Kan, we could [illegible] back.

Project Report for Robotics Seminar

by Chuck Anderson and Francisco Cervantes

January 3, 1982

1. Introduction

The Adaptive Networks Group here at UMass has been investigating ways of combining searching and learning methods as an approach to solving difficult problems. They have designed the Associative Search Network, or ASN, as a working model of such an approach. Our objective in this project is to show how an ASN can be applied to problems in manipulator control.

The analysis of a manipulator can be quite complex and the application of classical control techniques to manipulators in general is restricted. It is a formidable, and often impossible task to determine the appropriate controls for each joint for every possible environmental situation. To tackle these complex control problems we must devise algorithms that are capable of searching for the correct control actions. The amount of search can be greatly reduced by incorporating the ability to learn from experience. Thus, the ASN algorithm miht be usefully applied to these problems.

Todate, the ASN has been applied to several abstract test beds and to the control of a simulated, physical system. Applying the ASN to a real system could reveal its true potential and expose those design flaws that might be hidden by the use of simulated test beds. Thus, our objectives are twofold: 1) To demonstrate the potential of the ASN as a device for controlling manipulators, 2) To further our research with the ASN by performing experiments with a real system.

2. Method

We planned to complete this project in one semester so we designed our experiments to require very little additional research with the ASN. At the semester's end the manipulator's hardware was not completed. However, we have finished the implementation of the ASN and have tested it with a crude simulation of the manipulator's behavior. Our experiments involved a manipulator task that is a direct analog of some previous experiments with an ASN and an abstract test bed. The manipulator task is outlined below, after which the ASN is described.

2.1 The Task

The task is to learn the control actions that will move the manipulator's end effector to a particular point within its work space from any other point. This is certainly not a difficult problem for simple manipulators and is routinely done by most manipulators in use today. However, most control algorithms require certain assumptions about the environment to be true, such as the placement of parts and obstacles. As more sensory feedback is provided, these assumptions are lessened, but the complexity of the controller can increase dramatically. Note that our objective in this project is to demonstrate the general paradigm with which an ASN can be applied to manipulator control, and not to solve a difficult control problem.

The generality of the paradigm is due to the lack of specific constraints on the input and output of the ASN. All the ASN requires from the manipulator is a vector of signals that specify the current position of the end effector and a signal indicating whether the previous control action caused the end effector to move towards or away from the desired point in space. The manner in which the current position is encoded by the vector of signals is not known to the ASN.

The output of the ASN is supplied to the manipulator as control signals. Here again the ASN does not know how these control signals affect the manipulator. For the primary task of this project, each control signal will simply move one of the joint variables by a fixed amount.

The task can be extended in several ways. One way would be to allow the desired point in space to be moved, and providing the ASN with an additional vector of signals indicating its location. The task can also be extended by altering the manner in which the ASN's output affects the manipulator. Rather than fixed increments in the joint variables, the control signals might indicate variable increments, velocities, or increments in velocities. The generality of the ASN leaves much room for experimentation.

2.2 The ASN

The structure of the ASN resembles an associative memory network in that it receives an input vector and generates an output vector whose components are functions of the inner product of the input vector and the weight vectors. The ASN that is proposed for this project is pictured in Fig. 1.

The computation of a y_j actually involves a threshold function of the input and weight vectors' inner product plus noise, defined by the equations

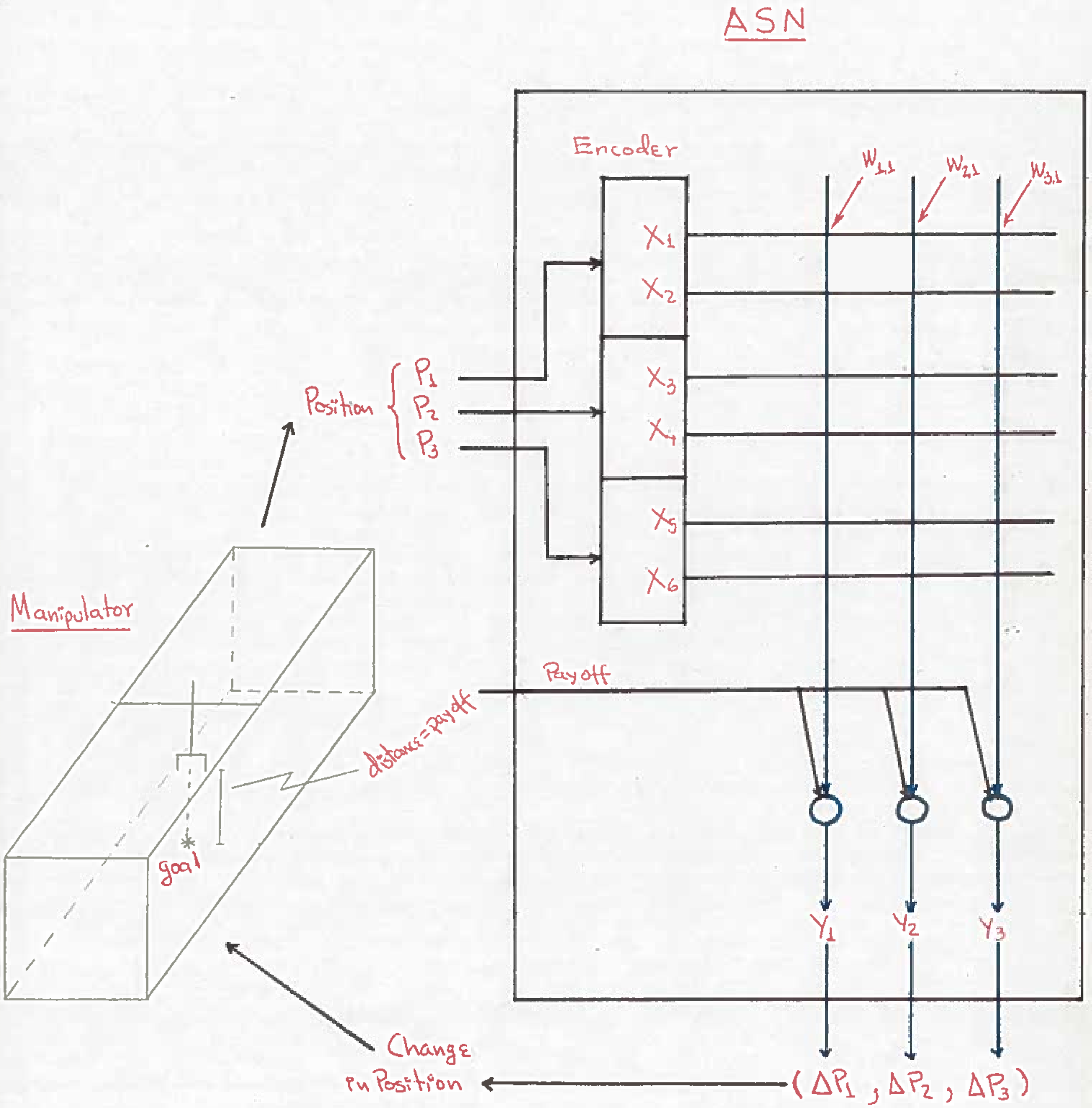
$$s_j(t) = \sum_{i=1}^{6} x_i(t)\, w_{ji}(t) \quad , \qquad y_j(t) = \begin{cases} 1, & \text{if } s_j(t) + Noise_j(t) > 0 \\ 0, & \text{if } \quad " \quad = 0 \\ -1, & \text{if } \quad " \quad < 0 \end{cases}$$

where $j=1,2,3$ and $Noise_j$ is a random variable sampled from a Gaussian distribution. The $Noise_j$ term provides the "search" function of the ASN by generating alternative values for the ASN's output.

Let's relate these variables to the project's task. The inputs $x_1, \ldots, x_6$ encode the current position of the end effector. Each x_i is dependent on one component of the position vector as detailed in Fig. 2.

The effect of the outputs y_1, y_2, y_3 on the position vector (p_1, p_2, p_3) is given by

$$\Delta p_i = a\, y_i \; , \quad \text{where } i = 1,2,3 \text{ and } a = 1$$



ASN
Encoder
$W_{1,1}$
$W_{2,1}$
$W_{3,1}$
X_1
X_2
X_3
X_4
X_5
X_6
Position
P_1
P_2
P_3
Manipulator
Pay off
distance=payoff
goal
Y_1
Y_2
Y_3
Change in Position
$(\Delta P_1, \Delta P_2, \Delta P_3)$

Fig. 1

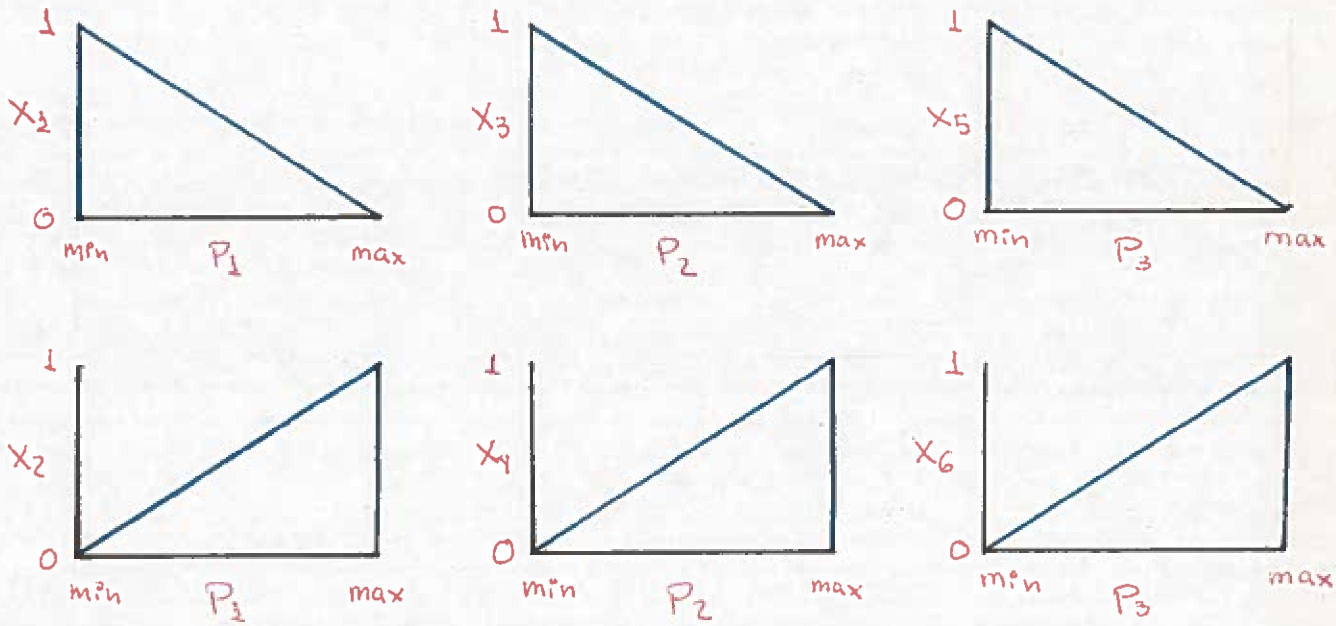

1
X1
0
min
P1
max
1
X3
0
min
P2
max
1
X5
0
min
P3
max
1
X2
0
min
P1
max
1
X4
0
min
P2
max
1
X6
0
min
P3
max

Fig. 2

The ASN "learns from experience" by receiving a payoff indicating the distance between the goal and the manipulator's current position, and computing an evaluation, or reinforcement r, of the previous action by subtracting this new payoff from the previous payoff. The weights are updated with the equation

$$r(t) = \text{payoff}(t-1) - \text{payoff}(t)$$

$$W_{ji}(t+1) = W_{ji}(t) + c\, r(t)\, y_j(t)\, x_i(t)$$

for i=1,...,4 and j=1,...,3.

2.3 Software

We are running our experiments with FORTH routines on the PDP 11/23. These routines can be divided into three groups: 1) those involved in computations to decide the next best action, i.e., setting initial conditions and solving the equations representing the ASN model; 2) the routines which serve to interact with the user at the terminal, i.e., running an experiment and specifying the detail of the display; 3) routines related with the movement of the manipulator's linear joints. All routines are contained in the FORTH listing in Appendix B.

2.3.1 ASN Routines

0 0 0 !GOAL

This routine serves to set the goal for our simulations. When we use the manipulator, the goal will be specified simply by placing the distance encoder at a place in the workspace.

INIT

This command initializes the ASN weights to zero, which represents the initial conditions at the start of any experiment.

10 C ! or 20 STD !

We use these FORTH instructions to initialize the values of the learning constant, C, and the standard deviation, STD, of the noise distribution.

CALC-ACT

This routine computes the next action to be applied to the manipulator's linear joints. The result is a 3-dimensional vector whose components are 1, 0 or -1. These values are multiplied by a fixed step size before being applied to the manipulator. In our simulations this step size is 1.

LEARN-NET

This routine updates the values of the ASN's weights.

2.3.2 Commands for Running Experiments and Displaying Results

SHOW-NET

This routine displays at the terminal the current state of the ASN, including the weight values, the action, and the input values.

100 STEPS

This routine serves to run an experiment for the specified number of steps. When complete, control returns to the terminal at which time parameters can be changed and the experiment continued by again using STEPS. This routine simply uses a sequence of commands to sense the manipulator's position, compute the actions, apply them to the manipulator, sense the payoff, update the ASN's weights, and display the current state of the experiment.

LONG

This sets the "display" switch to indicate that for every step the results of each substep, i.e., sensing, action calculation and application, and learning, will be shown.

SHORT

This sets the "display" switch to indicate that just the step number and the current position of the manipulator will be shown for each step.

VERY-SHORT

This sets the "display" switch to indicate that the step number and the current position will be shown for the final step only.

2.3.3 Manipulator-dependent Commands

These commands are what we need from the Robotics Group to run our experiments with the real manipulator.

SENSE (-- JX JY JZ)

This routine should return the current values of the joint variables for the three linear joints. These will be encoded (by one of our routines) into the ASN's input values $x_1, \ldots, x_2$.

33 100 -66 MOVE-TO (JX JY JZ --)

This routine must allow us to place the end effector of the manipulator at any point in the working space by specifying the values of the joint variables for each of the three linear joints. We are free to use any convenient units, such as counter increments. This routine would not be required, if we could move the manipulator manually before starting each training session.

2 -2 2 ACT (JX JY JZ --)

This routine is used to change the manipulator's position, according to the actions calculated by the routine CALC-ACT. The values JX, JY, and JZ would most simply be increments or decrements in the joint counters.

EVAL (-- Distance)

This routine should return the current value of the distance encoder, which will actually be the encoder on the fourth joint, i.e., the rotational joint. This value, as all others, could simply be in counter units.

3. Results

For our simulations, we arbitrarily defined the manipulator's workspace to have dimensions of −100 to 100 in joint variable units. Appendix A shows the results of a particular experiment for which the goal was set to (0,0,0), the center of the workspace. The position of the manipulator was initialized to (10,10,10) and the ASN's weights were set to zero. The first 10 steps were run using the long display as a demonstration of the substeps involved. The ASN's input, weights, and output are displayed in a format meant to resemble the structure drawn in Fig. 1. The next 90 steps were run in short form, to show just the movement of the simulated manipulator.

The position changes from (10,10,10) to positions where all components are negative, and then back to all positive components. This oscillatory behavior is due to the manner in which learned actions are generalized throughout the workspace. Associating a ($-\Delta JX, -\Delta JY, -\Delta JZ$) action with position (10,10,10) initially generalizes to other positions, including those with all negative components, thus driving the position away from

(0,0,0) as it becomes negative. This continues until a $(+\Delta JX, +\Delta JY, +\Delta JZ)$ action becomes reliably associated with a negative position. The magnitude of the oscillations tend to decrease as the generalizations of opposite actions "cancel" each other in the region near the goal, which is (0,0,0) for this experiment.

After suppressing most of the output with the VERY-SHORT command a further 1900 steps are run. We then demonstrated what the ASN had learned by setting the ASN's parameters C and STD to zero to inhibit any subsequent learning and then letting the ASN move the manipulator from a number of starting positions. After 100 steps the position had either stopped changing or was oscillating between two positions. The results are listed below.

STARTING POSITION			POSITION AFTER 100 STEPS		
JX	JY	JZ	JX	JY	JZ
50	20	-70	0	2	-3
-80	5	-65	0	3	-3
10	70	-55	0	2	-3
60	60	60	0	2	-3
70	-2	-55	0	3	-3

With additional learning we would expect these final positions to get increasingly closer to the goal of (0,0,0).

4. Discussion

Using a simulation of the gross behavior of the manipulator, we have shown that an ASN consisting of three elements each receiving six inputs is capable of performing a simple goal-seeking task with the manipulator. The importance of this demonstration is not in the selection of the task, but in the type of controller used, i.e., the ASN.

Learning controllers can be compared in at least two major ways. One way is by the type of representation used for the state of the controlled plant. This is often described geometrically in terms of the state space of the plant. For example, the BOXES system of Michie and Chambers represented the state of an inverted pendulum by dividing the state space into many fixed, nonoverlapping rectangles, or "boxes", and designating which box contained the state at specific times. Raibert also divides the state space into fixed, nonoverlapping regions. The CMAC system of Albus also uses a state space with many fixed regions, but the regions are overlapping. In our demonstration, the ASN used only a few large, overlapping regions.

Two opposing factors contribute to the choice of a representation. The first factor is generalization. The ASN, using large overlapping regions, was able to generalize, or apply, an action learned in one state to many other states. This greatly facilitated learning, since experience with a relatively small number of states enabled the ASN to apply the appropriate actions throughout the state space. A representation consisting of many smaller regions would not have this property. Experience in one region would generalize to relatively few other states. The amount of generalization depends on the size of the regions; to maximize the amount of generalization the regions should be as large as possible. However, the size of the regions is limited by a second factor, the resolution required by the action-selection mechanism. The action-selection algorithm and the particular control surface that is to be implemented by the action-selection algorithm set an upper limit on the size of the regions. Thus, there is a tradeoff between the amount of generalizatiion and the complexity of the possible control surfaces that can be implemented with a certain representation. Ideally, a learning controller would be capable of developing a representation that is appropriate for a given task.

This is the approach that we are currently pursuing with the ASN. We believe that by constructing a system of goal-seeking elements this type of behavior can be achieved. Briefly, we are considering a two-layered ASN. Both layers are similar to the network used in this project. The output of the first layer becomes the input to the second layer. With this structure, the first layer will be learning a transformation of the state space into a representation with which the second layer can perform the action-selection task. We have preliminary results from experiments employing two-layered ASN's, including test beds involving a simple type of obstacle avoidance tasks.

A second way of categorizing learning controllers is by the types of learning algorithms, or rules. The Albus CMAC system employs an error-correction rule. When applied to control problems, this requires the a priori knowledge of the desired control surface with which the output of the CMAC system can be compared and an error computed. However, for many control problems the desired control surface is unknown. Michie and Chambers' BOXES system performed a type of reinforcement learning in which the only evaluation received was an infrequent punishment. The desired control surface need not be known a priori for this procedure. One problem, though, is the infrequent opportunity to learn. The ASN in this project assumed that an evaluation was available after every movement.

We are currently investigating ways of bridging this gap between infrequent and frequent evaluations. Our approach has been to design an additional element that learns to predict an evaluation for each state. Thus, for steps in which no external evaluation is available, this internal "critic" provides the needed information about the desirability of the previous action.

In summary, this project involves a simple ASN that learns to perform a simple control task. We do not claim that the performance of this task is worthy of consideration, but we do claim that the adaptive control algorithm of the ASN has potential significance for difficult control problems, such as those that arise in manipulator control.

APPENDIX A

```
OK HELP

          AVAILABLE COMMANDS:
MOVE-TO     MOVES ARM TO GIVEN POSITION (X Y Z MOVE-TO)
INIT        RESETS THE NETWORK VARS TO ZERO (INIT)
STEPS       RUNS THE EXPERIMENT FOR N STEPS (N STEPS)
LONG        SHOW ALL DETAILS OF EACH STEP (LONG)
SHORT       JUST SHOW POSITION AT EACH STEP (SHORT)
VERY-SHORT  SHOW NOTHING AT EACH STEP (VERY-SHORT)
SHOW-NET    DISPLAYS CURRENT STATE OF NET (SHOW-NET)
TEST        FIND THE EQUILIBRIUM POSITION OF THE NET (TEST)

OK LONG
OK 10 10 10 MOVE-TO
DISTANCE ENCODER RETURNED 300
OK INIT
GOAL IS 0 0 0
DISTANCE ENCODER RETURNED 300
C= 100   STD= 200
STEP NUMBER 0   POSITION = 10   10   10
PAYOFF= 300   OLD PAYOFF= 300   REINFORCEMENT= 13
X   0.00      0.00    0.00    0.00
    0.00      0.00    0.00    0.00
Y   0.00      0.00    0.00    0.00
    0.00      0.00    0.00    0.00
Z   0.00      0.00    0.00    0.00
    0.00      0.00    0.00    0.00

              0.00    0.00    0.00
              0.00    0.00    0.00
               DX      DY      DZ
OK 10 STEPS
C= 100   STD= 200
SENSES RETURNED POSITION 10  10  10
NETWORK GENERATED THE ACTION (DX,DY,DZ)= -1.00-1.00-1.00
ARM MOVED FROM POSITION 10  10  10  TO 9  9  9
DISTANCE ENCODER RETURNED 243
NETWORK WEIGHTS UPDATED USING REINFORCEMENT OF 57
STEP NUMBER 1   POSITION = 9   9   9
PAYOFF= 243   OLD PAYOFF= 300   REINFORCEMENT= 57
X   0.12     -0.06   -0.06   -0.06
    0.38     -0.21   -0.21   -0.21
Y   0.12     -0.06   -0.06   -0.06
    0.38     -0.21   -0.21   -0.21
Z   0.12     -0.06   -0.06   -0.06
    0.38     -0.21   -0.21   -0.21

              0.00    0.00    0.00
             -1.00   -1.00   -1.00
               DX      DY      DZ
SENSES RETURNED POSITION 9  9  9
NETWORK GENERATED THE ACTION (DX,DY,DZ)= -1.00-1.00-1.00
ARM MOVED FROM POSITION 9  9  9  TO 8  8  8
DISTANCE ENCODER RETURNED 192
NETWORK WEIGHTS UPDATED USING REINFORCEMENT OF 51
STEP NUMBER 2   POSITION = 8   8   8
PAYOFF= 192   OLD PAYOFF= 243   REINFORCEMENT= 51
X   0.13     -0.12   -0.12   -0.12
    0.37     -0.39   -0.39   -0.39
Y   0.13     -0.12   -0.12   -0.12
    0.37     -0.39   -0.39   -0.39
Z   0.13     -0.12   -0.12   -0.12
    0.37     -0.39   -0.39   -0.39
```

```
              -0.21  -0.21  -0.21
              -1.00  -1.00  -1.00
                DX     DY     DZ
SENSES RETURNED POSITION 8  8  8
NETWORK GENERATED THE ACTION (DX,DY,DZ)= -1.00 1.00 1.00
ARM MOVED FROM POSITION 8  8  8  TO 7  9  9
DISTANCE ENCODER RETURNED 211
NETWORK WEIGHTS UPDATED USING REINFORCEMENT OF -19
STEP NUMBER 3  POSITION = 7  9  9
PAYOFF= 211   OLD PAYOFF= 192   REINFORCEMENT= -19
X   0.15    -0.10  -0.14  -0.14
    0.35    -0.33  -0.45  -0.45
Y   0.15    -0.10  -0.14  -0.14
    0.35    -0.33  -0.45  -0.45
Z   0.15    -0.10  -0.14  -0.14
    0.35    -0.33  -0.45  -0.45

            -0.42  -0.42  -0.42
            -1.00   1.00   1.00
              DX     DY     DZ
SENSES RETURNED POSITION 7  9  9
NETWORK GENERATED THE ACTION (DX,DY,DZ)=  1.00 1.00-1.00
ARM MOVED FROM POSITION 7  9  9  TO 8  10  8
DISTANCE ENCODER RETURNED 228
NETWORK WEIGHTS UPDATED USING REINFORCEMENT OF -17
STEP NUMBER 4  POSITION = 8  10  8
PAYOFF= 228   OLD PAYOFF= 211   REINFORCEMENT= -17
X   0.16    -0.12  -0.16  -0.12
    0.34    -0.38  -0.50  -0.40
Y   0.13    -0.12  -0.16  -0.12
    0.37    -0.39  -0.51  -0.39
Z   0.13    -0.12  -0.16  -0.12
    0.37    -0.39  -0.51  -0.39

            -0.38  -0.51  -0.51
             1.00   1.00  -1.00
              DX     DY     DZ
SENSES RETURNED POSITION 8  10  8
NETWORK GENERATED THE ACTION (DX,DY,DZ)=  1.00-1.00-1.00
ARM MOVED FROM POSITION 8  10  8  TO 9  9  7
DISTANCE ENCODER RETURNED 211
NETWORK WEIGHTS UPDATED USING REINFORCEMENT OF 17
STEP NUMBER 5  POSITION = 9  9  7
PAYOFF= 211   OLD PAYOFF= 228   REINFORCEMENT= 17
X   0.15    -0.10  -0.18  -0.14
    0.35    -0.33  -0.55  -0.45
Y   0.12    -0.10  -0.18  -0.14
    0.38    -0.33  -0.57  -0.45
Z   0.15    -0.10  -0.18  -0.14
    0.35    -0.34  -0.56  -0.44

            -0.43  -0.58  -0.44
             1.00  -1.00  -1.00
              DX     DY     DZ
SENSES RETURNED POSITION 9  9  7
NETWORK GENERATED THE ACTION (DX,DY,DZ)=  1.00 1.00 1.00
ARM MOVED FROM POSITION 9  9  7  TO 10  10  8
DISTANCE ENCODER RETURNED 264
NETWORK WEIGHTS UPDATED USING REINFORCEMENT OF -53
STEP NUMBER 6  POSITION = 10  10  8
PAYOFF= 264   OLD PAYOFF= 211   REINFORCEMENT= -53
X   0.13    -0.16  -0.24  -0.20
    0.37    -0.52  -0.74  -0.64
Y   0.13    -0.16  -0.24  -0.20
    0.37    -0.52  -0.74  -0.64
```

```
     0.34    -0.52  -0.74  -0.62

             -0.38  -0.66  -0.50
              1.00   1.00   1.00
               DX     DY     DZ
SENSES RETURNED POSITION 10  10  8
NETWORK GENERATED THE ACTION (DX,DY,DZ)= -1.00 1.00-1.00
ARM MOVED FROM POSITION 10  10  8  TO 9  11  7
DISTANCE ENCODER RETURNED 251
NETWORK WEIGHTS UPDATED USING REINFORCEMENT OF 13
STEP NUMBER 7  POSITION = 9  11  7
PAYOFF= 251   OLD PAYOFF= 264   REINFORCEMENT= 13
X    0.12    -0.17  -0.23  -0.21
     0.38    -0.56  -0.70  -0.68
Y    0.12    -0.17  -0.23  -0.21
     0.38    -0.56  -0.72  -0.68
Z    0.15    -0.19  -0.25  -0.23
     0.35    -0.56  -0.70  -0.66

             -0.60  -0.88  -0.76
             -1.00   1.00  -1.00
               DX     DY     DZ
SENSES RETURNED POSITION 9  11  7
NETWORK GENERATED THE ACTION (DX,DY,DZ)=  1.00 1.00-1.00
ARM MOVED FROM POSITION 9  11  7  TO 10  12  6
DISTANCE ENCODER RETURNED 280
NETWORK WEIGHTS UPDATED USING REINFORCEMENT OF -29
STEP NUMBER 8  POSITION = 10  12  6
PAYOFF= 280   OLD PAYOFF= 251   REINFORCEMENT= -29
X    0.13    -0.20  -0.26  -0.18
     0.37    -0.66  -0.80  -0.58
Y    0.11    -0.20  -0.26  -0.18
     0.39    -0.67  -0.83  -0.57
Z    0.16    -0.23  -0.29  -0.19
     0.34    -0.65  -0.79  -0.57

             -0.66  -0.84  -0.80
              1.00   1.00  -1.00
               DX     DY     DZ
SENSES RETURNED POSITION 10  12  6
NETWORK GENERATED THE ACTION (DX,DY,DZ)= -1.00 1.00-1.00
ARM MOVED FROM POSITION 10  12  6  TO 9  13  5
DISTANCE ENCODER RETURNED 275
NETWORK WEIGHTS UPDATED USING REINFORCEMENT OF 5
STEP NUMBER 9  POSITION = 9  13  5
PAYOFF= 275   OLD PAYOFF= 280   REINFORCEMENT= 5
X    0.12    -0.20  -0.26  -0.18
     0.38    -0.67  -0.79  -0.59
Y    0.09    -0.20  -0.26  -0.18
     0.41    -0.69  -0.81  -0.59
Z    0.17    -0.23  -0.29  -0.19
     0.33    -0.66  -0.78  -0.58

             -0.79  -0.99  -0.69
             -1.00   1.00  -1.00
               DX     DY     DZ
SENSES RETURNED POSITION 9  13  5
NETWORK GENERATED THE ACTION (DX,DY,DZ)= -1.00 1.00 1.00
ARM MOVED FROM POSITION 9  13  5  TO 8  14  6
DISTANCE ENCODER RETURNED 296
NETWORK WEIGHTS UPDATED USING REINFORCEMENT OF -21
STEP NUMBER 10  POSITION = 8  14  6
PAYOFF= 296   OLD PAYOFF= 275   REINFORCEMENT= -21
X    0.13    -0.18  -0.28  -0.20
     0.37    -0.60  -0.84  -0.66
```

```
     0.42     -0.61   -0.89   -0.67
Z    0.19     -0.20   -0.32   -0.22
     0.31     -0.60   -0.84   -0.64

              -0.79   -0.97   -0.68
              -1.00    1.00    1.00
                DX      DY      DZ

OK



OK SHORT
OK 90 STEPS
C= 100    STD= 200
STEP NUMBER 11  POSITION = 9  13  5
STEP NUMBER 12  POSITION = 8  14  4
STEP NUMBER 13  POSITION = 7  15  3
STEP NUMBER 14  POSITION = 6  16  4
STEP NUMBER 15  POSITION = 7  17  5
STEP NUMBER 16  POSITION = 6  16  4
STEP NUMBER 17  POSITION = 7  15  3
STEP NUMBER 18  POSITION = 8  14  2
STEP NUMBER 19  POSITION = 7  13  1
STEP NUMBER 20  POSITION = 6  12  0
STEP NUMBER 21  POSITION = 5  11  -1
STEP NUMBER 22  POSITION = 4  10  -2
STEP NUMBER 23  POSITION = 3  9  -3
STEP NUMBER 24  POSITION = 2  8  -4
STEP NUMBER 25  POSITION = 3  7  -3
STEP NUMBER 26  POSITION = 4  6  -4
STEP NUMBER 27  POSITION = 3  5  -5
STEP NUMBER 28  POSITION = 4  4  -6
STEP NUMBER 29  POSITION = 3  5  -7
STEP NUMBER 30  POSITION = 4  4  -8
STEP NUMBER 31  POSITION = 3  3  -9
STEP NUMBER 32  POSITION = 2  2  -10
STEP NUMBER 33  POSITION = 3  1  -9
STEP NUMBER 34  POSITION = 4  0  -10
STEP NUMBER 35  POSITION = 3  -1  -9
STEP NUMBER 36  POSITION = 2  0  -10
STEP NUMBER 37  POSITION = 1  -1  -11
STEP NUMBER 38  POSITION = 0  -2  -10
STEP NUMBER 39  POSITION = -1  -3  -11
STEP NUMBER 40  POSITION = -2  -4  -12
STEP NUMBER 41  POSITION = -3  -3  -11
STEP NUMBER 42  POSITION = -2  -4  -12
STEP NUMBER 43  POSITION = -3  -5  -11
STEP NUMBER 44  POSITION = -4  -6  -12
STEP NUMBER 45  POSITION = -5  -7  -13
STEP NUMBER 46  POSITION = -4  -8  -14
STEP NUMBER 47  POSITION = -5  -9  -13
STEP NUMBER 48  POSITION = -6  -10  -12
STEP NUMBER 49  POSITION = -7  -11  -13
STEP NUMBER 50  POSITION = -8  -12  -12
STEP NUMBER 51  POSITION = -9  -13  -11
STEP NUMBER 52  POSITION = -10  -14  -12
STEP NUMBER 53  POSITION = -9  -15  -11
STEP NUMBER 54  POSITION = -10  -16  -10
STEP NUMBER 55  POSITION = -11  -17  -9
STEP NUMBER 56  POSITION = -12  -16  -8
STEP NUMBER 57  POSITION = -11  -17  -7
STEP NUMBER 58  POSITION = -12  -16  -6
STEP NUMBER 59  POSITION = -11  -17  -7
```

```
STEP NUMBER 61  POSITION = -13  -19  -9
STEP NUMBER 62  POSITION = -12  -18  -8
STEP NUMBER 63  POSITION = -11  -17  -9
STEP NUMBER 64  POSITION = -10  -16  -8
STEP NUMBER 65  POSITION = -9  -15  -9
STEP NUMBER 66  POSITION = -8  -14  -8
STEP NUMBER 67  POSITION = -7  -13  -7
STEP NUMBER 68  POSITION = -6  -12  -6
STEP NUMBER 69  POSITION = -5  -11  -5
STEP NUMBER 70  POSITION = -4  -10  -4
STEP NUMBER 71  POSITION = -3  -9  -3
STEP NUMBER 72  POSITION = -2  -8  -2
STEP NUMBER 73  POSITION = -1  -9  -3
STEP NUMBER 74  POSITION = 0  -8  -2
STEP NUMBER 75  POSITION = 1  -7  -1
STEP NUMBER 76  POSITION = 2  -6  0
STEP NUMBER 77  POSITION = 3  -5  -1
STEP NUMBER 78  POSITION = 4  -4  0
STEP NUMBER 79  POSITION = 5  -3  1
STEP NUMBER 80  POSITION = 6  -2  2
STEP NUMBER 81  POSITION = 7  -3  3
STEP NUMBER 82  POSITION = 8  -2  4
STEP NUMBER 83  POSITION = 9  -1  5
STEP NUMBER 84  POSITION = 10  0  6
STEP NUMBER 85  POSITION = 9  1  7
STEP NUMBER 86  POSITION = 10  0  8
STEP NUMBER 87  POSITION = 11  1  7
STEP NUMBER 88  POSITION = 10  2  6
STEP NUMBER 89  POSITION = 9  3  5
STEP NUMBER 90  POSITION = 10  4  4
STEP NUMBER 91  POSITION = 9  5  3
STEP NUMBER 92  POSITION = 10  6  4
STEP NUMBER 93  POSITION = 11  7  3
STEP NUMBER 94  POSITION = 12  8  2
STEP NUMBER 95  POSITION = 13  7  3
STEP NUMBER 96  POSITION = 12  8  4
STEP NUMBER 97  POSITION = 11  7  5
STEP NUMBER 98  POSITION = 12  6  6
STEP NUMBER 99  POSITION = 11  5  5
STEP NUMBER 100  POSITION = 10  4  6

OK VERY-SHORT
OK 900 STEPS
C= 100   STD= 200

STEP NUMBER 1000  POSITION 0  5  1
OK



OK SHOW-NET
X    0.26     6.42    0.86    0.43
     0.24    -6.19   -1.37    0.09
Y    0.17     2.34    6.86    0.72
     0.33    -1.69   -7.33   -0.05
Z    0.23    -0.08    0.13    7.71
     0.27     0.35   -0.52   -7.08

              0.05   -1.42    0.13
              1.00   -1.00   -1.00
               DX      DY      DZ
OK 1000 STEPS
C= 100   STD= 200
```

```
OK

OK SHOW-NET
X    0.25     8.33    1.06    0.22
     0.25    -8.29   -0.35   -0.79
Y    0.20     1.80    9.27   -0.01
     0.30    -1.13   -8.50   -0.40
Z    0.25     0.25    0.60    8.64
     0.25    -0.12    0.05   -9.11

              0.05   -0.35   -0.40
              1.00   -1.00    1.00
               DX      DY      DZ
OK

OK 0 C !  0 STD !
OK SHOW-PRMS
C= 0    STD= 0
OK 50 20 -70 MOVE-TO
OK SHORT
OK 100 STEPS
C= 0    STD= 0
STEP NUMBER 2001  POSITION = 49  19  -69
STEP NUMBER 2002  POSITION = 48  18  -68
STEP NUMBER 2003  POSITION = 47  17  -67
STEP NUMBER 2004  POSITION = 46  16  -66
STEP NUMBER 2005  POSITION = 45  15  -65
STEP NUMBER 2006  POSITION = 44  14  -64
STEP NUMBER 2007  POSITION = 43  13  -63
STEP NUMBER 2008  POSITION = 42  12  -62
STEP NUMBER 2009  POSITION = 41  11  -61
STEP NUMBER 2010  POSITION = 40  10  -60
STEP NUMBER 2011  POSITION = 39  9  -59
STEP NUMBER 2012  POSITION = 38  8  -58
STEP NUMBER 2013  POSITION = 37  7  -57
STEP NUMBER 2014  POSITION = 36  6  -56
STEP NUMBER 2015  POSITION = 35  5  -55
STEP NUMBER 2016  POSITION = 34  4  -54
STEP NUMBER 2017  POSITION = 33  3  -53
STEP NUMBER 2018  POSITION = 32  2  -52
STEP NUMBER 2019  POSITION = 31  3  -51
STEP NUMBER 2020  POSITION = 30  2  -50
STEP NUMBER 2021  POSITION = 29  3  -49
STEP NUMBER 2022  POSITION = 28  2  -48
STEP NUMBER 2023  POSITION = 27  3  -47
STEP NUMBER 2024  POSITION = 26  2  -46
STEP NUMBER 2025  POSITION = 25  3  -45
STEP NUMBER 2026  POSITION = 24  2  -44
STEP NUMBER 2027  POSITION = 23  3  -43
STEP NUMBER 2028  POSITION = 22  2  -42
STEP NUMBER 2029  POSITION = 21  3  -41
STEP NUMBER 2030  POSITION = 20  2  -40
STEP NUMBER 2031  POSITION = 19  3  -39
STEP NUMBER 2032  POSITION = 18  2  -38
STEP NUMBER 2033  POSITION = 17  3  -37
STEP NUMBER 2034  POSITION = 16  2  -36
STEP NUMBER 2035  POSITION = 15  3  -35
STEP NUMBER 2036  POSITION = 14  2  -34
STEP NUMBER 2037  POSITION = 13  3  -33
STEP NUMBER 2038  POSITION = 12  2  -32
STEP NUMBER 2039  POSITION = 11  3  -31
```

```
STEP NUMBER 2040  POSITION = 10  2  -30
STEP NUMBER 2041  POSITION = 9  3  -29
STEP NUMBER 2042  POSITION = 8  2  -28
STEP NUMBER 2043  POSITION = 7  3  -27
STEP NUMBER 2044  POSITION = 6  2  -26
STEP NUMBER 2045  POSITION = 5  3  -25
STEP NUMBER 2046  POSITION = 4  2  -24
STEP NUMBER 2047  POSITION = 3  3  -23
STEP NUMBER 2048  POSITION = 2  2  -22
STEP NUMBER 2049  POSITION = 1  3  -21
STEP NUMBER 2050  POSITION = 2  2  -20
STEP NUMBER 2051  POSITION = 1  3  -19
STEP NUMBER 2052  POSITION = 2  2  -18
STEP NUMBER 2053  POSITION = 1  3  -17
STEP NUMBER 2054  POSITION = 1  2  -16
STEP NUMBER 2055  POSITION = 2  3  -15
STEP NUMBER 2056  POSITION = 1  2  -14
STEP NUMBER 2057  POSITION = 2  3  -13
STEP NUMBER 2058  POSITION = 1  2  -12
STEP NUMBER 2059  POSITION = 2  3  -11
STEP NUMBER 2060  POSITION = 1  2  -10
STEP NUMBER 2061  POSITION = 2  3  -9
STEP NUMBER 2062  POSITION = 1  2  -8
STEP NUMBER 2063  POSITION = 2  3  -7
STEP NUMBER 2064  POSITION = 1  2  -6
STEP NUMBER 2065  POSITION = 2  3  -5
STEP NUMBER 2066  POSITION = 1  2  -4
STEP NUMBER 2067  POSITION = 1  3  -3
STEP NUMBER 2068  POSITION = 0  2  -2
STEP NUMBER 2069  POSITION = 1  3  -2
STEP NUMBER 2070  POSITION = 0  2  -3
STEP NUMBER 2071  POSITION = 1  3  -2
STEP NUMBER 2072  POSITION = 0  2  -3
STEP NUMBER 2073  POSITION = 1  3  -2
STEP NUMBER 2074  POSITION = 0  2  -3
STEP NUMBER 2075  POSITION = 1  3  -2
STEP NUMBER 2076  POSITION = 0  2  -3
STEP NUMBER 2077  POSITION = 1  3  -2
STEP NUMBER 2078  POSITION = 0  2  -3
STEP NUMBER 2079  POSITION = 1  3  -2
STEP NUMBER 2080  POSITION = 0  2  -3
STEP NUMBER 2081  POSITION = 1  3  -2
STEP NUMBER 2082  POSITION = 0  2  -3
STEP NUMBER 2083  POSITION = 1  3  -2
STEP NUMBER 2084  POSITION = 0  2  -3
STEP NUMBER 2085  POSITION = 1  3  -2
STEP NUMBER 2086  POSITION = 0  2  -3
STEP NUMBER 2087  POSITION = 1  3  -2
STEP NUMBER 2088  POSITION = 0  2  -3
STEP NUMBER 2089  POSITION = 1  3  -2
STEP NUMBER 2090  POSITION = 0  2  -3
STEP NUMBER 2091  POSITION = 1  3  -2
STEP NUMBER 2092  POSITION = 0  2  -3
STEP NUMBER 2093  POSITION = 1  3  -2
STEP NUMBER 2094  POSITION = 0  2  -3
STEP NUMBER 2095  POSITION = 1  3  -2
STEP NUMBER 2096  POSITION = 0  2  -3
STEP NUMBER 2097  POSITION = 1  3  -2
STEP NUMBER 2098  POSITION = 0  2  -3
STEP NUMBER 2099  POSITION = 1  3  -2
STEP NUMBER 2100  POSITION = 0  2  -3

OK -80 5 -65 MOVE-TO
OK 100 STEPS
C= 0    STD= 0
STEP NUMBER 2101  POSITION = -79  6  -64
```

```
STEP NUMBER 2103  POSITION = -77  8  -62
STEP NUMBER 2104  POSITION = -76  9  -61
STEP NUMBER 2105  POSITION = -75  8  -60
STEP NUMBER 2106  POSITION = -74  9  -59
STEP NUMBER 2107  POSITION = -73  8  -58
STEP NUMBER 2108  POSITION = -72  9  -57
STEP NUMBER 2109  POSITION = -71  8  -56
STEP NUMBER 2110  POSITION = -70  9  -55
STEP NUMBER 2111  POSITION = -69  8  -54
STEP NUMBER 2112  POSITION = -68  9  -53
STEP NUMBER 2113  POSITION = -67  8  -52
STEP NUMBER 2114  POSITION = -66  9  -51
STEP NUMBER 2115  POSITION = -65  8  -50
STEP NUMBER 2116  POSITION = -64  9  -49
STEP NUMBER 2117  POSITION = -63  8  -48
STEP NUMBER 2118  POSITION = -62  9  -47
STEP NUMBER 2119  POSITION = -61  8  -46
STEP NUMBER 2120  POSITION = -60  7  -45
STEP NUMBER 2121  POSITION = -59  8  -44
STEP NUMBER 2122  POSITION = -58  7  -43
STEP NUMBER 2123  POSITION = -57  8  -42
STEP NUMBER 2124  POSITION = -56  7  -41
STEP NUMBER 2125  POSITION = -55  8  -40
STEP NUMBER 2126  POSITION = -54  7  -39
STEP NUMBER 2127  POSITION = -53  8  -38
STEP NUMBER 2128  POSITION = -52  7  -37
STEP NUMBER 2129  POSITION = -51  8  -36
STEP NUMBER 2130  POSITION = -50  7  -35
STEP NUMBER 2131  POSITION = -49  6  -34
STEP NUMBER 2132  POSITION = -48  7  -33
STEP NUMBER 2133  POSITION = -47  6  -32
STEP NUMBER 2134  POSITION = -46  7  -31
STEP NUMBER 2135  POSITION = -45  6  -30
STEP NUMBER 2136  POSITION = -44  7  -29
STEP NUMBER 2137  POSITION = -43  6  -28
STEP NUMBER 2138  POSITION = -42  6  -27
STEP NUMBER 2139  POSITION = -41  5  -26
STEP NUMBER 2140  POSITION = -40  6  -25
STEP NUMBER 2141  POSITION = -39  5  -24
STEP NUMBER 2142  POSITION = -38  6  -23
STEP NUMBER 2143  POSITION = -37  5  -22
STEP NUMBER 2144  POSITION = -36  6  -21
STEP NUMBER 2145  POSITION = -35  5  -20
STEP NUMBER 2146  POSITION = -34  6  -19
STEP NUMBER 2147  POSITION = -33  5  -18
STEP NUMBER 2148  POSITION = -32  6  -17
STEP NUMBER 2149  POSITION = -31  5  -16
STEP NUMBER 2150  POSITION = -30  6  -15
STEP NUMBER 2151  POSITION = -29  5  -14
STEP NUMBER 2152  POSITION = -28  6  -13
STEP NUMBER 2153  POSITION = -27  5  -12
STEP NUMBER 2154  POSITION = -26  6  -11
STEP NUMBER 2155  POSITION = -25  5  -10
STEP NUMBER 2156  POSITION = -24  4  -9
STEP NUMBER 2157  POSITION = -23  5  -8
STEP NUMBER 2158  POSITION = -22  4  -7
STEP NUMBER 2159  POSITION = -21  5  -6
STEP NUMBER 2160  POSITION = -20  4  -5
STEP NUMBER 2161  POSITION = -19  5  -4
STEP NUMBER 2162  POSITION = -18  4  -3
STEP NUMBER 2163  POSITION = -17  3  -2
STEP NUMBER 2164  POSITION = -16  4  -1
STEP NUMBER 2165  POSITION = -15  3  0
STEP NUMBER 2166  POSITION = -14  4  -1
STEP NUMBER 2167  POSITION = -13  3  -2
```

```
STEP NUMBER 2169  POSITION = -11  3  -2
STEP NUMBER 2170  POSITION = -10  4  -1
STEP NUMBER 2171  POSITION = -9  3  -2
STEP NUMBER 2172  POSITION = -8  3  -1
STEP NUMBER 2173  POSITION = -7  2  -2
STEP NUMBER 2174  POSITION = -6  3  -1
STEP NUMBER 2175  POSITION = -5  2  -2
STEP NUMBER 2176  POSITION = -4  3  -1
STEP NUMBER 2177  POSITION = -3  2  -2
STEP NUMBER 2178  POSITION = -2  3  -1
STEP NUMBER 2179  POSITION = -1  2  -2
STEP NUMBER 2180  POSITION = 0  3  -1
STEP NUMBER 2181  POSITION = 1  2  -2
STEP NUMBER 2182  POSITION = 0  3  -3
STEP NUMBER 2183  POSITION = 1  2  -2
STEP NUMBER 2184  POSITION = 0  3  -3
STEP NUMBER 2185  POSITION = 1  2  -2
STEP NUMBER 2186  POSITION = 0  3  -3
STEP NUMBER 2187  POSITION = 1  2  -2
STEP NUMBER 2188  POSITION = 0  3  -3
STEP NUMBER 2189  POSITION = 1  2  -2
STEP NUMBER 2190  POSITION = 0  3  -3
STEP NUMBER 2191  POSITION = 1  2  -2
STEP NUMBER 2192  POSITION = 0  3  -3
STEP NUMBER 2193  POSITION = 1  2  -2
STEP NUMBER 2194  POSITION = 0  3  -3
STEP NUMBER 2195  POSITION = 1  2  -2
STEP NUMBER 2196  POSITION = 0  3  -3
STEP NUMBER 2197  POSITION = 1  2  -2
STEP NUMBER 2198  POSITION = 0  3  -3
STEP NUMBER 2199  POSITION = 1  2  -2
STEP NUMBER 2200  POSITION = 0  3  -3

OK SHORT
OK 10 70 -55 MOVE-TO
OK 100 STEPS
C= 0    STD= 0
STEP NUMBER 2201  POSITION = 9  69  -54
STEP NUMBER 2202  POSITION = 8  68  -53
STEP NUMBER 2203  POSITION = 7  67  -52
STEP NUMBER 2204  POSITION = 6  66  -51
STEP NUMBER 2205  POSITION = 5  65  -50
STEP NUMBER 2206  POSITION = 4  64  -49
STEP NUMBER 2207  POSITION = 3  63  -48
STEP NUMBER 2208  POSITION = 2  62  -47
STEP NUMBER 2209  POSITION = 1  61  -46
STEP NUMBER 2210  POSITION = 0  60  -45
STEP NUMBER 2211  POSITION = -1  59  -44
STEP NUMBER 2212  POSITION = -2  58  -43
STEP NUMBER 2213  POSITION = -3  57  -42
STEP NUMBER 2214  POSITION = -4  56  -41
STEP NUMBER 2215  POSITION = -5  55  -40
STEP NUMBER 2216  POSITION = -4  54  -39
STEP NUMBER 2217  POSITION = -5  53  -38
STEP NUMBER 2218  POSITION = -4  52  -37
STEP NUMBER 2219  POSITION = -5  51  -36
STEP NUMBER 2220  POSITION = -4  50  -35
STEP NUMBER 2221  POSITION = -5  49  -34
STEP NUMBER 2222  POSITION = -4  48  -33
STEP NUMBER 2223  POSITION = -3  47  -32
STEP NUMBER 2224  POSITION = -4  46  -31
STEP NUMBER 2225  POSITION = -3  45  -30
STEP NUMBER 2226  POSITION = -4  44  -29
STEP NUMBER 2227  POSITION = -3  43  -28
STEP NUMBER 2228  POSITION = -4  42  -27
```

```
STEP NUMBER 2230  POSITION = -4  40  -25
STEP NUMBER 2231  POSITION = -3  39  -24
STEP NUMBER 2232  POSITION = -4  38  -23
STEP NUMBER 2233  POSITION = -3  37  -22
STEP NUMBER 2234  POSITION = -4  36  -21
STEP NUMBER 2235  POSITION = -3  35  -20
STEP NUMBER 2236  POSITION = -4  34  -19
STEP NUMBER 2237  POSITION = -3  33  -18
STEP NUMBER 2238  POSITION = -2  32  -17
STEP NUMBER 2239  POSITION = -3  31  -16
STEP NUMBER 2240  POSITION = -2  30  -15
STEP NUMBER 2241  POSITION = -3  29  -14
STEP NUMBER 2242  POSITION = -2  28  -13
STEP NUMBER 2243  POSITION = -3  27  -12
STEP NUMBER 2244  POSITION = -2  26  -11
STEP NUMBER 2245  POSITION = -3  25  -10
STEP NUMBER 2246  POSITION = -2  24  -9
STEP NUMBER 2247  POSITION = -3  23  -8
STEP NUMBER 2248  POSITION = -2  22  -7
STEP NUMBER 2249  POSITION = -3  21  -6
STEP NUMBER 2250  POSITION = -2  20  -5
STEP NUMBER 2251  POSITION = -3  19  -4
STEP NUMBER 2252  POSITION = -2  18  -3
STEP NUMBER 2253  POSITION = -3  17  -2
STEP NUMBER 2254  POSITION = -2  16  -3
STEP NUMBER 2255  POSITION = -3  15  -2
STEP NUMBER 2256  POSITION = -2  14  -3
STEP NUMBER 2257  POSITION = -1  13  -2
STEP NUMBER 2258  POSITION = -2  12  -3
STEP NUMBER 2259  POSITION = -1  11  -2
STEP NUMBER 2260  POSITION = -2  10  -3
STEP NUMBER 2261  POSITION = -1  9  -2
STEP NUMBER 2262  POSITION = 0  8  -3
STEP NUMBER 2263  POSITION = -1  7  -2
STEP NUMBER 2264  POSITION = 0  6  -3
STEP NUMBER 2265  POSITION = -1  5  -2
STEP NUMBER 2266  POSITION = 0  4  -3
STEP NUMBER 2267  POSITION = 1  3  -2
STEP NUMBER 2268  POSITION = 0  2  -3
STEP NUMBER 2269  POSITION = 1  3  -2
STEP NUMBER 2270  POSITION = 0  2  -3
STEP NUMBER 2271  POSITION = 1  3  -2
STEP NUMBER 2272  POSITION = 0  2  -3
STEP NUMBER 2273  POSITION = 1  3  -2
STEP NUMBER 2274  POSITION = 0  2  -3
STEP NUMBER 2275  POSITION = 1  3  -2
STEP NUMBER 2276  POSITION = 0  2  -3
STEP NUMBER 2277  POSITION = 1  3  -2
STEP NUMBER 2278  POSITION = 0  2  -3
STEP NUMBER 2279  POSITION = 1  3  -2
STEP NUMBER 2280  POSITION = 0  2  -3
STEP NUMBER 2281  POSITION = 1  3  -2
STEP NUMBER 2282  POSITION = 0  2  -3
STEP NUMBER 2283  POSITION = 1  3  -2
STEP NUMBER 2284  POSITION = 0  2  -3
STEP NUMBER 2285  POSITION = 1  3  -2
STEP NUMBER 2286  POSITION = 0  2  -3
STEP NUMBER 2287  POSITION = 1  3  -2
STEP NUMBER 2288  POSITION = 0  2  -3
STEP NUMBER 2289  POSITION = 1  3  -2
STEP NUMBER 2290  POSITION = 0  2  -3
STEP NUMBER 2291  POSITION = 1  3  -2
STEP NUMBER 2292  POSITION = 0  2  -3
STEP NUMBER 2293  POSITION = 1  3  -2
STEP NUMBER 2294  POSITION = 0  2  -3
```

```
STEP NUMBER 2296  POSITION = 0  2  -3
STEP NUMBER 2297  POSITION = 1  3  -2
STEP NUMBER 2298  POSITION = 0  2  -3
STEP NUMBER 2299  POSITION = 1  3  -2
STEP NUMBER 2300  POSITION = 0  2  -3

OK VERY-SHORT
OK 60 60 60 MOVE-TO
OK 100 STEPS
C= 0   STD= 0

STEP NUMBER 2400  POSITION 0  2  -3
OK 70 -2 -55 MOVE-TO
OK 100 STEPS
C= 0   STD= 0

STEP NUMBER 2500  POSITION 0  3  -3
OK
```

APPENDIX B

```
PDP-11 Forth-79 Standard System Rev 2.8                  1-JAN-80   00:00:00


BLK 300  ( VARIABLE DEFINITIONS )
BLK 301  ( STORING AND RETRIEVING FROM MAIN ARRAYS )
BLK 302  ( *E: MULTIPLYING FIXED POINT NUMBERS WITH EXPONENTS )
BLK 303  ( RANDOM NUMBER GENERATOR:  GAUSS AND CHOOSE )
BLK 304  ( .NROW: TYPES NUMBERS FROM ADR1 TO ADR2 HAVING EXP )
BLK 305  ( CALC-ACT: COMPUTES THE NETWORK'S OUTPUT )
BLK 306  ( LEARN-NET: UPDATES THE NETWORK'S WEIGHTS)
BLK 307
BLK 308  ( TRACES: UPDATES TIME TRACES OF NETWORK VARIABLES )
BLK 309
BLK 310  ( SHOW-NET, SHOW-STATE, SHOWPOS )
BLK 311  ( SHOW-PRMS )
BLK 312
BLK 313  ( SENSE, ACT, AND EVAL:  MACHINE DEPENDENT STUFF )
BLK 314  ( MOVE-TO: MACHINE DEPENDENT,  !GOAL )
BLK 315
BLK 316  ( ZERO-NET )
BLK 317  ( TOP LEVEL WORDS:  STEPS, INIT, LONG, SHORT )
BLK 318  ( TEST:  RUNS FOR 100 STEPS, NO LEARNING, TO FIND EQUILIBRIUM )
BLK 319  ( HELP:  LISTS TOP LEVEL WORDS )


*****************  BLOCK    300     ******************

( VARIABLE DEFINITIONS )
0 CON ALL 3 CON NUM-CELLS  6 CON NUM-INPUTS

100 VAR C  200 VAR STD  0 VAR P  0 VAR POLD  0 VAR DEBUG
0 VAR NSTEPS  0 VAR R  1 VAR LONGV

NUM-CELLS NUM-INPUTS * 2* ARRAY W
NUM-CELLS 2* ARRAY S  NUM-CELLS 2* ARRAY Y
NUM-CELLS 2* ARRAY YOLD  NUM-INPUTS 2* ARRAY X
NUM-INPUTS 2* ARRAY XOLD
-2 VAR WEXP  -2 VAR XEXP  -2 VAR SEXP  -2 VAR YEXP
-4 VAR CEXP  0 VAR PEXP

6 ARRAY POS  : @POS 2* POS + @ ;  : !POS 2* POS + ! ;

280 LOAD                                          -->


*****************  BLOCK    301     ******************

( STORING AND RETRIEVING FROM MAIN ARRAYS )
: @W  NUM-CELLS * + 2* W + @ ;
: !W  NUM-CELLS * + 2* W + ! ;
: @S  2* S + @ ;  : !S  2* S + ! ;
: @Y  2* Y + @ ;  : !Y  2* Y + ! ;
: @YOLD  2* YOLD + @ ;  : !YOLD  2* YOLD + ! ;
: @X  2* X + @ ;  : !X  2* X + ! ;
: @XOLD  2* XOLD + @ ;  : !XOLD  2* XOLD + ! ;
: @LONG  LONGV @ 0> ;
```

```
                                                    -->


*****************  BLOCK    302     ******************

( *E: MULTIPLYING FIXED POINT NUMBERS WITH EXPONENTS )
12 ARRAY ARGS
: TYPE5 DEBUG @ IF CR
        5 0 DO ARGS I 2* + ! LOOP 0 4 DO ARGS I 2* + ? -1 +LOOP
           0 4 DO ARGS I 2* + @ -1 +LOOP    THEN ;

: 10^ 1 SWAP 0 DO 10 * LOOP ;
: *E SWAP ROT + - DUP 0> IF 10^ */ 1 THEN
                  DUP 0= IF DROP * 0 THEN
                  DUP 0< IF ABS 10^ * * 1 THEN DROP ;




                                                    -->


*****************  BLOCK    303     ******************

( RANDOM NUMBER GENERATOR:  GAUSS AND CHOOSE )
HERE VAR RND
: RANDOM RND @ 31421 * 6927 + DUP RND ! ;
: CHOOSE RANDOM 0 ROT D* SWAP DROP ;
: DIST 0 8 0 DO 100 CHOOSE + LOOP ;
: GAUSS DIST 400 - 122 100 */ STD @ 100 */ ;
(   : GAUSS STD @ 2* CHOOSE STD @ - ; )






                                                    -->


*****************  BLOCK    304     ******************

( .NROW: TYPES NUMBERS FROM ADR1 TO ADR2 HAVING EXP )
( IN FIELD OF COLS COLUMNS (ADR2 ADR1 EXP COLS .NROW)

0 VAR NCOLS    0 VAR EXP
: SETUP <# ;
 ( DNUM EXP -- )
: -EXP EXP @ DUP 0< IF ABS 0 DO # LOOP
        46 HOLD 0 THEN DROP ;
 ( EXP -- )
: +EXP EXP @ DUP 0> IF 0 DO 48 HOLD LOOP 0 THEN DROP ;
: FINISH #S SIGN #> ;

: MAKEDN DUP ABS 0 ;
: .N ( SWAP OVER DABS ) ( ASSUMES EXP AND COLS ARE ASSIGNED )
MAKEDN SETUP -EXP +EXP FINISH DUP NCOLS @ SWAP - SPACES TYPE ;
: .NROW NCOLS ! EXP ! DO I @ .N 2 +LOOP ;          -->
```

```
******************  BLOCK    305     ******************

( CALC-ACT: COMPUTES THE NETWORK'S OUTPUT )
0 VAR NC   : @NC NC @ ;
: D1 DEBUG @ IF ." = " DUP . THEN ;
: CELLNC   NC !   0 @NC !S
0  NUM-INPUTS 0 DO  @NC I @W   I @X  WEXP @ XEXP @ SEXP @ TYPE5
*E D1
 + LOOP D1 DUP @NC !S GAUSS + D1
 DUP 0>  IF YEXP @ ABS 10^ @NC !Y THEN
 DUP 0=  IF 0 @NC !Y THEN
     0<  IF YEXP @ ABS 10^ NEGATE @NC !Y THEN ;

: CALC-ACT NUM-CELLS 0 DO I CELLNC LOOP   @LONG IF
." NETWORK GENERATED THE ACTION (DX,DY,DZ)= " Y DUP NUM-CELLS
2* + SWAP YEXP @ 5 .NROW CR THEN ;

                                                    -->


******************  BLOCK    306     ******************

( LEARN-NET: UPDATES THE NETWORK'S WEIGHTS)
: REINF POLD @ P @ - R ! ;
( : EXPY @S DUP 100 > IF DROP 100 ELSE DUP -100 < IF DROP -100
 THEN THEN ; )
: UPDATE-WS  NUM-CELLS 0 DO  NUM-INPUTS 0 DO
 I @X J @Y XEXP @ YEXP @ WEXP @ TYPE5 *E D1
( J @Y J EXPY - I @X YEXP @ XEXP @ WEXP @ *E )
 R @ WEXP @ PEXP @ WEXP @ TYPE5 *E D1
 C @ WEXP @ CEXP @ WEXP @ TYPE5 *E D1
 J I @W + D1 J I !W                   LOOP  LOOP ;

: LEARN-NET
  REINF UPDATE-WS  @LONG IF ." NETWORK WEIGHTS "
." UPDATED USING REINFORCEMENT OF " R ? CR THEN ;

                                                    -->


******************  BLOCK    307     ******************




                                                    -->


******************  BLOCK    308     ******************

( TRACES: UPDATES TIME TRACES OF NETWORK VARIABLES )
: TRACES
```

```
( NUM-CELLS 0 DO I @Y I !YOLD LOOP )
              P @ POLD ! ;




                                                  -->


*****************  BLOCK    309     ******************




                                                  -->


*****************  BLOCK    310     ******************

( SHOW-NET, SHOW-STATE, SHOWPOS )
7 CON COLUMNS
: INP-LABEL DUP 0 = IF ." X" ELSE DUP 2 = IF ." Y" ELSE
 DUP 4 = IF ." Z" ELSE ."   " THEN THEN THEN DROP ;
: SHOW-NET  NUM-INPUTS 0 DO
 I INP-LABEL X I 2* + DUP XEXP @ COLUMNS .NROW  2 SPACES
 W I NUM-CELLS * 2* + DUP NUM-CELLS 2* + SWAP WEXP @ COLUMNS
    .NROW CR LOOP CR   COLUMNS 3 + SPACES
S DUP NUM-CELLS 2* + SWAP SEXP @ COLUMNS .NROW CR
 COLUMNS 3 + SPACES Y DUP NUM-CELLS 2* + SWAP YEXP @ COLUMNS
.NROW CR COLUMNS 3 + SPACES COLUMNS 3 - SPACES ." DX" COLUMNS 2-
 SPACES ." DY" COLUMNS 2- SPACES ." DZ" CR ;
: SHOWPOS 3 0 DO I @POS . LOOP ;
: SHOW-STATE LONGV @ 0< IF          ELSE ." STEP NUMBER " NSTEPS
 ? ." POSITION = " SHOWPOS CR @LONG IF ." PAYOFF= " P ?
."  OLD PAYOFF= " POLD ? ."  REINFORCEMENT= " R ? CR        -->


*****************  BLOCK    311     ******************

( SHOW-PRMS )
  ( SHOW-STATE CONTINUED )  SHOW-NET THEN THEN ;
: SHOW-PRMS  ." C= " C ? ."  STD= " STD ? CR ;
```

-->

****************** BLOCK 312 ******************

-->

****************** BLOCK 313 ******************

```
( SENSE, ACT, AND EVAL:  MACHINE DEPENDENT STUFF )
: CLAMP  DUP 0< IF DROP 0 ELSE DUP 100 > IF DROP 100 THEN THEN ;
 100 CON RANGE ( OF POSITION )
: SENSE  3 0 DO I @POS DUP  -100 75       */ 25 + CLAMP I 2* !X
 100 75       */ 25 + CLAMP I 2* 1+ !X LOOP @LONG
IF ." SENSES RETURNED POSITION " SHOWPOS CR THEN ;
1 VAR STEPSIZE
: ACT @LONG IF ." ARM MOVED FROM POSITION " SHOWPOS ." TO " THEN
 3 0 DO I @POS  I @Y 1 YEXP @ 0 0 *E STEPSIZE @ * + I !POS LOOP
@LONG IF SHOWPOS CR THEN ;

6 ARRAY GOAL : @GOAL 2* GOAL + @ ;
: EVAL  0 3 0 DO I @POS I @GOAL - DUP 1 */  + LOOP  P ! @LONG IF
 ." DISTANCE ENCODER RETURNED " P ? CR THEN ;
```

-->

****************** BLOCK 314 ******************

```
( MOVE-TO: MACHINE DEPENDENT,  !GOAL )
: MOVE-TO  2 !POS  1 !POS  0 !POS  EVAL  TRACES ;
: !GOAL 4 GOAL + !  2 GOAL + !  GOAL ! ;
```

```
*****************  BLOCK    315     ******************

                                                    -->

*****************  BLOCK    316     ******************

( ZERO-NET )
: ZERO DO I OSET LOOP ;
: ZERO-NET  W DUP NUM-CELLS NUM-INPUTS * 2* + SWAP ZERO
 X DUP NUM-INPUTS 2* + SWAP ZERO  S DUP NUM-CELLS 2* + SWAP ZERO
 Y DUP NUM-CELLS 2* + SWAP ZERO  YOLD DUP NUM-CELLS 2* + SWAP
  ZERO  0 NSTEPS ! ;

                                                    -->

*****************  BLOCK    317     ******************

( TOP LEVEL WORDS:  STEPS, INIT, LONG, SHORT )
: STEPS   SHOW-PRMS   0 DO  NSTEPS 1+!
 SENSE ( X AND P ARE ASSIGNED FROM MANIPULATOR )
 CALC-ACT ( CALCULATES S AND Y )
 ACT    ( APPLIES Y TO MANIPULATOR )
 EVAL   ( GET PAYOFF)
 LEARN-NET ( W IS UPDATED)
 SHOW-STATE ( TYPES CURRENT STATE OF NETWORK)
 TRACES ( UPDATES TRACES )
               LOOP CR
 LONGV @ 0< IF ." STEP NUMBER " NSTEPS ? ." POSITION " SHOWPOS
 CR THEN ;
: INIT  0 0 0 !GOAL  ." GOAL IS 0 0 0 " CR  ZERO-NET EVAL TRACES
     SHOW-PRMS SHOW-STATE ;
: LONG 1 LONGV ! ;   : SHORT  0 LONGV ! ; : VERY-SHORT -1 LONGV !
 ;                                                  -->

*****************  BLOCK    318     ******************
```

```
0 VAR STD-TEMP
: TEST  STD @ STD-TEMP ! 0 STD !
VERY-SHORT 95 0 DO  SENSE CALC-ACT ACT  LOOP
SHORT 5 0 DO  SENSE CALC-ACT ACT SHOW-STATE  LOOP
STD-TEMP @ STD ! ;
```

-->

```
*****************  BLOCK    319     ******************

( HELP:  LISTS TOP LEVEL WORDS )
: HELP CR 10 SPACES ." AVAILABLE COMMANDS:" CR
." MOVE-TO    MOVES ARM TO GIVEN POSITION (X Y Z MOVE-TO) " CR
." INIT       RESETS THE NETWORK VARS TO ZERO (INIT) " CR
." STEPS      RUNS THE EXPERIMENT FOR N STEPS (N STEPS) " CR
." LONG       SHOW ALL DETAILS OF EACH STEP (LONG) " CR
." SHORT      JUST SHOW POSITION AT EACH STEP (SHORT) " CR
." VERY-SHORT SHOW NOTHING AT EACH STEP (VERY-SHORT) " CR
." SHOW-NET   DISPLAYS CURRENT STATE OF NET (SHOW-NET) " CR
." TEST       FIND THE EQUILIBRIUM POSITION OF THE NET (TEST) "
CR CR ;
```

;S