4.5, which shows that the motion of the manipulator is in the direction toward the goal object. These two figure illuastrate that as the arm moves, the distance between the end-effector and the target point decreases. The figure gives 3D tracking of the end-effector motion in the robot working space.

If we take a look at the initial point, and the target point in the task, we will find that the arm is required to satisfy the following constraints as it moves:

- Reduce the X-Axis values
- Maintain the Y-Axis value
- Increase the Z-Axis value

If we invistigate plane projections, we will find that these requirement were fulfilled. In the XY plane projection, it was required to reduce the X-axis values to approach the goal. As we see, the End-Effector started moving away, and then appraoched the target point. The same bahavior was repeated in the ZY plane, the value of the Y-Axis was restored to reach the goal, see Figure4.6. For the Z-Axis, we can see a monotonic increase in the right direction from the beginning approaching the target point, see Figure4.7. A combined XYZ index value tracking is given in Figure4.9 to give in further in the simulation process.

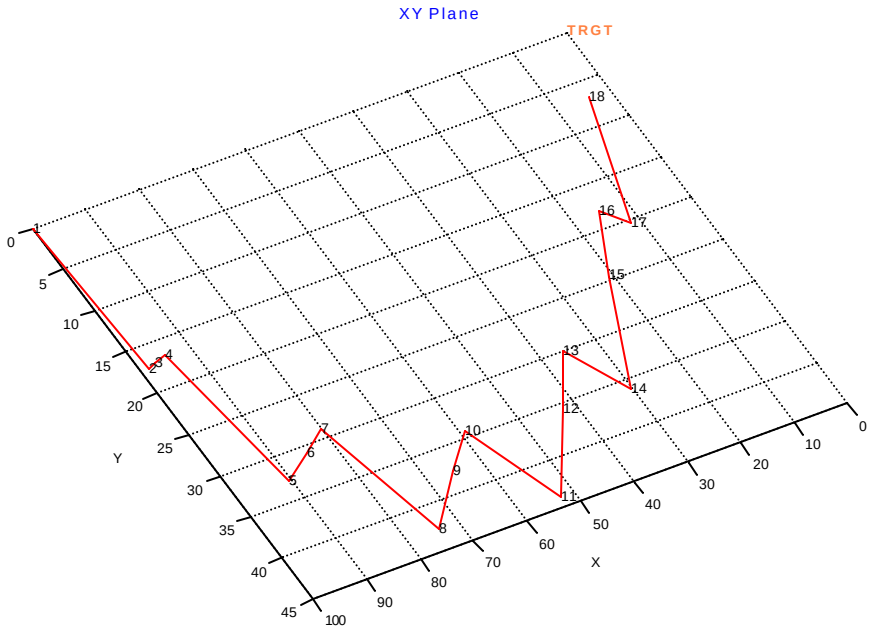


Figure4.6 XY plane projection

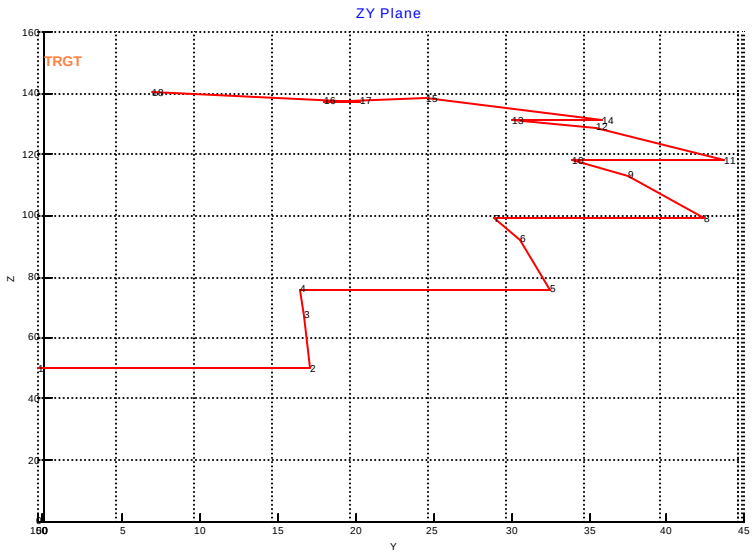


Figure4.7 ZY Projection

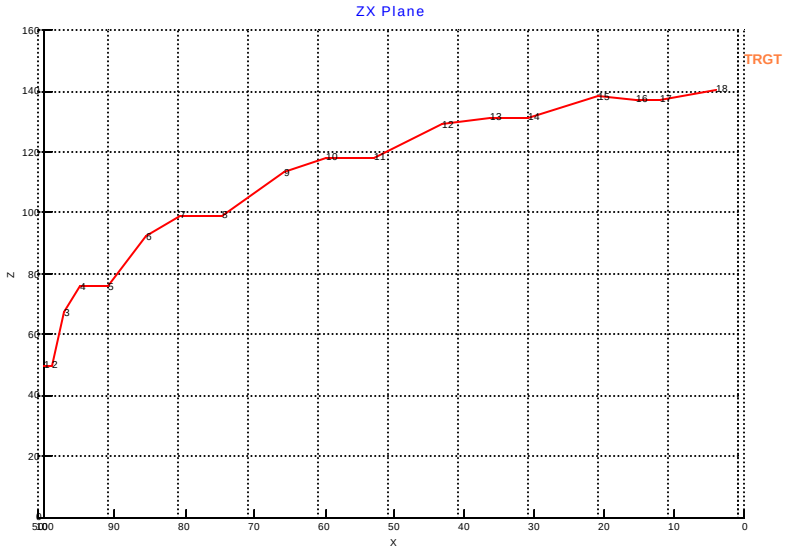


Figure4.8 ZX Projection

Figure4.10 shows the behavior concentration for each link of the manipulator. We notice that all the concentration of each behavior is increasing as the simulation progresses. This is expected since the distance from the target position is decreasing all the time.

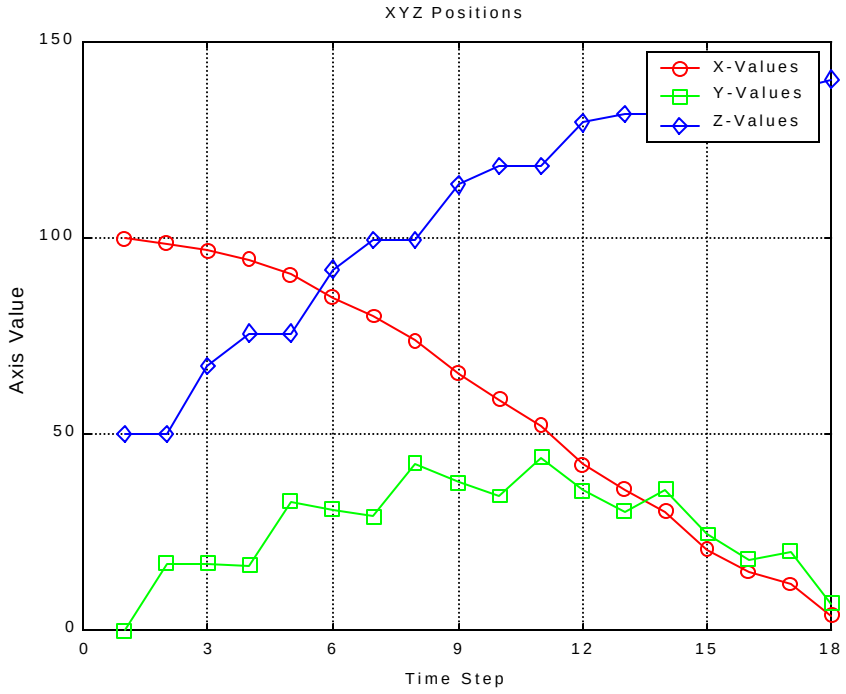


Figure4.9 XYZ Values

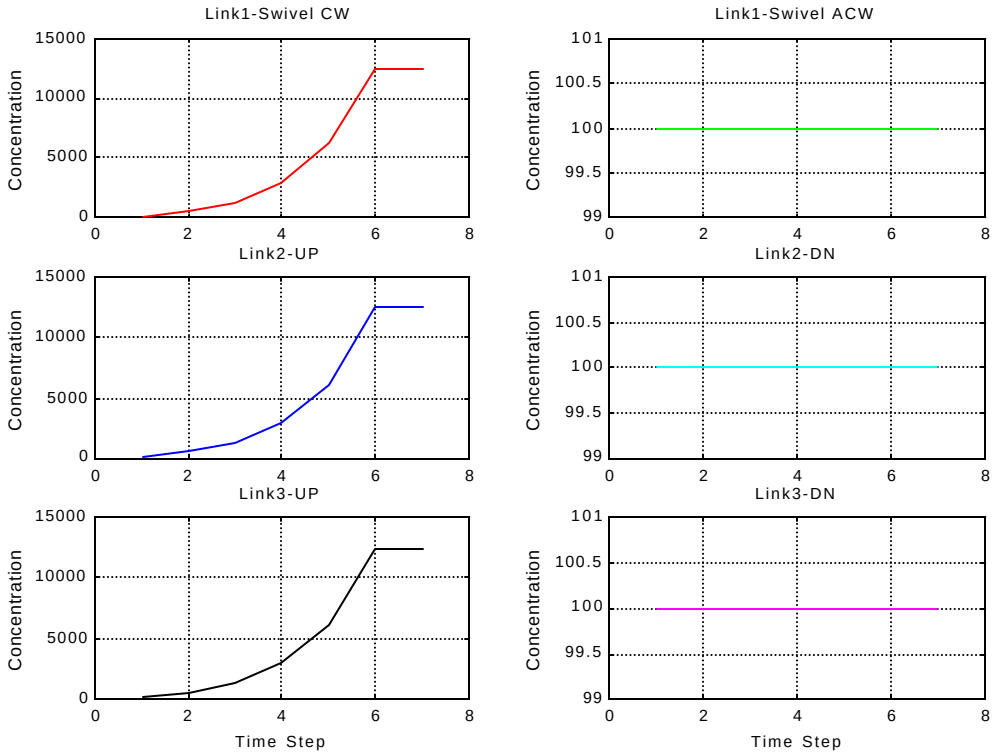


Figure4.10 Behavior Concentrations

4. More Training Cases for the AIN

Task#2: it is required to control the motion of the robot arm while moving through the path from point (100 0 50) to (0 0 50). $\Delta\theta_i=\pi/2$, and $\xi<=15$.

The following figures summerizes the results of using the Atificial Immune Network in controling the motion of the robot arm. We can judge the performance of the immune network from motion analysis dipicted in Figure4.11, which shows the motion of the manipulator was in the right direction. These two figures illustrate that as the arm moves, the distance between the end-effector and the target point decreases. This time, we used a larger step, and the target was reached after 7 steps.

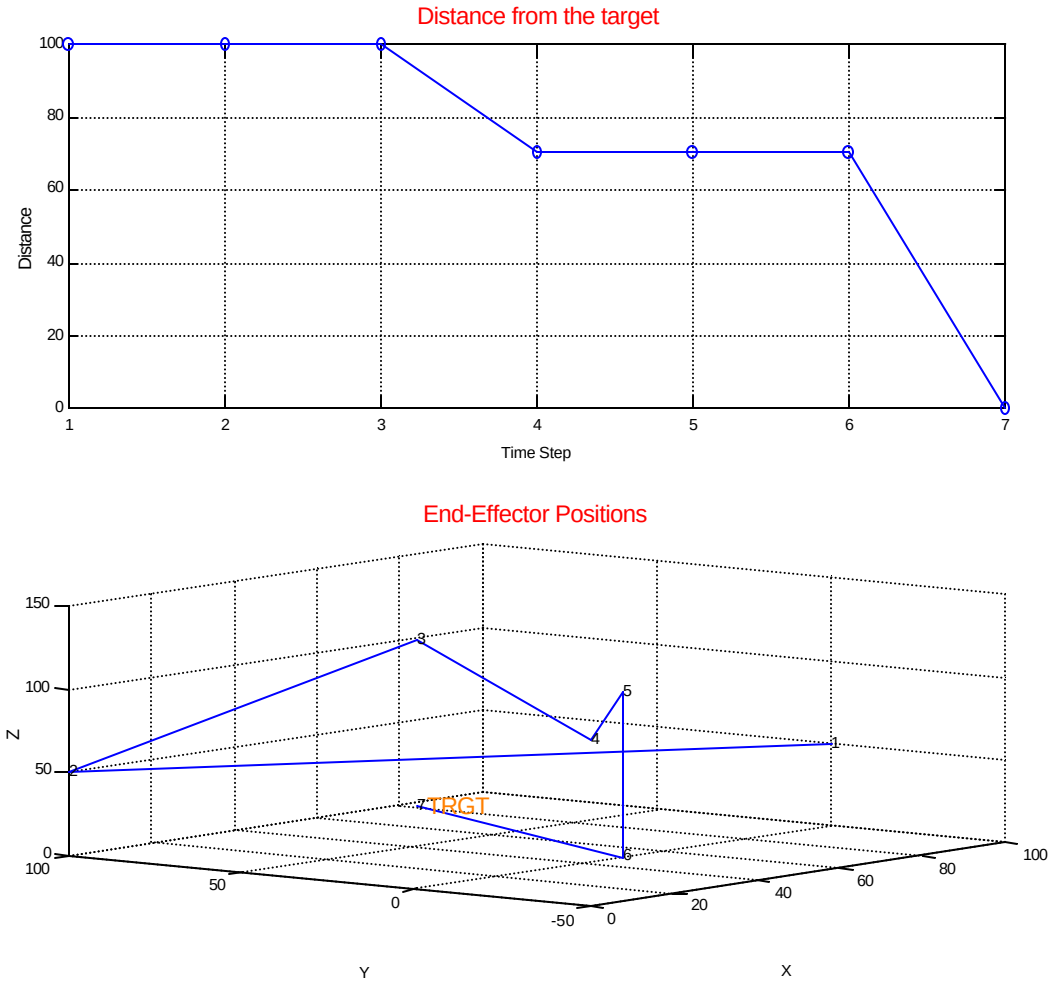


Figure4.11 Motion Analysis

If we take a look at the initial point and the target point in the task, we will find that it is required to do the following:

- Reduce the X-Axis values
- Maintain the Y-Axis value
- Maintain the Z-Axis value

If we invistigate plane projections, we will find that these requirement were fulfilled. In the XY plane projection, it was required to reduce the X-axis values to approach the goal, see Figure4.12. As we see, the End-Effector started moving away, and then appraoched the target point. The same bahavior was repeated in the ZY plane, the value of the Y-Axis was restored to reach the goal, see Figure4.13. For the Z-Axis, we can see that the value of the Z-Axis was restored approaching the target point, see Figure4.14. A combined XYZ index value tracking is given in Figure4.15 to give in further analysis in the simulation process.

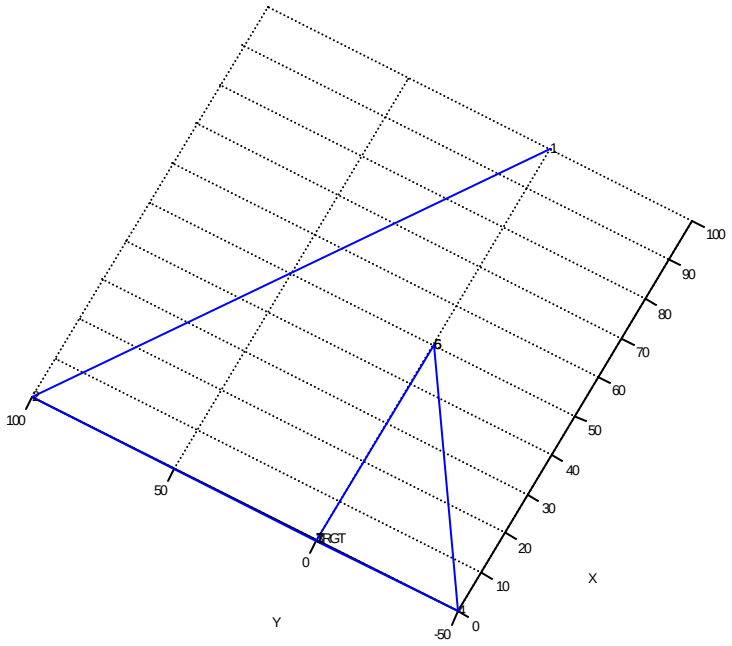


Figure4.12 XY Plane Projection

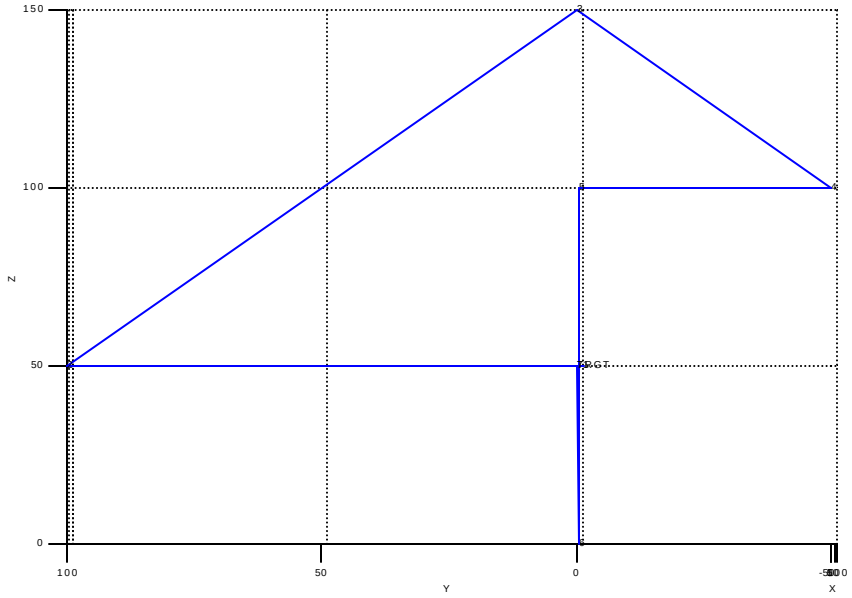


Figure 4.13 ZY Plane Projection

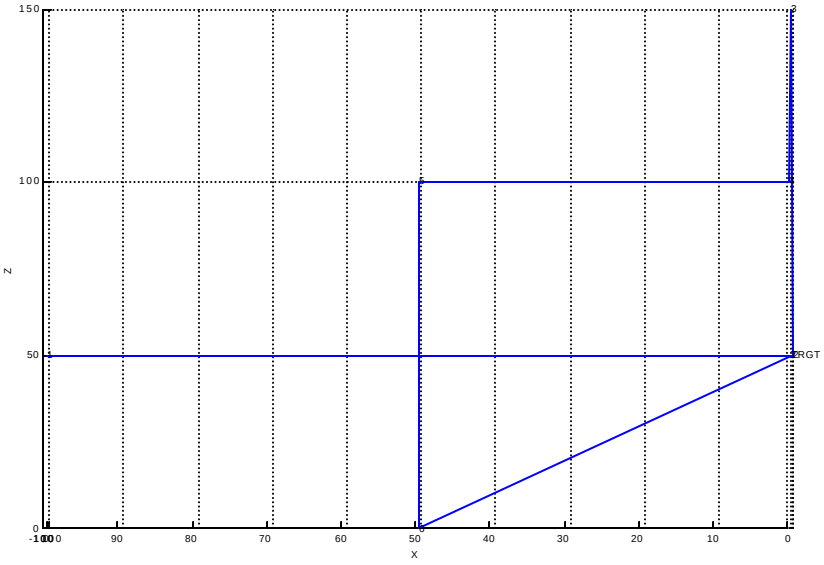


Figure 4.14 XZ Plane Projection

Figure4.16 shows the behavior concentration for each link of the manipulator. We notice that all the concentration of each behavior is increaing as the simulation progresses. This is expected since the distance from the target position is decreasing all the time.

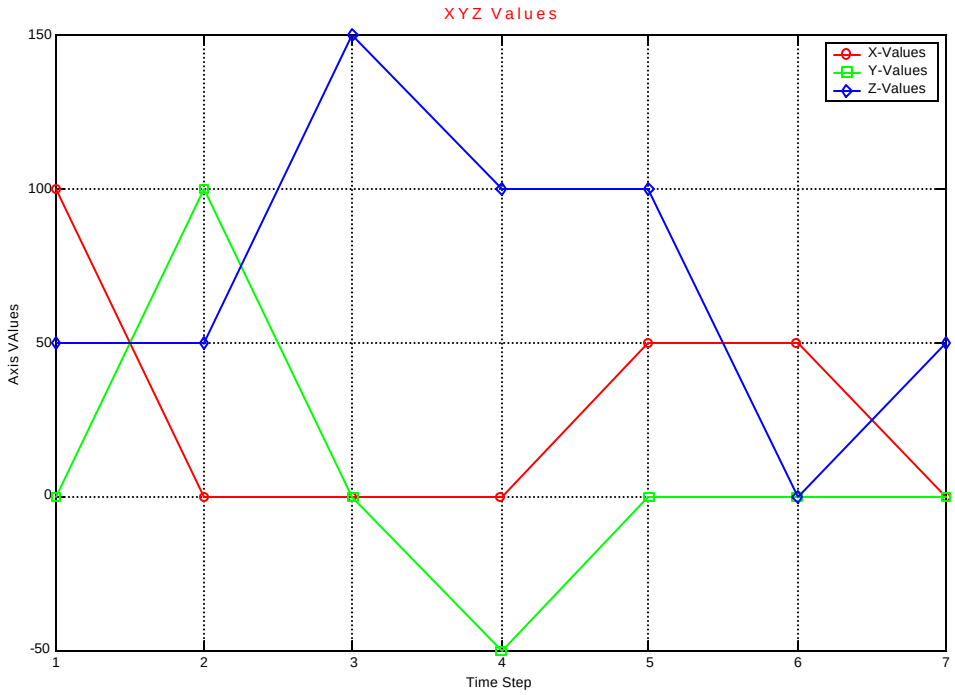


Figure4.15 XYZ Values

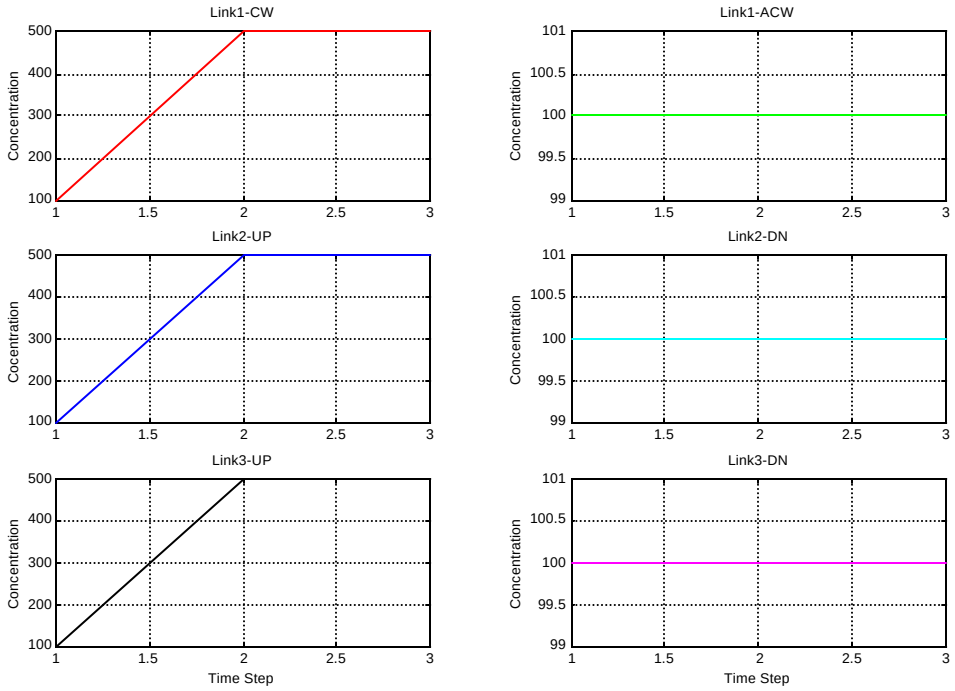


Figure 4.16 Behavior Concentrations

5. Concentration Recharging Problem

Behavior concentration plays an important role in the action selection process. A behavior might be forbidden from being executed, because its concentration level is below the acceptable concentration threshold level. A problem arises when the network fails to reach the desired goal, because all behaviors' concentrations are under the acceptable threshold. At this point the AIN training is paralyzed, and no further improvement can be achieved. We developed a modification for the simulation algorithm such that all concentrations are recharged to start a new round trying to reach the desired goal.

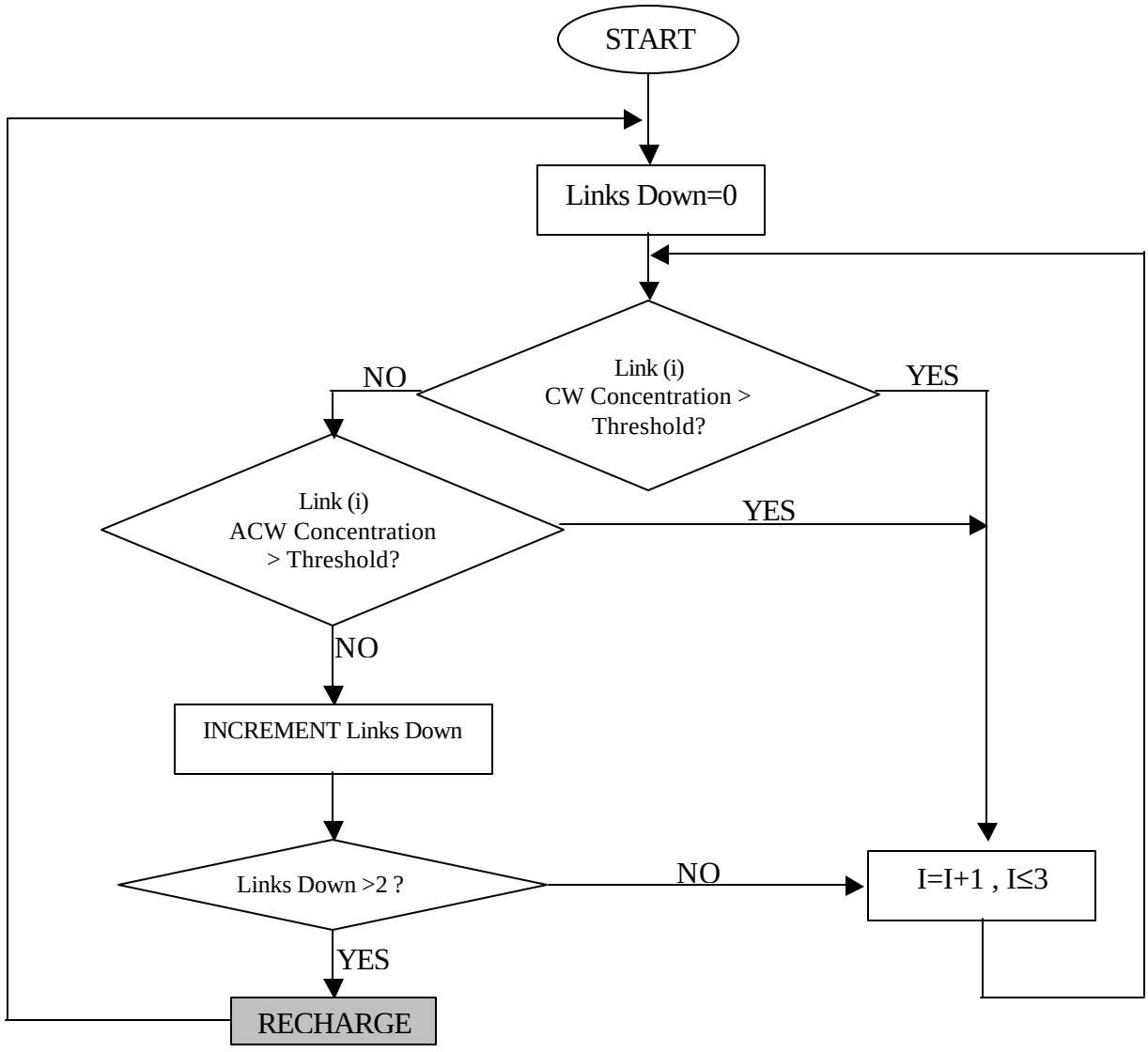


Figure4.17 Recharging Subroutine

The recharging is weigted by the previous success history of each behavior. We used this method to save simulation time, otherwise all the previous experience would be wasted resulting in redundant processing cases. Two questions arise when we want to deal with this problem:

1- When to recharge?
2- How to recharge?

To answer the first question, a subroutine is developed to solve this problem. This subroutine is executed after each iteration, where each link is given a chance to operate. Figure4.17 illustrates the flow diagram of the subroutine operation. The second problem to tackle is how to recharge again your current concentrations? We use the first recharging policy only one time at the beginning to give the network another learning chance, see Equation (4.5). The weighted recharging policy is used every time afterward, see Equation (4.6). General equations that we use to decide the new values after recharging concentrations are as follows:

- *First Recharging Policy:* $C(i+1) = C_{initial}$ (4.5)

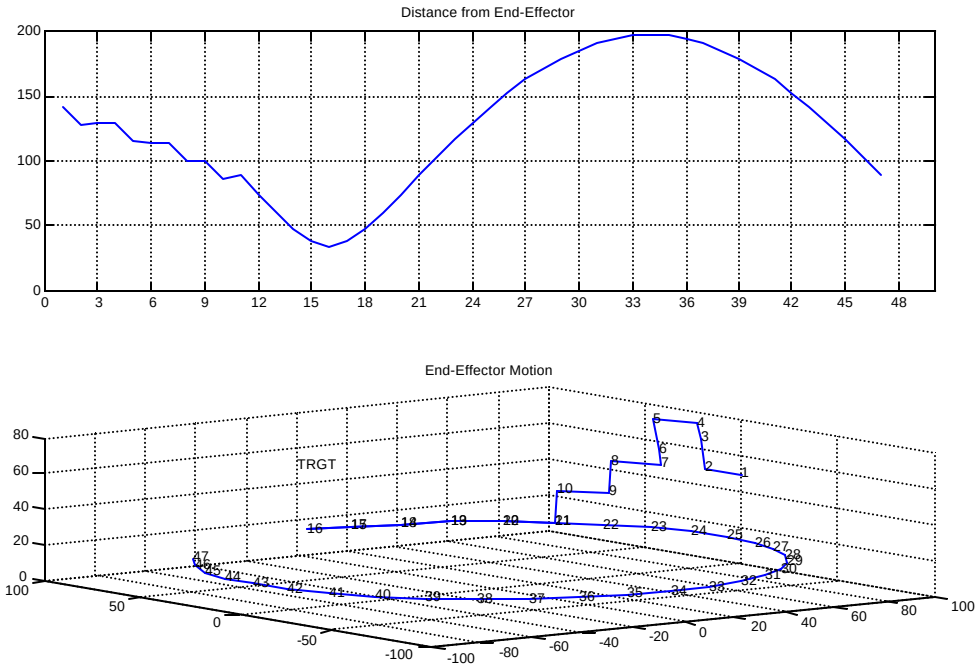
- *Second Recharging Policy:* $C(i+1) = \frac{C(i)}{C_{Total}} \times C_{initial}$ (4.6)

Where,

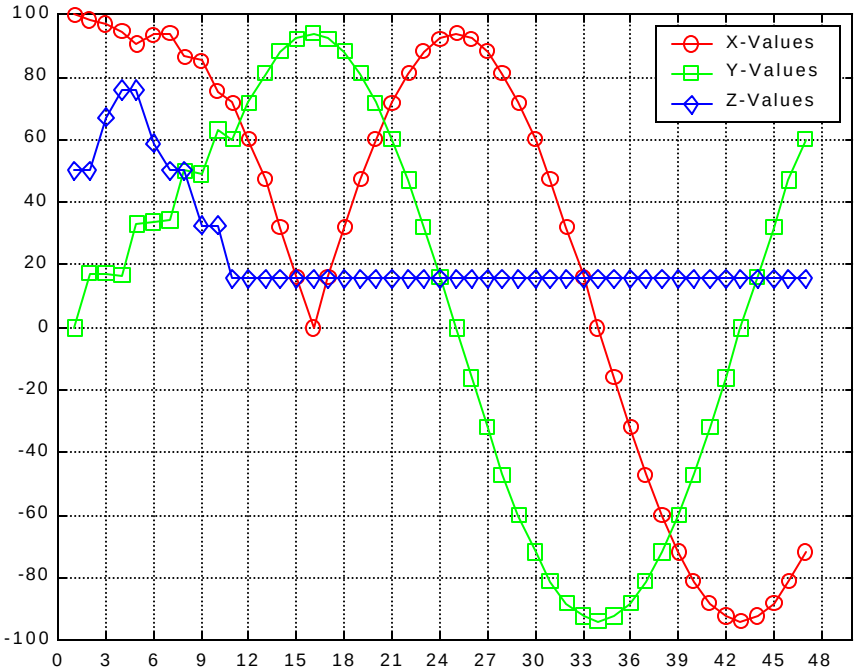
$C(i)$	The concentration of a behavior before recharging.
$C(i+1)$	The concentration of a behavior after recharging.
C_{Total}	The total concentrations of all behaviors before recharging.
$C_{initial}$	The initial Concentration that is given to all behaviors

Normally, $C_{initial}$ is much larger than $C_{Threshold}$, and as a result, the new concentration value assigned to each behavior is guaranteed to start a new round of simulation. The recharging subroutine is frequently called in case of small step size ($\Delta\theta_i$). We will list some examples in which recharging solves the training paralysis problem of the network.

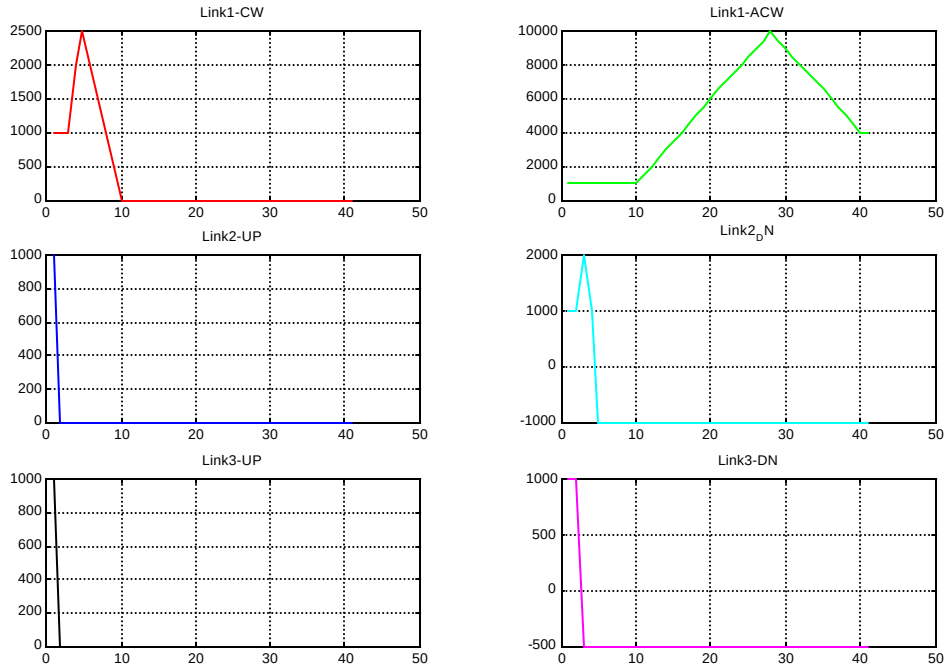
Task#3: it is required to control the motion of the robot arm while moving through the path from point (100 0 50) to (0 100 50). $\Delta\theta_i=\pi/18$, and $\xi<=20$.



(a)



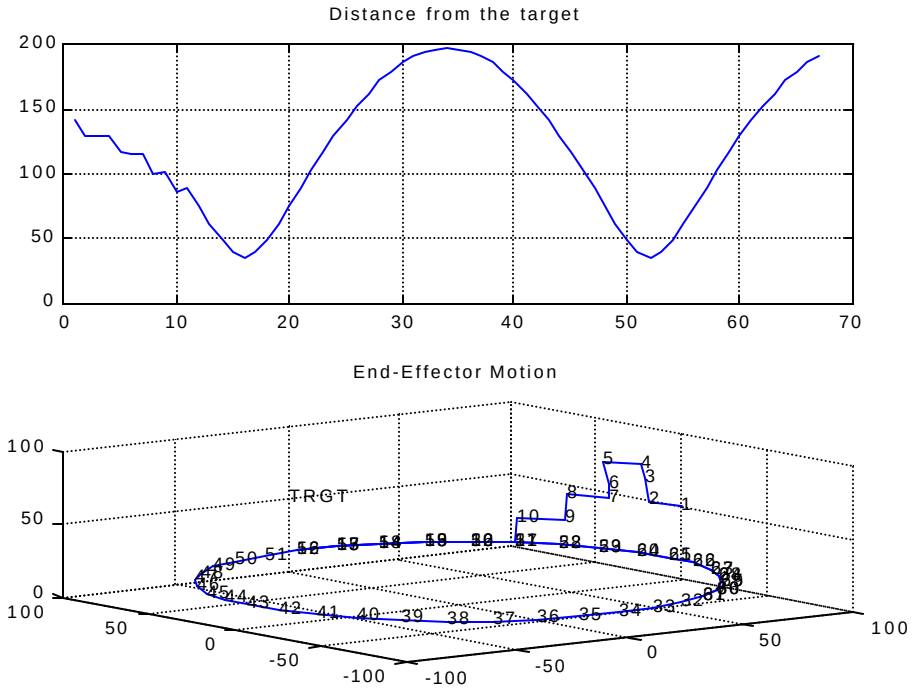
(b)



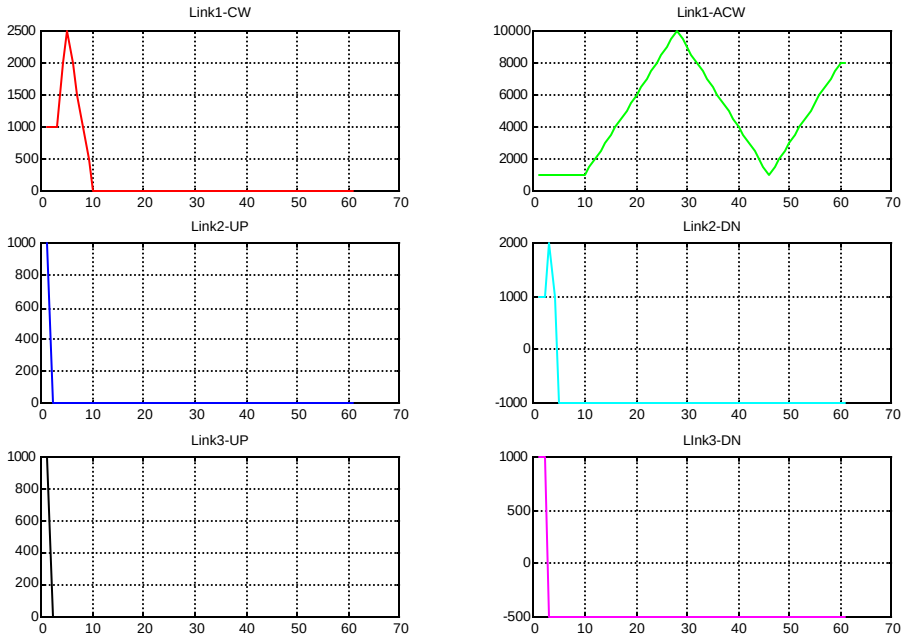
(c)

Figure4.18 (a) End-Effector Motion Before Using the Recharging Subroutine, (b) XYZ Values, and (c) Behaviors' Concentrations

The problem that arises at Figure4.18.(a) is that the concentrations of Link2, and Link3 behaviors do not improve the learning progresses. At the same time, Link1 behaviors' concentrations continue to be updated for both CW behavior, and the ACW behavior. In the end the only active behavior in the network is Link1 ACW, which leads to undesireable End-Effector motion. When we give the network more time to learn, the behavior starts to oscillate, see Figure4.19. The reason for doing this is that at the time step number 48, which was the end of the learning session in Figure 4.18(a), gives a promissing hope that the distance from the target is getting smaller. Now, we will use the recharging subroutine to overcome the oscillation effect.



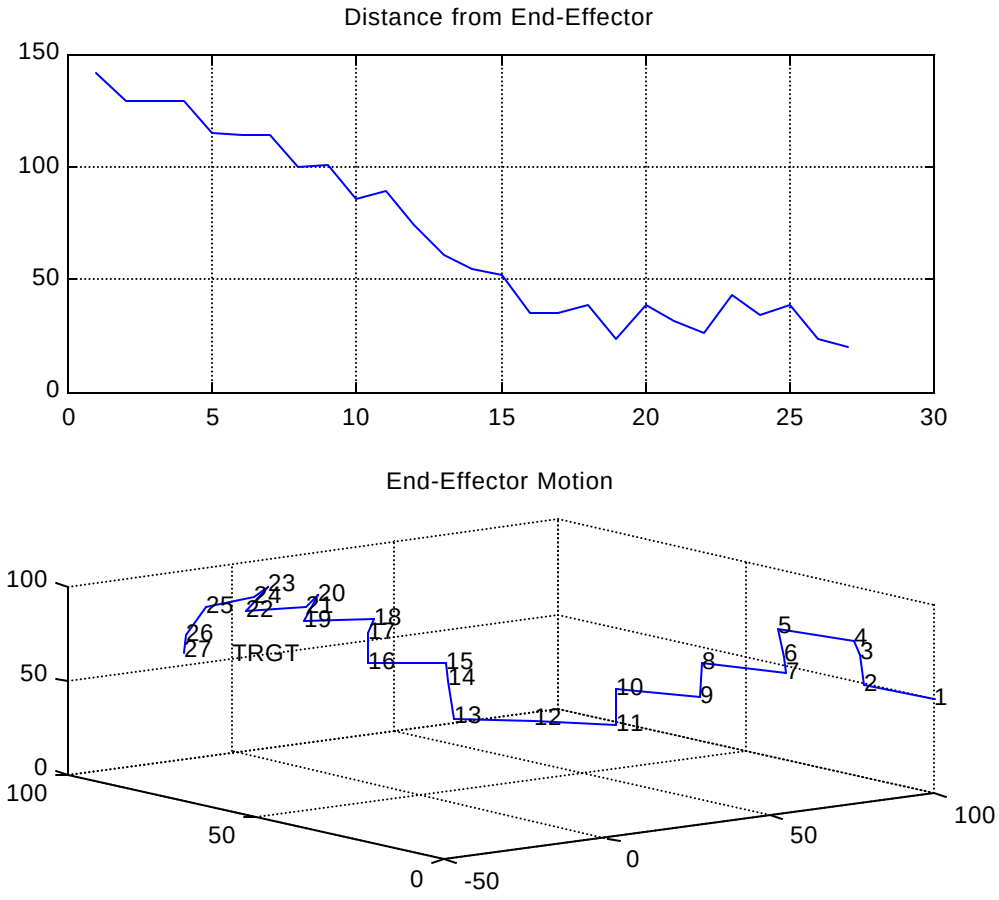
(a)



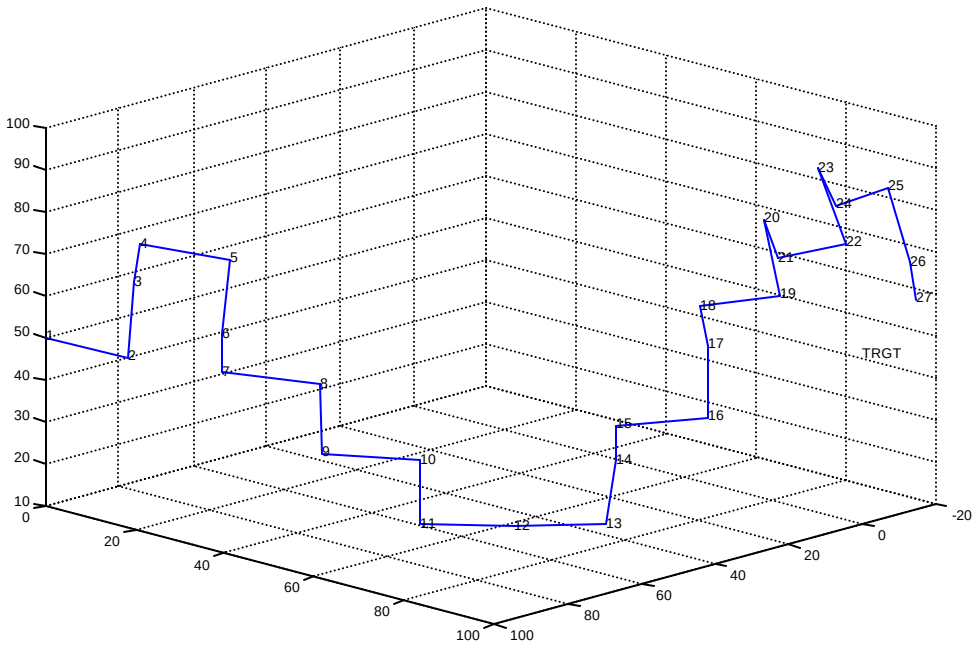
(b)

Figure4.19 Further Training for Task#3 (a) Distance from the Target (b) Behaviors' Concentrations

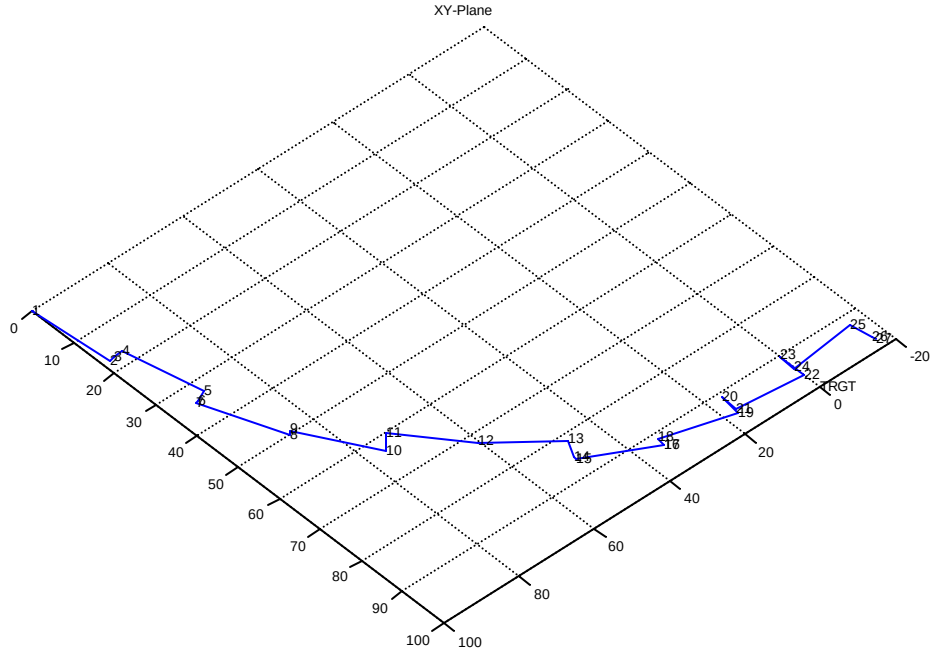
The recharging subroutine is coded and added to the arm simulation Matlab program. The progress report shows that the recharging subroutine is called once at step 13 specifically for this simulation run. The results shows that concentration recharging has a dramatic effect on the AIN learning process. Figure4.20(a) shows that the distance from the end-effector is gradually decreasing after point 13. Further analysis for the end-effector motion can be deducted from different motion projections on Figure 4.20 (c,d,e, and f). Another advantage is the undesired performance of Link#1-ACW is completely suppressed compared to the previous case, compare Figure4.19 (c) and Figure4.20 (g).



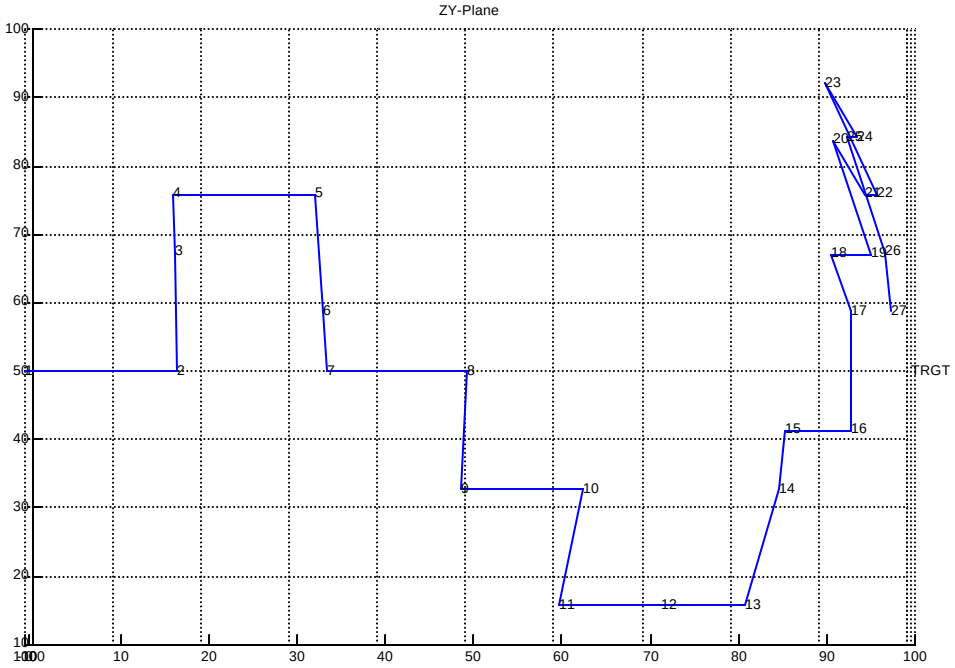
(a)



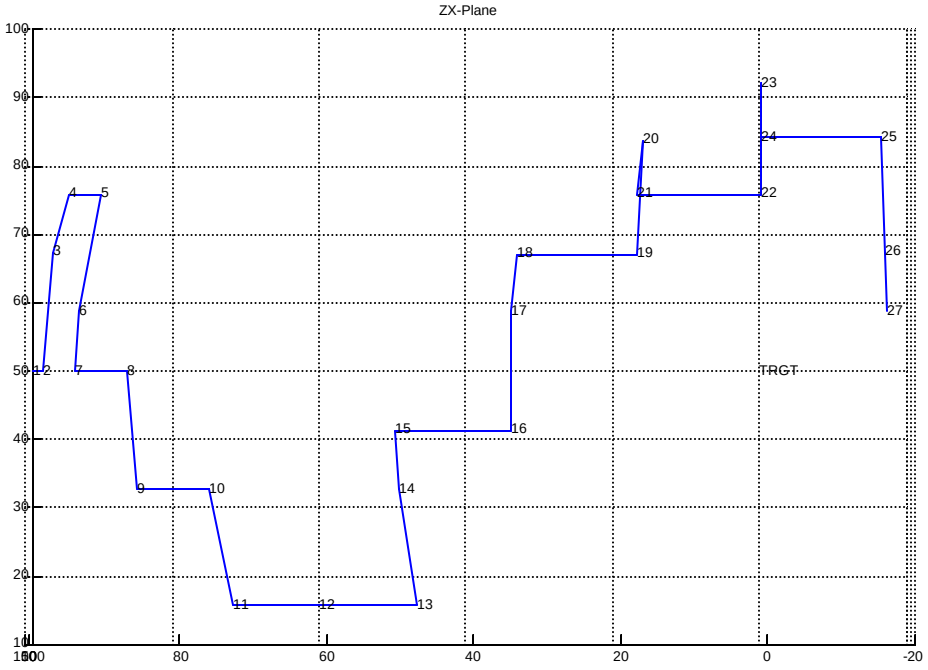
(b)



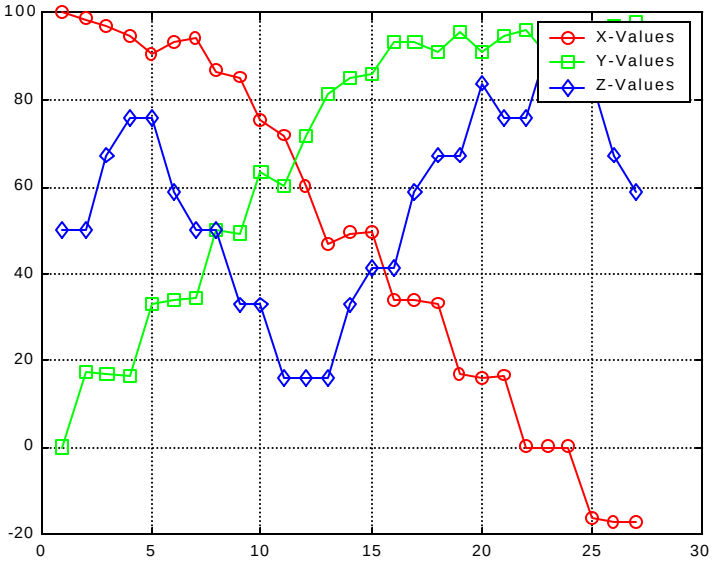
(c)



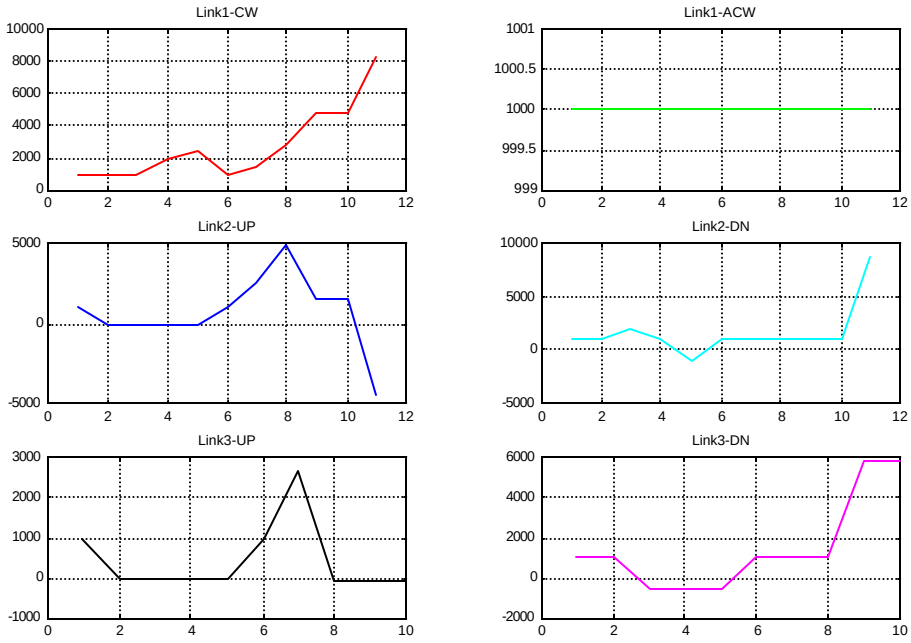
(d)



(e)



(f)



(g)

Figure4.20 AIN Simulation Learning Results After Using the Recharging Subroutine (a) Distance of End-Effector From the Target (b) 3D Motion of the End-effector (c) XY Plane Projection (d) ZY Plane Projection (e) ZX Plane Projection (f)XYZ Values (g) Behavior's Concentrations

In this chapter we spotlighted the AIN training, and different difficulties that are encountered. We gave some examples showing how the elements of the immune network interact among each other to achieve the desired goal. The next step is to feed this data into the memory of the immune system, which in turn will enhance the response in the future performance of the neuro-immune network as a whole.