Durham University

Durham E-Theses

Stochastic arrays and learning networks

Richard A. Leaver

How to cite:

Leaver, Richard A. (1988) Stochastic arrays and learning networks. Doctoral thesis, Durham University.

Use policy

The full-text may be used and/or reproduced, and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not-for-profit purposes provided that:

- a full bibliographic reference is made to the original source
- a https://etheses.durham.ac.uk/id/eprint/6706/ is made to the metadata record in Durham E-Theses
- the full-text is not changed in any way

The full-text must not be sold in any format or medium without the formal permission of the copyright holders.

Please consult the full Durham E-Theses policy for further details.

Academic Support Office, The Palatine Centre, Durham University, Stockton Road, Durham, DH1 3LE
e-mail: e-theses.admin@durham.ac.uk Tel: +44 0191 334 6107
https://etheses.durham.ac.uk

Stochastic Arrays and Learning Networks

Two Volumes - Volume Two

Richard A. Leaver

B.Sc. (Dunelm), M.I.E.E.E, A.M.I.E.E.

The copyright of this thesis rests with the author.
No quotation from it should be published without
his prior written consent and information derived
from it should be acknowledged.

School of Engineering and Applied Science

University of Durham

August 1988

A Thesis submitted for the degree of

Doctor of Philosophy of the University of Durham.

- 6 JUL 1989

Volume Two

Table of Contents

Simulation programs for Appendix 2 – Stochastic Systolic Array

FORTRAN77 program ARRAY.MAIN

– main array simulation

```
C     ***********************************************
C     *                                             *
C     *   Systolic Array Production Program         *
C     *                                             *
C     * - This program is the Double Precision      *
C     *   Version of the finalised prototype        *
C     *   simulation program for running extensive  *
C     *   tests on the stochastic systolic cells.   *
C     * - This uses the repeatable initial setting  *
C     *   for the random number generator enabling  *
C     *   comparisons to be made between successive *
C     *   runs.                                     *
C     * - The outputs are fed into a simulated shift *
C     *   register SHIFT so that the Moving Average *
C     *   can be tested.                            *
C     * - A Gaussian noise component has been       *
C     *   introduced on the x input data.           *
C     *   The NAG routine G05DDF is used.           *
C     *   The Standard Deviation of this is input   *
C     *   as a dB value to the program.             *
C     *   The dB value is defined as noise standard *
C     *   deviation to maximum input scaled signal. *
C     * - this program performs a Cyclic Correlation *
C     *   operation using stochastic arithmetic via *
C     *   a N x N systolic array simulation.        *
C     * - The simulation is for complex positive    *
C     *   or negative normalised values.            *
C     * - The program converts the x and h data     *
C     *   & coefficients to independent pn sequences *
C     *   representing these values using a two line *
C     *   bipolar representation.                   *
C     * - NB the values should have been normalised *
C     *   so that the greatest does not exceed      *
C     *   +/- one.                                  *
C     * - The program outputs calculated values     *
C     *   every clock cycle to show increasing      *
C     *   precision.                                *
C     *   On completion, the full result is outputted*
C     *   on channel 4.                             *
C     * - The scale is entered manually and should  *
C     *   be at least the magnitude of the peak of  *
C     *   autocorrelation function of the reference. *
C     * - There is a wide choice of possible cells  *
C     *   to be simulated, all are in the CELL.SUB  *
C     *   format all using special COMMON blocks    *
C     *   of memory.                                *
C     * - The correlation operation is defined as   *
C     *   a summation E x.y (with appropriate       *
C     *   subscripts). If x= (a+ib) and y= (c+id)   *
C     *   then the four correlations required are ; *
C     *   E (ac) - E (bd) + iE (bc) + iE (ad)       *
C     * - For a purely real sequence, b=d=0 and we  *
C     *   are left with E (ac) as before.           *
C     *   We therefore calculate one complex        *
C     *   correlation as four real correlations.    *
C     * - The program performs this by multiplexing *
C     *   NODE NOW and NEXT values for each of the  *
C     *   four cases so that at any clock cycle,    *
C     *   the full complex result may be examined.  *
C     * - ISLOT is the effective length of the      *
C     *   shift register and is entered manually;   *
C     * - ISELEC selects the mode of averaging.     *
C     *   ISELEC=0          - Standard Averaging    *
C     *   ISELEC=1          - Moving Average        *
C     * - Functions called are;-                    *
```


```
C     *                                                *
C     *   INIT in INIT.SUB  (Non repeatable)           *
C     *   READER in READER.SUB                         *
C     *   RAND in RAND.SUB  (NB Double Precision)      *
C     *   MAP1 in MAPPER.SUB                           *
C     *   *NAG library                                 *
C     *   CELL in CELL.SUB                             *
C     *   OUTCNT in OUTCNT.SUB                         *
C     *   AVERAG in AVERAG.SUB                         *
C     *                                                *
C     * - Maximum N is 100.                            *
C     *                                                *
C     * Author: Richard Leaver                         *
C     * Created:14th November 1986                     *
C     * Update :18th February 1988                     *
C     * Frozen : 5th May 1988                          *
C     **************************************************
      PROGRAM ARRAY
C
C     Define node values arrays -
C     We define node-now and node-next representing the values
C     of the node for the current round of computations and those
C     values of the node as computed/inputted at this time to be
C     used for the next round of computation;
C     The ,2 dimension is because we are using one line for positive and
C     one line for negative representation.
C
C     The array sizes are defined as H*(N*(N+1),4,2),
C     Y*((N+1)*N,4,2), X*((N+1)*(N+N-1),4,2)
C
      INTEGER HNOW(10100,4,2), HNEXT(10100,4,2)
      INTEGER XNOW(20099,4,2), XNEXT(20099,4,2)
      INTEGER YNOW(10100,4,2), YNEXT(10100,4,2), COUNT(10000,4,2)
C
C     Set aside space for the input and reference data ;
C
      DOUBLE PRECISION XREAL(0:95), XIMAG(0:95), HREAL(0:95),
     &       HIMAG(0:95)
C
C     Define space for the arrays which will form the data and reference
C     processing arrays for the four simultaneous correlations ;
C
      DOUBLE PRECISION TXREAL(0:95,4), THREAL(0:95,4)
C
C     Now define array mapping;
C     This is MAP(no of processors,6) where the 6 represents six
C     addresses
C     in the MAP array -
C     1=HIN 2=HOUT 3=XIN 4=XOUT 5=YIN 6=YOUT
C
      INTEGER MAP(10000,6)
C
C     Define integer blocks for the I/O node values ;
C
      INTEGER HNODE(96), XNODE(191), YNODE(96)
C
C     Define integer array for the output averaging of the four
C     correlations ;
C
      INTEGER OUT(96,4,2)
C
C     Define Shift Register for the storing of output bit streams;
C
```

```
      INTEGER SHIFT(10000,6,4,2)
C
C     Define Random Number PN and Clock and Scale variables
C     as Double Precision ;
C
      DOUBLE PRECISION PN, CLOCK, CLOCK1, SCALE, SCALE1
C
C     Define Noisy data working variable
C     as Double Precision ;
C
      DOUBLE PRECISION XNOISE, STADEV
C
C     Define Integer Variables ;
C
      INTEGER HIN, HOUT, XIN, XOUT, YIN, YOUT, HNUM, XNUM, YNUM
      INTEGER HIN1, HOUT1, XIN1, XOUT1, YIN1, YOUT1
      INTEGER TIME, PROC, MXTIME, CORRNO
C
C     Define standard COMMON block;
C
      COMMON COUNT, OUT, HNOW, HNEXT, YNOW, YNEXT, XNOW, XNEXT, NPOINT
C
C     Define INTERF named  COMMON block;
C
      COMMON /INTERF/ SHIFT, ISLOT, ISELEC
C
C     Read Input values - set ICOM=1 to OPEN files ;
C
      ICOM = 1
      CALL READER(ICOM, XREAL, XIMAG, HREAL, HIMAG)
C
C     Place data in mapping array - see page 150 notes;
C
      CALL MAP1(HNUM, XNUM, YNUM, MAP, HNODE, XNODE, YNODE)
C
C     Initialise Random Number Generator;
C
      CALL INIT
      PN = 1.0D0
C
C     Read in register length
C
      WRITE (6,*)'Input Window Length'
      READ (5,*) ISLOT
C
C     Read in mode of averaging
C
      WRITE (6,*)'Input Averaging Mode'
      READ (5,*) ISELEC
C
C     Read in Standard Deviation of noise component (dB)
C
      WRITE (6,*)'Input Standard Deviation of Noise (dB)'
      READ (5,*) DBHISS
C
C     The Probability Distribution Function for the NAG routine G05DDF
C     is:
C
C     P(X) = 1/ (SQRT(2 PI) B) EXP {- (X-A)^2 / 2 B^2 }
C
C     Where A = Mean and B = Standard Deviation
C
C     Define Ratio DBHISS relative to maximum input scale
```

```
C
C      DBHISS = -20 LOG10 ((1/SQRT(SCALE)) / B)
C
C      Read in Maximum Time
C
       WRITE (6,*)'Input Max Time'
       READ (5,*) MXTIME
C
C      Scale values ;
C
       WRITE (6,*)'Input Scale'
       READ (5,*) SCALE1
       SCALE = DSQRT(SCALE1)
C
C      Compute Standard Deviation of noise;
C
       STADEV = (1.0/SCALE) * 10 ** (DBHISS/20.0)
C
       DO 1 I = 0, (NPOINT - 1)
         XREAL(I) = XREAL(I) / SCALE
         XIMAG(I) = XIMAG(I) / SCALE
         HREAL(I) = HREAL(I) / SCALE
         HIMAG(I) = HIMAG(I) / SCALE
     1 CONTINUE
C
C      Place the data values into the TXREAL and THREAL arrays;
C      1= XREAL & HREAL
C      2= XIMAG & HIMAG
C      3= XIMAG & HREAL
C      4= XREAL & HIMAG
       DO 2 I = 0, NPOINT - 1
         TXREAL(I,1) = XREAL(I)
         THREAL(I,1) = HREAL(I)
         TXREAL(I,2) = XIMAG(I)
         THREAL(I,2) = HIMAG(I)
         TXREAL(I,3) = XIMAG(I)
         THREAL(I,3) = HREAL(I)
         TXREAL(I,4) = XREAL(I)
         THREAL(I,4) = HIMAG(I)
     2 CONTINUE
C
C      Define CLOCK start value;
C
       CLOCK = 1.0
C
C      Begin loop calculating each processors contribution at each time
C      interval
C      t where t=1,MXTIME  and no processors = 1,NPOINT**2 ;
C      This is carried out over the four correlations ;
C      The array averaging is only started after the array is 'flushed'
C      with data, this prevents a slight start up error in the averaging
C      process ;
C
       DO 12 TIME = 1, MXTIME + NPOINT
         IF (TIME .GT. NPOINT) CLOCK = DFLOAT(TIME) - NPOINT
         WRITE (6,*) CLOCK, TIME
C
C      Multiplex correlations in turn ;
C
         DO 10 CORRNO = 1, 4
C
C      Encode input terms into pn sequences as required and perform
C      computation ;
C
C      Define the ,1 dimension as positive values and the ,2 dimension as
C      negative values ;
```

```
C
C     Use XNOISE as working variable containing the Gaussian noise
C     data;
C
          J = 0
C
          DO 3 I = 1, XNUM
            CALL RAND(PN)
            XNOISE = G05DDF(TXREAL(J,CORRNO),STADEV)
            IF (XNOISE .GE. 0.0D0) THEN
              IF (PN .LT. XNOISE) THEN
                XNOW(XNODE(I),CORRNO,1) = 1
              ELSE
                XNOW(XNODE(I),CORRNO,1) = 0
              END IF
            ELSE
              IF (PN .LT. (-1.0D0*XNOISE)) THEN
                XNOW(XNODE(I),CORRNO,2) = 1
              ELSE
                XNOW(XNODE(I),CORRNO,2) = 0
              END IF
            END IF
            J = J + 1
            IF (J .EQ. NPOINT) J = 0
    3     CONTINUE
C
          DO 4 J = 0, HNUM - 1
            I = J + 1
            CALL RAND(PN)
            IF (THREAL(J,CORRNO) .GE. 0.0D0) THEN
              IF (PN .LT. THREAL(J,CORRNO)) THEN
                HNOW(HNODE(I),CORRNO,1) = 1
              ELSE
                HNOW(HNODE(I),CORRNO,1) = 0
              END IF
            ELSE
              IF (PN .LT. (-1.0D0*THREAL(J,CORRNO))) THEN
                HNOW(HNODE(I),CORRNO,2) = 1
              ELSE
                HNOW(HNODE(I),CORRNO,2) = 0
              END IF
            END IF
    4     CONTINUE
C
C     Calculate each processors contribution ;
C     Incorporate in this part a test input for the initial C lines.
C     This is normally set to zero;
C
          DO 7 PROC = 1, NPOINT * NPOINT
C
            DO 6 I = 1, NPOINT
C
              DO 5 K = 1, 2
                PN = G05CAF(PN)
                IF (PN .LT. 0.0) THEN
                  IF (CORRNO .EQ. 1) THEN
                    IF (K .EQ. 1) THEN
                      YNOW(NPOINT*NPOINT + I,CORRNO,K) = 1
                    ELSE
                      YNOW(NPOINT*NPOINT + I,CORRNO,K) = 0
                    END IF
                  END IF
                END IF
```

```
    5         CONTINUE
C
    6       CONTINUE
C
C     Use mapping to set node values for processor ;
C
            HIN = HNOW(MAP(PROC,1),CORRNO,1)
            XIN = XNOW(MAP(PROC,3),CORRNO,1)
            YIN = YNOW(MAP(PROC,5),CORRNO,1)
            HIN1 = HNOW(MAP(PROC,1),CORRNO,2)
            XIN1 = XNOW(MAP(PROC,3),CORRNO,2)
            YIN1 = YNOW(MAP(PROC,5),CORRNO,2)
C
C     Call processor CELL ;
C
            CALL CELL(HIN, HIN1, HOUT, HOUT1, XIN, XIN1, XOUT, XOUT1,
     &           YIN, YIN1, YOUT, YOUT1, PROC, CORRNO)
C
C     Set next node values from output of cell. (This leaves now node
C     values unchanged) ;
C
            HNEXT(MAP(PROC,2),CORRNO,1) = HOUT
            XNEXT(MAP(PROC,4),CORRNO,1) = XOUT
            YNEXT(MAP(PROC,6),CORRNO,1) = YOUT
            HNEXT(MAP(PROC,2),CORRNO,2) = HOUT1
            XNEXT(MAP(PROC,4),CORRNO,2) = XOUT1
            YNEXT(MAP(PROC,6),CORRNO,2) = YOUT1
    7     CONTINUE
C
C     Current round of node computations complete,
C
C     Call output averaging interface subroutine;
C
          CALL OUTCNT(CORRNO, TIME, YNUM)
C
C     Set next node values to now node values ;
C
          DO 8 I = 1, NPOINT * (NPOINT + 1)
            IVALH1 = HNEXT(I,CORRNO,1)
            HNOW(I,CORRNO,1) = IVALH1
            IVALY1 = YNEXT(I,CORRNO,1)
            YNOW(I,CORRNO,1) = IVALY1
            IVALH2 = HNEXT(I,CORRNO,2)
            HNOW(I,CORRNO,2) = IVALH2
            IVALY2 = YNEXT(I,CORRNO,2)
            YNOW(I,CORRNO,2) = IVALY2
    8     CONTINUE
C
          DO 9 I = 1, (NPOINT + 1) * (NPOINT + NPOINT - 1)
            IVALX1 = XNEXT(I,CORRNO,1)
            XNOW(I,CORRNO,1) = IVALX1
            IVALX2 = XNEXT(I,CORRNO,2)
            XNOW(I,CORRNO,2) = IVALX2
    9     CONTINUE
C
C     Continue with next real correlation (series of 4) ;
C
   10   CONTINUE
C
C     Call averaging and complex combination routine;
C
```

```
       CALL AVERAG(CLOCK, SCALE1)
C
C     Continue with next clock cycle;
C
C
C
C     Write out rescaled correlation output to file;
C
C
       DO 11 I = 1, NPOINT
         IF (CLOCK .GT. 0) THEN
           AC = ((DFLOAT((OUT(I,1,1) - OUT(I,1,2))))/CLOCK) * SCALE1
           BD = ((DFLOAT((OUT(I,2,1) - OUT(I,2,2))))/CLOCK) * SCALE1
           BC = ((DFLOAT((OUT(I,3,1) - OUT(I,3,2))))/CLOCK) * SCALE1
           AD = ((DFLOAT((OUT(I,4,1) - OUT(I,4,2))))/CLOCK) * SCALE1
         END IF
         IF ((CLOCK .EQ. 1) .AND. (TIME .GE. NPOINT)) WRITE (50,*) (AC
    &          - BD), (BC + AD)
         IF (CLOCK .EQ. 10) WRITE (51,*) (AC - BD), (BC + AD)
         IF (CLOCK .EQ. 100) WRITE (52,*) (AC - BD), (BC + AD)
         IF (CLOCK .EQ. 1000) WRITE (53,*) (AC - BD), (BC + AD)
         IF (CLOCK .EQ. 10000) WRITE (54,*) (AC - BD), (BC + AD)
C
C
  11   CONTINUE
C
  12 CONTINUE
C
C     Close all files -  set ICOM=0 ;
C
     ICOM = 0
     CALL READER(ICOM, XREAL, XIMAG, HREAL, HIMAG)
     STOP
     END
```

FORTRAN77 subroutine CELL.SUB

– inner product cell

```
C     ***********************************************
C     * Systolic stochastic Cell Subroutine ;       *
C     *                                             *
C     * - this subroutine provides a stochastic     *
C     *   set of operations on four inputs -        *
C     *   XIN,HIN,YIN and a memory block for each   *
C     *   processor, COUNT(proc) which is common    *
C     *   to the main routine and is used for the   *
C     *   summation together with the OR operation. *
C     * - The ,1 dimension is used for positive     *
C     *   values and the ,2 dimension is used for   *
C     *   negative values.                          *
C     * - The routine provides the following outputs.*
C     *   XOUT,HOUT and YOUT. (& the -ve dimension) *
C     * - XOUT=XIN                                  *
C     *   HOUT=HIN                                  *
C     *   YOUT=[YIN+(HIN.XIN)]                      *
C     * - Normalised ONE corresponds to a sequence  *
C     *   of all 1's.                               *
C     * - ZERO corresponds to all 0's.              *
C     * - Multiplication in stochastic systems as   *
C     *   defined on the above mapping may be done  *
C     *   via an AND operation. Full field (ie. + & *
C     *   -ve arithmetic) needs four AND gates.     *
C     * - Addition is accomplished by using an OR   *
C     *   operation on the input sequences.         *
C     * - The OR operation is validated for         *
C     *   coincident stochastic pulses by providing *
C     *   a pseudo carry. This is a counter which   *
C     *   increments for each coincident event      *
C     *   and decrements for each non OR or AND     *
C     *   event, at the same time inserting an      *
C     *   output pulse for every available slot     *
C     *   until the counter is zero. This corresponds*
C     *   to compensation for coincident pulses.    *
C     * - The COMMON block is divided by 4 for the  *
C     *   four correlation operations and by 2 for  *
C     *   the positive & negative side.             *
C     *                                             *
C     * Author: Richard Leaver                      *
C     * Created:14th November 1986                  *
C     * Update : 2nd February 1987                  *
C     * Frozen : 5th May 1988                       *
C     ***********************************************
      SUBROUTINE CELL(HIN, HIN1, HOUT, HOUT1, XIN, XIN1, XOUT, XOUT1,
     &           YIN, YIN1, YOUT, YOUT1, PROC, CORRNO)
C
C     Define standard COMMON block;
C
      COMMON COUNT, OUT, HNOW, HNEXT, YNOW, YNEXT, XNOW, XNEXT, NPOINT
C
C     Define Integer input and output variables;
C
      INTEGER HIN, HOUT, XIN, XOUT, YIN, YOUT, RSULT, PROC, CORRNO
      INTEGER HIN1, HOUT1, XIN1, XOUT1, YIN1, YOUT1, RSULT1
C
C     Define arrays;
C
      INTEGER COUNT(10000,4,2)
      INTEGER OUT(96,4,2)
      INTEGER HNOW(10100,4,2), HNEXT(10100,4,2)
      INTEGER YNOW(10100,4,2), YNEXT(10100,4,2)
```

```
      INTEGER XNOW(20099,4,2), XNEXT(20099,4,2)
C
C     Perform operations;
C     Pass out the input values of HIN and XIN to HOUT and XOUT.
C
      HOUT = HIN
      XOUT = XIN
      HOUT1 = HIN1
      XOUT1 = XIN1
C
C     This part performs the multiply/accumulate operation.
C     First the multiply - using an AND operation.
C     This needs four operations for full arithmetic fields
C     NB: This only works for the mapping zero=0,normalised one=1 ;
C
      IF (((HIN .EQ. 1) .AND. (XIN .EQ. 1)) .OR. ((HIN1 .EQ. 1) .AND. (
     &       XIN1 .EQ. 1))) THEN
        RSULT = 1
      ELSE
        RSULT = 0
      END IF
      IF (((HIN .EQ. 1) .AND. (XIN1 .EQ. 1)) .OR. ((HIN1 .EQ. 1) .AND. (
     &       XIN .EQ. 1))) THEN
        RSULT1 = 1
      ELSE
        RSULT1 = 0
      END IF
C
C     Apply a slight improvement to the hardware. Test to see if when
C     the product of x*h is one polarity and the Y input diagonal is
C     the opposite, can we cancel them out with the resultant saving
C     in counter capacity?;
      IF ((YIN .EQ. 1) .AND. (RSULT1 .EQ. 1)) THEN
        YIN = 0
        RSULT1 = 0
      END IF
      IF ((YIN1 .EQ. 1) .AND. (RSULT .EQ. 1)) THEN
        YIN1 = 0
        RSULT = 0
      END IF
C
C     Now the summation :add the result to YIN and pass to YOUT;
C     First the positive side ;
C
      IF ((YIN .EQ. 1) .OR. (RSULT .EQ. 1)) THEN
        YOUT = 1
      ELSE
        YOUT = 0
      END IF
      IF ((YIN .EQ. 1) .AND. (RSULT .EQ. 1)) THEN
        COUNT(PROC,CORRNO,1) = COUNT(PROC,CORRNO,1) + 1
      END IF
      IF ((YIN .EQ. 0) .AND. (RSULT .EQ. 0) .AND. (COUNT(PROC,CORRNO,1)
     &       .GT. 0)) THEN
        COUNT(PROC,CORRNO,1) = COUNT(PROC,CORRNO,1) - 1
        YOUT = 1
      END IF
C
C     Now the negative side ;
C
```

```
      IF ((YIN1 .EQ. 1) .OR. (RSULT1 .EQ. 1)) THEN
        YOUT1 = 1
      ELSE
        YOUT1 = 0
      END IF
      IF ((YIN1 .EQ. 1) .AND. (RSULT1 .EQ. 1)) THEN
        COUNT(PROC,CORRNO,2) = COUNT(PROC,CORRNO,2) + 1
      END IF
      IF ((YIN1 .EQ. 0) .AND. (RSULT1 .EQ. 0) .AND. (COUNT(PROC,CORRNO,
     &       2) .GT. 0)) THEN
        COUNT(PROC,CORRNO,2) = COUNT(PROC,CORRNO,2) - 1
        YOUT1 = 1
      END IF
C
C     Option: write out the counter state;
C
C     IF (CORRNO .EQ. 1) THEN
C       WRITE (20,*) COUNT(PROC,CORRNO,1)
C       WRITE (21,*) COUNT(PROC,CORRNO,2)
C     END IF
      RETURN
      END
```

INPUT

– Input data file

```
6 POINTS
6
.6234904,-.781831,
-.9009672,-.433887,
-.2225188,-.9749284,
.6234863,.7818343,
-.9009703,.4338807,
-.2225256,.9749269,
```

REFER

– Reference data file

```
6 Points
6,
1.,0.,
1.,0.,
1.,0.,
0.,0.,
0.,0.,
0.,0.,
```

FORTRAN77 subroutine INIT.SUB

– Initialisation

```
C     ***********************************************
C     * Stochastic PN Generator Initialisation      *
C     * Routine ;                                   *
C     *                                             *
C     * - this subroutine initialises the NAG       *
C     *   library G05CAF pn generator by calling    *
C     *   the NAG routine G05CBF.                   *
C     * - This must be performed at the start of    *
C     *   any program using the pn generator.       *
C     *   Use of G05CBF (rather than G05CCF)        *
C     *   results in repeatable sequences.          *
C     * - the subroutine then initialises all the   *
C     *   various nodes and counters to zero;       *
C     *                                             *
C     * Author: Richard Leaver                      *
C     * Created:14th November 1986                  *
C     * Update : 5th January  1987                  *
C     * Frozen : 8th January  1987                  *
C     ***********************************************
      SUBROUTINE INIT
C
C     Define standard COMMON block;
C
      COMMON COUNT, OUT, HNOW, HNEXT, YNOW, YNEXT, XNOW, XNEXT, NPOINT
C
C     Define INTERF named COMMON block;
C
      COMMON /INTERF/ SHIFT, ISLOT, ISELEC
C
C     Define arrays;
C
      INTEGER COUNT(10000,4,2)
      INTEGER OUT(96,4,2)
      INTEGER HNOW(10100,4,2), HNEXT(10100,4,2)
      INTEGER YNOW(10100,4,2), YNEXT(10100,4,2)
      INTEGER XNOW(20099,4,2), XNEXT(20099,4,2)
      INTEGER SHIFT(10000,6,4,2)
C
C     Call NAG repeatable initialisation routine ;
C     Use INTEGER seed set to 1;
C
      CALL G05CBF(1)
C
C     Initialise output node counters ;
C
      DO 3 I = 1, NPOINT
C
        DO 2 J = 1, 4
C
          DO 1 K = 1, 2
            OUT(I,J,K) = 0
    1     CONTINUE
C
    2   CONTINUE
C
    3 CONTINUE
C
C     Initialise all nodes to value 0 ;
C
      DO 6 I = 1, NPOINT * (NPOINT + 1)
C
        DO 5 J = 1, 4
C
```

```
            DO 4 K = 1, 2
              HNOW(I,J,K) = 0
              HNEXT(I,J,K) = 0
              YNOW(I,J,K) = 0
              YNEXT(I,J,K) = 0
    4       CONTINUE
C
    5     CONTINUE
C
    6   CONTINUE
C
        DO 9 I = 1, (NPOINT + 1) * (NPOINT + NPOINT - 1)
C
          DO 8 J = 1, 4
C
            DO 7 K = 1, 2
              XNOW(I,J,K) = 0
              XNEXT(I,J,K) = 0
    7       CONTINUE
C
    8     CONTINUE
C
    9   CONTINUE
C
C       Initialise the counters to 0 ;
C
        DO 12 I = 1, NPOINT * NPOINT
C
          DO 11 J = 1, 4
C
            DO 10 K = 1, 2
              COUNT(I,J,K) = 0
   10       CONTINUE
C
   11     CONTINUE
C
   12   CONTINUE
C
        RETURN
        END
```

FORTRAN77 subroutine RAND.SUB

– Random numbers

```
C     ************************************************
C     * Random Number Generator Subroutine ;         *
C     *                                              *
C     * - routine  uses the NAG routine G05CAF       *
C     *   which generates random numbers.            *
C     * - The calling program must have called       *
C     *   the NAG initialisation  routine G05CCF.    *
C     * - routine  to be used for stochastic         *
C     *   systolic processing.                       *
C     *                                              *
C     * - NAG routine requires Double Precision      *
C     *   variables - must be used otherwise         *
C     *   errors will occur.                         *
C     *                                              *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created:14th November 1986                   *
C     * Update :14th November 1986                   *
C     * Frozen :14th November 1986                   *
C     ************************************************
      SUBROUTINE RAND(X)
C
C     Define standard COMMON block;
C
      COMMON COUNT, OUT, HNOW, HNEXT, YNOW, YNEXT, XNOW, XNEXT, NPOINT
C
C     Define arrays;
C
      INTEGER COUNT(10000,4,2)
      INTEGER OUT(96,4,2)
      INTEGER HNOW(10100,4,2), HNEXT(10100,4,2)
      INTEGER YNOW(10100,4,2), YNEXT(10100,4,2)
      INTEGER XNOW(20099,4,2), XNEXT(20099,4,2)
C
C     Define Double Precision variables ;
C
      DOUBLE PRECISION X, G05CAF
C
C     Call NAG routine ;
C
      X = G05CAF(X)
      RETURN
      END
```

FORTRAN77 subroutine READER.SUB

– Input/Output functions

```
C     ***********************************************
C     * Reading Subroutine - Systolic Array ;       *
C     *                                             *
C     * - This subroutine is used to OPEN and CLOSE *
C     *   files and to read into the arrays;        *
C     * - If ICOM=1 then OPEN, else CLOSE           *
C     *                                             *
C     * Author: Richard Leaver                      *
C     * Created:14th November 1986                  *
C     * Update : 8th January  1987                  *
C     * Frozen : 8th January  1987                  *
C     ***********************************************
      SUBROUTINE READER(ICOM, XREAL, XIMAG, HREAL, HIMAG)
C
C     Define standard COMMON block;
C
      COMMON COUNT, OUT, HNOW, HNEXT, YNOW, YNEXT, XNOW, XNEXT, NPOINT
C
C     Define arrays;
C
      INTEGER COUNT(10000,4,2)
      INTEGER OUT(96,4,2)
      INTEGER HNOW(10100,4,2), HNEXT(10100,4,2)
      INTEGER YNOW(10100,4,2), YNEXT(10100,4,2)
      INTEGER XNOW(20099,4,2), XNEXT(20099,4,2)
C
C     Define Input arrays as DOUBLE PRECISION;
C
      DOUBLE PRECISION XREAL(0:95), XIMAG(0:95), HREAL(0:95), HIMAG(0:95)
C
C     Define character space for the title;
C
      CHARACTER*30 TITLE, TITLE1
C
C     If ICOM=1 then OPEN files else CLOSE them;
C
      IF (ICOM .EQ. 1) THEN
C
C     Open datafiles & files ;
C
       OPEN (1,FILE='INPUT')
       OPEN (2,FILE='REFER')
C
C     3 used for the second x input data (non stationary inputs) ;
C
       OPEN (3,FILE='SECOND')
C
C      4 used for the final output file - only useful for stationary
C      input;
C
       OPEN (4,FILE='RESULT')
C
C
C
       OPEN (11,FILE='A1')
       OPEN (12,FILE='A2')
       OPEN (13,FILE='A3')
       OPEN (14,FILE='A4')
       OPEN (15,FILE='A5')
       OPEN (16,FILE='A6')
C
C     Read in Title and No of Points for input sequence;
C
```

```
        READ (1,2) TITLE
        READ (1,*) NPOINT
C
C     Read in Title and No of Points for reference sequence ;
C     TITLE1 and MPOINT are dummy variables - assume NPOINT=MPOINT
C     for Cyclic correlation processing ;
C
        READ (2,2) TITLE1
        READ (2,*) MPOINT
C
C     Label Output File ;
C
        WRITE (4,2) TITLE
        WRITE (4,*) NPOINT
C
C     Read in values;
C
        DO 1 I = 0, (NPOINT - 1)
          READ (1,*) XREAL(I), XIMAG(I)
          READ (2,*) HREAL(I), HIMAG(I)
    1   CONTINUE
C
      ELSE
        CLOSE (1)
        CLOSE (2)
        CLOSE (3)
        CLOSE (11)
        CLOSE (12)
        CLOSE (13)
        CLOSE (14)
        CLOSE (15)
        CLOSE (16)
      END IF
      RETURN
    2 FORMAT (A)
      END
```

FORTRAN77 subroutine MAPPER.SUB

– arranges cell interconnections within array

```
C     ************************************************
C     * Mapping Program    - Systolic Array ;        *
C     *                                              *
C     * - this subroutine provides the mapping       *
C     *   for the interconnection of systolic cells  *
C     *   in an N x N configuration representing     *
C     *   an N point correlation.                    *
C     * - The program takes the number of points as  *
C     *   an input and outputs the node mapping in   *
C     *   the MAP (Processor number,node type)       *
C     *   array plus three arrays HNODE,XNODE,YNODE  *
C     *   which contain the values of the array I/O  *
C     *   nodes.                                     *
C     *   These three arrays are accompanied by      *
C     *   the variables HNUM,XNUM and YNUM which     *
C     *   contain the end values for reading the     *
C     *   three arrays.                              *
C     * - This subroutine was specially adjusted to  *
C     * - be used by the ARRAY1.FTN complex          *
C     *   correlation program which requires a lot   *
C     *   more memory than the real positive         *
C     *   correlation performed in ARRAY.FTN         *
C     * - maximum number of points is 6.             *
C     *   This is because of memory limitations.     *
C     * - This subroutine is exactly the same as     *
C     *   MAPPER.SUB.1 except that it uses the       *
C     *   COMMON block for NPOINT.                   *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created:14th November 1986                   *
C     * Update :14th November 1986                   *
C     * Frozen :14th November 1986                   *
C     ************************************************
C
      SUBROUTINE MAP1(HNUM, XNUM, YNUM, MAP, HNODE, XNODE, YNODE)
C
C
C      Define arrays to hold the intermediate result mappings;
C      HARRAY(N,N+1),YARRAY(N+1,N),XARRAY(N+1,N+N-1)
C      together with MAP(N*N,6)
C      All arrays are defined in terms of H inputs H(0) to H(N-1);
C      This is via the row select of the array (N,      ;
C       As in the thesis, H(0) is at the bottom left corner of the array
C       with H(N-1) at the top left. Consequently mapping values are presented
C       with the cell numbering commencing top left to top right etc.
C       XARRAY is special in that since the array is a parallelogram,
C       we have to be careful. I define array value 0 as a filler for
C       cases where no connection is made in the matrix.
C       X0 is the bottom left corner. The Y array inputs to the cells
C       H values will have input and output. The Y array inputs to the
C       cells are defined as zero. The Y array extends to -1 to include these
C       YIN's ;
C
C
C      Define standard COMMON block;
C
      COMMON COUNT, OUT, HNOW, HNEXT, YNOW, YNEXT, XNOW, XNEXT, NPOINT
C
C      Define arrays;
C
      INTEGER COUNT(10000,4,2)
      INTEGER OUT(96,4,2)
```

```
      INTEGER HNOW(10100,4,2), HNEXT(10100,4,2)
      INTEGER YNOW(10100,4,2), YNEXT(10100,4,2)
      INTEGER XNOW(20099,4,2), XNEXT(20099,4,2)
      INTEGER HARRAY(0:95,97), YARRAY(-1:95,97)
      INTEGER XARRAY(0:96,0:190)
      INTEGER MAP(10000,6)
C
C     Define arrays to contain the I/O nodes ;
C
      INTEGER HNODE(96), XNODE(191), YNODE(96)
C
C     Define integer variables for processing ;
C
      INTEGER HNUM, XNUM, YNUM
C
C     Define HNUM,XNUM and YNUM with end point values ;
C
      HNUM = NPOINT
      XNUM = NPOINT + NPOINT - 1
      YNUM = NPOINT
C
C     Calculate H mapping;
C
      ICOUNT = 0
C
      DO 2 I = NPOINT - 1, 0, -1
C
        DO 1 J = 1, NPOINT + 1
          HARRAY(I,J) = ICOUNT + J
    1   CONTINUE
C
        ICOUNT = ICOUNT + NPOINT + 1
    2 CONTINUE
C
C     Fill the mapping array with HIN and HOUT values from the above
C     process. 1=HIN, 2=HOUT ;
C
      LEVEL = 0
C
      DO 4 ICOUNT = NPOINT - 1, 0, -1
C
        DO 3 JCOUNT = 1, NPOINT
C
C     Allow for increment in Processor number as we descend levels ;
C
          IPROC = JCOUNT + LEVEL
          MAP(IPROC,1) = HARRAY(ICOUNT,JCOUNT)
          MAP(IPROC,2) = HARRAY(ICOUNT,JCOUNT + 1)
    3   CONTINUE
C
        LEVEL = LEVEL + NPOINT
    4 CONTINUE
C
C     Write out H Input nodes into HNODE array ;
C
      DO 5 I = 0, NPOINT - 1
        J = I + 1
        HNODE(J) = HARRAY(I,1)
    5 CONTINUE
C
C     Calculate Y mapping;
C
```

```
      ICOUNT = 0
C
      DO 7 I = NPOINT - 1, -1, -1
C
        DO 6 J = 1, NPOINT
          YARRAY(I,J) = ICOUNT + J
    6   CONTINUE
C
        ICOUNT = ICOUNT + NPOINT
    7 CONTINUE
C
C     Fill the mapping array with YIN and YOUT values from the above
C     process. 5=YIN, 6=YOUT ;
C
      LEVEL = 0
C
      DO 9 ICOUNT = NPOINT - 1, 0, -1
C
        DO 8 JCOUNT = 1, NPOINT
C
C     Allow for increment in Processor number as we descend levels ;
C
          IPROC = JCOUNT + LEVEL
          MAP(IPROC,5) = YARRAY(ICOUNT - 1,JCOUNT)
          MAP(IPROC,6) = YARRAY(ICOUNT,JCOUNT)
    8   CONTINUE
C
        LEVEL = LEVEL + NPOINT
    9 CONTINUE
C
C     Write out Y Input nodes into YNODE array ;
C
      DO 10 I = 1, NPOINT
        YNODE(I) = YARRAY(NPOINT - 1,I)
   10 CONTINUE
C
C     Initialise XARRAY;
C
      DO 12 I = 0, NPOINT
C
        DO 11 J = 0, NPOINT + NPOINT - 2
          XARRAY(I,J) = 0
   11   CONTINUE
C
   12 CONTINUE
C
C     Calculate XARRAY values ;
C
      IFLAG = 0
      ICOUNT = 1
      JCOUNT = 1
C
      DO 14 J = 0, NPOINT + NPOINT - 2
C
        DO 13 I = 0, NPOINT
          IF (IFLAG .EQ. 0) THEN
            IF (I .LE. ICOUNT) THEN
              XARRAY(I,J) = JCOUNT
              JCOUNT = JCOUNT + 1
            END IF
```

```
          ELSE
            IF (I .GT. (NPOINT + 1 - ICOUNT)) THEN
              XARRAY(I,J) = JCOUNT
              JCOUNT = JCOUNT + 1
            END IF
          END IF
   13   CONTINUE
C
        IF (IFLAG .EQ. 0) ICOUNT = ICOUNT + 1
        IF (IFLAG .EQ. 1) ICOUNT = ICOUNT - 1
        IF (ICOUNT .EQ. NPOINT + 1) IFLAG = 1
   14 CONTINUE
C
C     Fill the mapping array with XIN and XOUT values from the above
C     process. 3=XIN, 4=XOUT ;
C
C
C     This case is the most complicated since X is a parallelogram ;
C
      LEVEL = NPOINT * NPOINT
C
      DO 16 ICOUNT = 0, NPOINT - 1
        LEVEL = LEVEL - NPOINT
C
        DO 15 JCOUNT = 1, NPOINT
          IPROC = LEVEL + JCOUNT
          KCOUNT = JCOUNT + ICOUNT
          MAP(IPROC,3) = XARRAY(ICOUNT,KCOUNT - 1)
          MAP(IPROC,4) = XARRAY(ICOUNT + 1,KCOUNT - 1)
   15   CONTINUE
C
   16 CONTINUE
C
C     Write out X Input nodes into XNODE array ;
C
      ICOUNT = 0
      KCOUNT = 1
C
      DO 17 JCOUNT = 0, NPOINT + NPOINT - 2
        IF (XARRAY(ICOUNT,JCOUNT) .NE. 0) THEN
          XNODE(KCOUNT) = XARRAY(ICOUNT,JCOUNT)
        ELSE
          ICOUNT = ICOUNT + 1
          XNODE(KCOUNT) = XARRAY(ICOUNT,JCOUNT)
        END IF
        KCOUNT = KCOUNT + 1
   17 CONTINUE
C
      RETURN
      END
```

FORTRAN77 subroutine OUTCNT.SUB

– output averaging

```
C     ***********************************************
C     * Stochastic Output Interface Subroutine ;     *
C     *                                              *
C     * - This routine performs the selectable       *
C     *   output counting.                           *
C     * - ISELEC=0       - Standard averaging        *
C     * - ISELEC=1       - Moving Average            *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created:13th November 1986                   *
C     * Update : 5th January  1987                   *
C     * Frozen : 8th January  1987                   *
C     ***********************************************
      SUBROUTINE OUTCNT(CORRNO, TIME, YNUM)
C
C     Define standard COMMON block;
C
      COMMON COUNT, OUT, HNOW, HNEXT, YNOW, YNEXT, XNOW, XNEXT, NPOINT
C
C     Define standard COMMON block;
C
      COMMON /INTERF/ SHIFT, ISLOT, ISELEC
C
C     Define integer array for the output averaging of the four
C     correlations;
C
C     Define arrays;
C
      INTEGER COUNT(10000,4,2)
      INTEGER OUT(96,4,2)
      INTEGER HNOW(10100,4,2), HNEXT(10100,4,2)
      INTEGER YNOW(10100,4,2), YNEXT(10100,4,2)
      INTEGER XNOW(20099,4,2), XNEXT(20099,4,2)
      INTEGER SHIFT(10000,6,4,2)
C
C     Define Integer variables;
C
      INTEGER CORRNO, TIME, YNUM
C
C     If ISELEC=0 then perform normal averaging. Do not reset the
C     OUT counter
C
C     Begin ISELEC IF statement
C
      IF (ISELEC .EQ. 0) THEN
C
        DO 2 I = 1, YNUM
C
          DO 1 K = 1, 2
            IF ((YNOW(I,CORRNO,K) .EQ. 1) .AND. (TIME .GT. NPOINT))
     &           THEN
              OUT(I,CORRNO,K) = OUT(I,CORRNO,K) + 1
            END IF
    1     CONTINUE
C
    2   CONTINUE
C
      ELSE
C
C     Initialise OUT array ready for counting ON states;
C
        DO 4 K = 1, YNUM
C
```

```
          DO 3 L = 1, 2
            OUT(K,CORRNO,L) = 0
    3     CONTINUE
C
    4   CONTINUE
C
C     Calculate no of ON states by examining 1 to ISLOT
C     where ISLOT is the window length;
C
       DO 7 I = 1, ISLOT
C
          DO 6 K = 1, YNUM
C
            DO 5 L = 1, 2
              IF ((SHIFT(I,K,CORRNO,L) .EQ. 1) .AND. (TIME .GT. NPOINT))
     &                THEN
                OUT(K,CORRNO,L) = OUT(K,CORRNO,L) + 1
              END IF
    5       CONTINUE
C
    6     CONTINUE
C
    7   CONTINUE
C
C     End ISELEC IF statement
C
      END IF
      RETURN
      END
```

FORTRAN77 subroutine AVERAG.SUB

– final processing

```
C     ***********************************************
C     * Stochastic Output Interface Subroutine ;     *
C     *                                              *
C     * - This routine performs selectable           *
C     *   output averaging.                          *
C     * - The mode of averaging is selected by       *
C     *   ISELEC                                     *
C     * - ISELEC=0        - Standard Averaging       *
C     * - ISELEC=1        - Moving Average           *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created:13th November 1986                   *
C     * Update : 5th January  1987                   *
C     * Frozen : 8th January  1987                   *
C     ***********************************************
      SUBROUTINE AVERAG(CLOCK, SCALE1)
C
C     Define standard COMMON block;
C
      COMMON COUNT, OUT, HNOW, HNEXT, YNOW, YNEXT, XNOW, XNEXT, NPOINT
      COMMON /INTERF/ SHIFT, ISLOT, ISELEC
C
C     Define INTERF named COMMON block;
C
C
C     Define arrays;
C
      INTEGER COUNT(10000,4,2)
      INTEGER OUT(96,4,2)
      INTEGER HNOW(10100,4,2), HNEXT(10100,4,2)
      INTEGER YNOW(10100,4,2), YNEXT(10100,4,2)
      INTEGER XNOW(20099,4,2), XNEXT(20099,4,2)
      INTEGER SHIFT(10000,6,4,2)
C
C     Define Double Precision Variables;
C
      DOUBLE PRECISION CLOCK, SCALE1, DIVIS
C
C     Calculate Real and Imaginary current values of the
C     four correlations ;
C     Convert back to deterministic representation including scaling ;
C
C     Look at all 6 outputs in this test case;
C
C     If ISELEC=0 then perform SUCCESSIVE averaging;
C
C     Define Divisor
C
      IF (ISELEC .EQ. 0) THEN
        DIVIS = CLOCK
      ELSE
        DIVIS = DFLOAT(ISLOT)
      END IF
C
C     Processor 1;
C
      IF (NPOINT .GE. 1) THEN
        I = 1
        AC = ((DFLOAT((OUT(I,1,1) - OUT(I,1,2))))/DIVIS) * SCALE1
        BD = ((DFLOAT((OUT(I,2,1) - OUT(I,2,2))))/DIVIS) * SCALE1
        BC = ((DFLOAT((OUT(I,3,1) - OUT(I,3,2))))/DIVIS) * SCALE1
        AD = ((DFLOAT((OUT(I,4,1) - OUT(I,4,2))))/DIVIS) * SCALE1
C
C     Write out  rescaled  correlation, output to file every
C     cycle;
```

```
C
        IF (MOD(INT(CLOCK),1) .EQ. 0) THEN
          WRITE (11,*) CLOCK, (AC - BD), (BC + AD)
        END IF
      END IF
      IF (NPOINT .GE. 2) THEN
C
C      Processor 2;
C
        I = 2
        AC = ((DFLOAT((OUT(I,1,1) - OUT(I,1,2))))/DIVIS) * SCALE1
        BD = ((DFLOAT((OUT(I,2,1) - OUT(I,2,2))))/DIVIS) * SCALE1
        BC = ((DFLOAT((OUT(I,3,1) - OUT(I,3,2))))/DIVIS) * SCALE1
        AD = ((DFLOAT((OUT(I,4,1) - OUT(I,4,2))))/DIVIS) * SCALE1
C
C      Write out  rescaled  correlation, output to file every
C      cycle;
C
        IF (MOD(INT(CLOCK),1) .EQ. 0) THEN
          WRITE (12,*) CLOCK, (AC - BD), (BC + AD)
        END IF
      END IF
      IF (NPOINT .GE. 3) THEN
C
C      Processor 3;
C
        I = 3
        AC = ((DFLOAT((OUT(I,1,1) - OUT(I,1,2))))/DIVIS) * SCALE1
        BD = ((DFLOAT((OUT(I,2,1) - OUT(I,2,2))))/DIVIS) * SCALE1
        BC = ((DFLOAT((OUT(I,3,1) - OUT(I,3,2))))/DIVIS) * SCALE1
        AD = ((DFLOAT((OUT(I,4,1) - OUT(I,4,2))))/DIVIS) * SCALE1
C
C      Write out  rescaled  correlation, output to file every
C      cycle;
C
        IF (MOD(INT(CLOCK),1) .EQ. 0) THEN
          WRITE (13,*) CLOCK, (AC - BD), (BC + AD)
        END IF
      END IF
C
      IF (NPOINT .GE. 4) THEN
C      Processor 4;
C
        I = 4
        AC = ((DFLOAT((OUT(I,1,1) - OUT(I,1,2))))/DIVIS) * SCALE1
        BD = ((DFLOAT((OUT(I,2,1) - OUT(I,2,2))))/DIVIS) * SCALE1
        BC = ((DFLOAT((OUT(I,3,1) - OUT(I,3,2))))/DIVIS) * SCALE1
        AD = ((DFLOAT((OUT(I,4,1) - OUT(I,4,2))))/DIVIS) * SCALE1
C
C      Write out  rescaled  correlation, output to file every
C      cycle;
C
        IF (MOD(INT(CLOCK),1) .EQ. 0) THEN
          WRITE (14,*) CLOCK, (AC - BD), (BC + AD)
        END IF
      END IF
      IF (NPOINT .GE. 5) THEN
C
C      Processor 5;
C
        I = 5
        AC = ((DFLOAT((OUT(I,1,1) - OUT(I,1,2))))/DIVIS) * SCALE1
        BD = ((DFLOAT((OUT(I,2,1) - OUT(I,2,2))))/DIVIS) * SCALE1
```

```
        BC = ((DFLOAT((OUT(I,3,1) - OUT(I,3,2))))/DIVIS) * SCALE1
        AD = ((DFLOAT((OUT(I,4,1) - OUT(I,4,2))))/DIVIS) * SCALE1
C
C     Write out  rescaled  correlation, output to file every
C     cycle;
C
        IF (MOD(INT(CLOCK),1) .EQ. 0) THEN
          WRITE (15,*) CLOCK, (AC - BD), (BC + AD)
        END IF
      END IF
      IF (NPOINT .GE. 6) THEN
C
C     Processor 6;
C
        I = 6
        AC = ((DFLOAT((OUT(I,1,1) - OUT(I,1,2))))/DIVIS) * SCALE1
        BD = ((DFLOAT((OUT(I,2,1) - OUT(I,2,2))))/DIVIS) * SCALE1
        BC = ((DFLOAT((OUT(I,3,1) - OUT(I,3,2))))/DIVIS) * SCALE1
        AD = ((DFLOAT((OUT(I,4,1) - OUT(I,4,2))))/DIVIS) * SCALE1
C
C     Write out  rescaled  correlation, output to file every
C     cycle;
C
        IF (MOD(INT(CLOCK),1) .EQ. 0) THEN
          WRITE (16,*) CLOCK, (AC - BD), (BC + AD)
        END IF
      END IF
      RETURN
      END
```

FORTRAN77 program INPUT.GEN

– generates reordered 'twiddle' coefficients for DFT via correlation

```
C     ************************************************
C     * - Discrete Fourier Transform - using         *
C     *   Correlation.                               *
C     *                                              *
C     * - This routine generates reordered sine      *
C     *   and cosine 'twiddle' terms using the       *
C     *   primitive root method for a 46 point       *
C     *   sequence (used to generate the 47 point    *
C     *   DFT).                                      *
C     *                                              *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created: 17th January 1987                   *
C     * Update : 19th March 1987                     *
C     * Frozen : 5th May 1988                        *
C     ************************************************
      INTEGER A(0:45)
      DATA A /46*1.0/
C
      DO 2 I = 0, 45
C
        DO 1 K = 1, I
          A(I) = MOD(5*A(I),47)
    1   CONTINUE
C     WRITE(6,*)I,A(I)
    2 CONTINUE
C
      PI = 3.14154
      WRITE (1,*)' TITLE'
      WRITE (1,*)' 46'
C
      DO 3 I = 0, 45
        X = COS(2.0*PI*FLOAT(A(I))/47.0)
        Y = -1.0 * SIN(2.0*PI*FLOAT(A(I))/47.0)
        WRITE (1,*) X, Y
    3 CONTINUE
C
      STOP
      END
```

FORTRAN77 program REFER.GEN

– reorders input data for DFT via correlation

```
C     ************************************************
C     * - Discrete Fourier Transform - using         *
C     *   Correlation.                               *
C     *                                              *
C     * - This routine generates reordered input     *
C     *   terms using the primitive root method      *
C     *   for a 46 point sequence (used to generate  *
C     *   the 47 point DFT).                         *
C     *                                              *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created: 17th January 1987                   *
C     * Update : 19th March 1987                     *
C     * Frozen : 5th May 1988                        *
C     ************************************************
      INTEGER A(0:45)
      INTEGER B(46)
      DATA A /46*1.0/
      DATA B /46*0.0/
      B(1) = 1
      B(2) = 1
      B(3) = 1
C
      DO 2 I = 0, 45
C
        DO 1 K = 1, I
          A(I) = MOD(5*A(I),47)
    1   CONTINUE
C      WRITE(6,*)I,A(I)
    2 CONTINUE
C
      PI = 3.14154
      WRITE (1,*)' TITLE'
      WRITE (1,*)' 46'
C
      DO 3 I = 0, 45
        X = B(A(I))
        Y = 0.0
        WRITE (1,*) X, Y
    3 CONTINUE
C
      STOP
      END
```

FORTRAN77 subroutine DFT.REARRANG

– reorders result for DFT via correlation

```
C     ************************************************
C     * - Discrete Fourier Transform - using         *
C     *   Correlation.                               *
C     *                                              *
C     * - This routine inputs a 46 point resultant   *
C     *   correlation and reorders it using the      *
C     *   primitive root method. It then inserts     *
C     *   the appropriate DC value as required for   *
C     *   this particular input sequence.            *
C     *                                              *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created: 17th January 1987                   *
C     * Update : 19th March 1987                     *
C     * Frozen : 5th May 1988                        *
C     ************************************************
      CHARACTER*70 TITLE
      REAL*4 R1(0:45), I1(0:45)
      INTEGER A(0:45)
      INTEGER B(46)
      DATA A /46*1.0/
      DATA B /46*1.0/
      READ (1,5) TITLE
      READ (1,*) NPOINT
C
      DO 1 I = 0, NPOINT - 1
        READ (1,*) R1(I), I1(I)
    1 CONTINUE
C
      DO 3 I = 0, NPOINT - 1
C
        DO 2 K = 1, I
          A(I) = MOD(5*A(I),47)
    2   CONTINUE
C
        B(A(I)) = I
    3 CONTINUE
C
      WRITE (3,5) TITLE
      WRITE (3,*) NPOINT + 1
C
      WRITE (3,*) 4., 0.0
C
      DO 4 I = 1, NPOINT
        WRITE (3,*) 1. + R1(B(I)), I1(B(I))
C        WRITE (6,*) I, B(I), I1(B(I))
    4 CONTINUE
C
      STOP
    5 FORMAT (A)
      END
```

Simulation programs for Appendix 4 – Neural Networks

FORTRAN77 program ARP.PREP

– prepares Input–Output vector files

```
C     ************************************************
C     * Associative Reward-Punish Element (ARP)      *
C     * Data Preparation Aid Program;                *
C     *                                              *
C     * - This program prepares the INPUT datafile   *
C     *   for the ARP.MAIN program in the correct    *
C     *   format;                                    *
C     * - Multiple images are available;             *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created: 1st June 1987                       *
C     * Update :23rd July 1987                       *
C     * Frozen : 5th May  1988                       *
C     ************************************************
      PROGRAM PREP
C
C     Define Implicit variables;
C
      IMPLICIT REAL*8 (A - H,Q - Z)
C
C     Real*8 variables;
C
      REAL*8 LAMBDA
      REAL*8 REW1, PUN1
C
C     Integers;
C
      INTEGER MAXCNT, NUMIM
      INTEGER LENGTH, WIDTH, DEPTH
C
C     Define Images;
C
      REAL*8 IMGE0(0:4,0:4)
      REAL*8 IMGE1(0:4,0:4)
      REAL*8 IMGE2(0:4,0:4)
      REAL*8 IMGE3(0:4,0:4)
      REAL*8 IMGE4(0:4,0:4)
      REAL*8 IMGE5(0:4,0:4)
      REAL*8 IMGE6(0:4,0:4)
      REAL*8 IMGE7(0:4,0:4)
      REAL*8 IMGE8(0:4,0:4)
      REAL*8 IMGE9(0:4,0:4)
C
C     Define Answers;
C
      REAL*8 ANS0(0:4,0:4)
      REAL*8 ANS1(0:4,0:4)
      REAL*8 ANS2(0:4,0:4)
      REAL*8 ANS3(0:4,0:4)
      REAL*8 ANS4(0:4,0:4)
      REAL*8 ANS5(0:4,0:4)
      REAL*8 ANS6(0:4,0:4)
      REAL*8 ANS7(0:4,0:4)
      REAL*8 ANS8(0:4,0:4)
      REAL*8 ANS9(0:4,0:4)
C
C     Define Corrupted image;
C
      REAL*8 BAD0(0:4,0:4)
      REAL*8 BAD1(0:4,0:4)
      REAL*8 BAD2(0:4,0:4)
      REAL*8 BAD3(0:4,0:4)
```

```
      REAL*8 BAD4(0:4,0:4)
      REAL*8 BAD5(0:4,0:4)
      REAL*8 BAD6(0:4,0:4)
      REAL*8 BAD7(0:4,0:4)
      REAL*8 BAD8(0:4,0:4)
      REAL*8 BAD9(0:4,0:4)
C
C     Characters;
C
      CHARACTER*70 TITLE
      CHARACTER*10 NAME
C
C     Assign image data;
C
      DATA IMGE0/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA IMGE1/
     &0.0D0,0.0D0,1.0D0,0.0D0,0.0D0,
     &0.0D0,0.0D0,1.0D0,0.0D0,0.0D0,
     &0.0D0,0.0D0,1.0D0,0.0D0,0.0D0,
     &0.0D0,0.0D0,1.0D0,0.0D0,0.0D0,
     &0.0D0,0.0D0,1.0D0,0.0D0,0.0D0
     &/
      DATA IMGE2/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,0.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA IMGE3/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &0.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA IMGE4/
     &0.0D0,0.0D0,1.0D0,1.0D0,0.0D0,
     &0.0D0,1.0D0,0.0D0,1.0D0,0.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,1.0D0,0.0D0,
     &0.0D0,0.0D0,0.0D0,1.0D0,0.0D0
     &/
      DATA IMGE5/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,0.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA IMGE6/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,0.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
```

```
     &1.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA IMGE7/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0
     &/
      DATA IMGE8/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA IMGE9/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0
     &/
C
C      Assign answer data;
C
      DATA ANS0/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA ANS1/
     &0.0D0,0.0D0,1.0D0,0.0D0,0.0D0,
     &0.0D0,0.0D0,1.0D0,0.0D0,0.0D0,
     &0.0D0,0.0D0,1.0D0,0.0D0,0.0D0,
     &0.0D0,0.0D0,1.0D0,0.0D0,0.0D0,
     &0.0D0,0.0D0,1.0D0,0.0D0,0.0D0
     &/
      DATA ANS2/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,0.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA ANS3/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &0.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA ANS4/
     &0.0D0,0.0D0,1.0D0,1.0D0,0.0D0,
     &0.0D0,1.0D0,0.0D0,1.0D0,0.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,1.0D0,0.0D0,
```

```
     &0.0D0,0.0D0,0.0D0,1.0D0,0.0D0
     &/
      DATA ANS5/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,0.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA ANS6/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,0.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA ANS7/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0
     &/
      DATA ANS8/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA ANS9/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0
     &/
C
C      Assign corrupted image data;
C
      DATA BAD0/
     &1.0D0,1.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA BAD1/
     &0.0D0,0.0D0,1.0D0,0.0D0,1.0D0,
     &0.0D0,0.0D0,1.0D0,1.0D0,0.0D0,
     &0.0D0,0.0D0,1.0D0,0.0D0,0.0D0,
     &0.0D0,0.0D0,0.0D0,1.0D0,0.0D0,
     &0.0D0,0.0D0,1.0D0,0.0D0,0.0D0
     &/
      DATA BAD2/
     &1.0D0,1.0D0,0.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,0.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,1.0D0,0.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
```

```
     &/
      DATA BAD3/
     &1.0D0,1.0D0,1.0D0,0.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,
     &0.0D0,1.0D0,1.0D0,1.0D0,0.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA BAD4/
     &0.0D0,0.0D0,1.0D0,0.0D0,0.0D0,
     &0.0D0,1.0D0,0.0D0,1.0D0,0.0D0,
     &1.0D0,1.0D0,0.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,1.0D0,1.0D0,0.0D0,
     &0.0D0,0.0D0,0.0D0,1.0D0,0.0D0
     &/
      DATA BAD5/
     &1.0D0,1.0D0,0.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,1.0D0,1.0D0,0.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA BAD6/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,1.0D0,0.0D0,
     &1.0D0,1.0D0,1.0D0,0.0D0,1.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,0.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA BAD7/
     &1.0D0,1.0D0,1.0D0,1.0D0,0.0D0,
     &0.0D0,1.0D0,0.0D0,0.0D0,0.0D0,
     &0.0D0,1.0D0,0.0D0,0.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0
     &/
      DATA BAD8/
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &1.0D0,1.0D0,0.0D0,0.0D0,1.0D0,
     &1.0D0,0.0D0,1.0D0,1.0D0,0.0D0,
     &1.0D0,1.0D0,0.0D0,1.0D0,0.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0
     &/
      DATA BAD9/
     &1.0D0,1.0D0,1.0D0,1.0D0,0.0D0,
     &1.0D0,0.0D0,0.0D0,0.0D0,0.0D0,
     &1.0D0,1.0D0,1.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,1.0D0,1.0D0,
     &0.0D0,0.0D0,0.0D0,0.0D0,1.0D0
     &/
C
C      Assign variables;
C
      LENGTH=5
      WIDTH=5
      DEPTH=1
      NUMIM=1
      REW1=0.9D0
      PUN1=0.1D0
      LAMBDA=0.01D0
```

```
      MAXCNT=10000
      TITLE='IMAGES'
      NAME='WEIGHTS'
C
C     Write out to file in correct format;
C
      WRITE(1,*)TITLE
      WRITE(1,*)LENGTH
      WRITE(1,*)WIDTH
      WRITE(1,*)DEPTH
      WRITE(1,*)NUMIM
      WRITE(1,*)NAME
C
C     Do this bit backwards to correspond with DATA;
C
      WRITE(1,*)((IMGE0(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      IF (NUMIM.GT.1) THEN
      WRITE(1,*)((IMGE1(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.2) THEN
      WRITE(1,*)((IMGE2(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.3) THEN
      WRITE(1,*)((IMGE3(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.4) THEN
      WRITE(1,*)((IMGE4(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.5) THEN
      WRITE(1,*)((IMGE5(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.6) THEN
      WRITE(1,*)((IMGE6(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.7) THEN
      WRITE(1,*)((IMGE7(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.8) THEN
      WRITE(1,*)((IMGE8(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.9) THEN
      WRITE(1,*)((IMGE9(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
C
C     Do this bit backwards to correspond with DATA;
C
      WRITE(1,*)((ANS0(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      IF (NUMIM.GT.1) THEN
      WRITE(1,*)((ANS1(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.2) THEN
      WRITE(1,*)((ANS2(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.3) THEN
      WRITE(1,*)((ANS3(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.4) THEN
      WRITE(1,*)((ANS4(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.5) THEN
```

```
      WRITE(1,*)((ANS5(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.6) THEN
      WRITE(1,*)((ANS6(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.7) THEN
      WRITE(1,*)((ANS7(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.8) THEN
      WRITE(1,*)((ANS8(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.9) THEN
      WRITE(1,*)((ANS9(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
C
C     Do this bit backwards to correspond with DATA;
C
      WRITE(1,*)((BAD0(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      IF (NUMIM.GT.1) THEN
      WRITE(1,*)((BAD1(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.2) THEN
      WRITE(1,*)((BAD2(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.3) THEN
      WRITE(1,*)((BAD3(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.4) THEN
      WRITE(1,*)((BAD4(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.5) THEN
      WRITE(1,*)((BAD5(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.6) THEN
      WRITE(1,*)((BAD6(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.7) THEN
      WRITE(1,*)((BAD7(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.8) THEN
      WRITE(1,*)((BAD8(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      IF (NUMIM.GT.9) THEN
      WRITE(1,*)((BAD9(I,J),J=0,WIDTH-1),I=LENGTH-1,0,-1)
      ENDIF
      STOP
      END
```

EXOR

– Input–Output vector data file

```
IMAGES
2,
1,
2,
4,
WEIGHTS
0.,0.,
1.,0.,
0.,1.,
1.,1.,
0.,0.,
1.,0.,
1.,0.,
0.,0.,
1.,1.,
0.,0.,
1.,1.,
1.,1.,
```

PARAM

– parameter file

```
.01,
5,
5,
2,
.5,
.50,
0.9,
0.1,
10000,
100,
1,
.5,
90,
1,
1,
1,
```

Example program run

– Exclusive OR using an A_{R-P} success/failure network

```
$Run -arp.exe+*nag+*ghost80 9=-9 98=-98
 Input file name...
 EXOR
 Defining Image...
 Defining Answer...
 Defining Corrupted Image...
 Closing file..f1..
 Closing file..f3..
 Do you want to run on Defaults?...
 Y
 Which set of defaults?...
 1
 Input title = exor
 File title = IMAGES
 Array Length = 2,
 Array Width = 1,
 Array Depth = 2,
 Image Length = 2,
 Image Width = 1,
 Number of images = 4,
 Dummy variable  =  WEIGHTS
 Rho  = .5,
 Temp = .5,
 Lambda = .1000000000000000E-01,
 Reward probability = .9,
 Punish probability = .1,
 Number of teaching trials = 100000,
 Number of post learning trials = 100,
 Maximum bound on number of attempts = 1,
 Proportion clamped in Hopfield case = .5,
 Length proportion clamped = 0,1,
 Width proportion clamped = 0,0,
 Confidence level (%) = 90.,
 No of program loops  = 1,
 Degree of Batching  = 1,
 Shift Register Length = 1,
 Defining Image...
 0.,0.,
 1.,0.,
 0.,1.,
 1.,1.,
 Defining Answer...
 0.,0.,
 1.,0.,
 1.,0.,
 0.,0.,
 Defining Corrupted Image...
 1.,1.,
 0.,0.,
 1.,1.,
 1.,1.,
  Plotting neural net connections DISABLED
  Block Plot ENABLED
  Plotting input/output pictures DISABLED
  Plotting weights DISABLED
  Global Reward scheme ENABLED
  Indirect Reward ENABLED
  Clipped Training Noise ENABLED
  Clipped Exercising Noise ENABLED
  Clipped Gaussian Noise ENABLED
```

```
 Plotting Noisy Image DISABLED
 Setting all initial weights to zero
 Normal Training
 Use of Random Inputs DISABLED
 Use of Null Output DISABLED
 Plotting Final Answer DISABLED
 Plotting Cash Values DISABLED
 Final Feedback DISABLED
 Use of Backpropagation Error Function ENABLED
 Use of Checksum Error Function DISABLED
 Asymmetric Weights ENABLED
 Normal Connection ENABLED
 Normal Error ENABLED
 Stimulus value = 1.0
 Normal Wii
 ARP Model ENABLED
 Randomising Weights ENABLED
 Deterministic model ENABLED
Training...IMAGES
   25000
Learning complete in 49981, cycles
Opening file... WEIGHTS
Recording weights...
Closing file...
Maximum Weight in array = 5.767002872341289,
Minimum Weight in array = -5.622133831327026,
Exercising array with Image...
      25
      50
      75
     100
Image number 0,Recognition %  100.,
      25
      50
      75
     100
Image number 1,Recognition %  98.,
      25
      50
      75
     100
Image number 2,Recognition %  100.,
      25
      50
      75
     100
Image number 3,Recognition %  97.,
Have learned correct response to 98.75, >90., %
Maximum number of program cycles (x100000,) = 1,
Exercising array with random image...
      25
      50
      75
     100
Incorrect Output %  4.,
Number of Images presented % 100.,
Number of correct answers %  96.,
Number of zero answers %  0.,
```

FORTRAN77 program ARP.MAIN

– main neural network simulation routine

```
C     ***********************************************
C     * Associative Reward-Punish Element (ARP)     *
C     * Image Recognition Program:                  *
C     * Program uses the Connectionist Model;       *
C     *                                             *
C     * - Program can deal with up to 10 images;    *
C     *   and has User limit of 5x5x4 arrays;       *
C     * - Training images may be presented in       *
C     *   'Batched' mode or singly;                 *
C     *   The depth of batching is set at compile   *
C     *   time;                                     *
C     * - Program loops round ITER times and zeroes *
C     *   the WEIGHT and SUMS array on each pass;   *
C     * - Program implements both GLOBAL and LOCAL  *
C     *   reward schemes although LOCAL should      *
C     *   only be used with DEPTH=1 ;               *
C     * - With the GLOBAL scheme, the choice exists *
C     *   to have direct encoding of the REWARD     *
C     *   line with a choice of error measure;      *
C     * - An optional routine has been included     *
C     *   at compilation time which will produce    *
C     *   error data as defined by the expression in *
C     *   the paper on Back Propagation;            *
C     * - This program seeks to recognise an        *
C     *   L1 x W1 Image using an ARP (or other type *
C     *   of element) in an array of size L x W x D ;*
C     * - Each ARP element is connected to all the  *
C     *   inputs on its level;                      *
C     * - Each ARP element has a single output to   *
C     *   the next plane of ARPs in exactly the same *
C     *   way ;                                     *
C     * - The program reads in general parameters   *
C     *   from a file called 'PARAM';               *
C     * - The program reads in the input parameters *
C     *   and image from a file called NAME1;       *
C     * - The program reads and records into a file *
C     *   called 'WEIGHTS', the current ARP element *
C     *   weights. In this way, the array states    *
C     *   can be examined and further runs performed *
C     *   from the conclusion of the previous one;  *
C     * - Symmetry can be imposed on the weights;   *
C     * - The program can be exercised without      *
C     *   training;                                 *
C     * - The program also reads in the desired     *
C     *   output answer image from the file, NAME1; *
C     * - If an incorrect image is presented,       *
C     *   the choice is given whereby the array is  *
C     *   required to output all zeroes (NULL);     *
C     * - ARP.PREP can be used to prepare a suitable *
C     *   input file;                               *
C     * - The program can be trained with a uniformly*
C     *   random component -either clipped to 0 or  *
C     *   1 or the actual random values between 0 and*
C     *   1;                                        *
C     * - The program can be exercised with a noisy *
C     *   image - the noise is Additive Gaussian    *
C     *   with zero mean and selectable Standard    *
C     *   Deviation either clipped to 0 or 1 or     *
C     *   using the actual random values clipped    *
C     *   between 0 and 1;                          *
C     * - The random input can be fixed on the first *
C     *   iteration and the output of the array     *
C     *   fed back to the input. It is hoped to     *
C     *   show in this case that the output settles *
C     *   to the nearest stable state when combined *
```

```
C     *   with a GLOBAL Direct Reward scheme and     *
C     *   NO NULL response (which would otherwise    *
C     *   erase the input!);                         *
C     * - A corrupted image data set is included     *
C     *   in the data file for this purpose;         *
C     * - A HOPFIELD, ARP or BACK prop model can be  *
C     *   selected. Note that the Hopfield model     *
C     *   requires that the HOPFLG is set to 0;      *
C     * - The Backpropagation model can operate      *
C     *   in a stochastic mode;                      *
C     * - Variable clamping is possible with the     *
C     *   PROPOR facility;                           *
C     * - The final array output may be plotted;     *
C     *                                              *
C     * - Variables; -                               *
C     *   FRAME  - input picture  (0:L-1,0:W-1,1)    *
C     *   FRAME  - output picture (0:L-1,0:W-1,D+1)  *
C     *   LENGTH - Vertical array length;            *
C     *   WIDTH  - Horizontal array length;          *
C     *   DEPTH  - Array depth;                      *
C     *   LEN1   - Vertical image length;            *
C     *   WID1   - Horizontal image length;          *
C     * - Program uses DOUBLE PRECISION variables;   *
C     * - Subroutines called;                        *
C     *   BLOCK DATA INIT in ARP.ZERO                *
C     *   QUEST in ARP.QUEST                         *
C     *   CONFIG in ARP.CONFIG                       *
C     *   PICT in ARP.PICT                           *
C     *   LINK in ARP.LINK                           *
C     *   LINE in ARP.LINE                           *
C     *   CAPT in ARP.CAPT                           *
C     *   COMP in ARP.COMP                           *
C     *   COMP1 in ARP.COMP1                         *
C     *   ERR in ARP.ERROR (Optional at Compile Time)*
C     *   ARRAY in ARP.CONNECT (Arp)                 *
C     *   HOPMOD in ARP.CONHOP  (Hopfield)           *
C     *   PROP in ARP.PROP (Back Propagation)        *
C     *   SYMM  in ARP.SYMM                          *
C     *   MATCH in ARP.MATCH                         *
C     *   NPLOT in ARP.NPLOT                         *
C     *   ELPLOT in ARP.ELPLOT                       *
C     *   ARP in ARP.NEURON (25  input ARP)          *
C     *   ELEM in ARP.BACKP (25  input Back Propag)  *
C     *   HOP in ARP.HOP (25  input Hopfield)        *
C     *   *NAG                                       *
C     *   *GHOST80                                   *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created:17th May  1987                       *
C     * Update :11th May  1988                       *
C     * Frozen :24th July 1988                       *
C     ************************************************
      PROGRAM ARPSIM
C
C     Define everything as implicit REAL*8
C
      IMPLICIT REAL*8(A - H,Q - Z)
C
C     Define X,Y from 0 to maximum-1;
C     Define Z from 1 to maximum;
C
C     Define X input for ARP elements;
C     26 inputs (25 + stimulus) vs 4 levels;
C
```

```
      REAL*8 XIN(0:25,4)
C
C      Image to be recognised;
C
      REAL*8 IMAGE(10,0:4,0:4)
C
C      Null Image;
C
      REAL*8 ZERO(10,0:4,0:4)
C
C      Required output;
C
      REAL*8 ANSWER(10,0:4,0:4)
C
C      Corrupted image for optional test;
C
      REAL*8 BADPIC(10,0:4,0:4)
C
C      Define Frame - Max X x Y x Z ;
C
C      FRAME(0:X-1,0:Y-1,Z+1)
C
C      Final Frame N+1 is the output;
C
      REAL*8 FRAME(0:4,0:4,5)
C
C      Define Weights for X x Y ARP elements with Z  Frames;
C
      REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
C
C      Define Delta Weights for X x Y ARP elements with Z  Frames;
C
      REAL*8 DWGTS(0:4,0:4,4,0:4,0:4)
C
C      Define Delta Weights for X x Y ARP elements with Z  Frames;
C      Acceleration values;
C
      REAL*8 DWGTS1(0:4,0:4,4,0:4,0:4)
C
C      Define Stimulus array for X x Y ARP elements with Z frames;
C
      REAL*8 STIM(0:4,0:4,4)
C
C      Define Delta Stim array for X x Y ARP elements with Z frames;
C
      REAL*8 DSTIM(0:4,0:4,4)
C
C      Define Delta Stim array for X x Y ARP elements with Z frames;
C      (Acceleration value)
C
      REAL*8 DSTIM1(0:4,0:4,4)
C
C      Define storage array for an ARP element over 10000 trials;
C      Real*4 for GHOST plots;
C
C      REAL*4 VALUES(0:4,0:4,10000)
C
C      Define storage array for Stimulus values 10000 trials;
C      Real*4 for GHOST plots;
C
C      REAL*4 STIVAL(10000)
C
C      Define storage array for noise trials;
```

```
C     Real*4 for GHOST plots;
C
      REAL*4 PLNOI(0:10000)
C
C     Define storage array for CASH plots;
C     Real*4 for GHOST plots;
C
      REAL*4 CSHPLT(10000)
C
C     Define shift arrays for testing error;
C     Tests over ISLOT cycles;
C
      REAL*8 ERRCHK(500)
      REAL*8 SPECIA(50)
C
C     Save SUMs for X x Y ARP elements, Z Frames ;
C
      REAL*8 SUMS(0:4,0:4,4)
C
C     Real *8 variables;
C
      REAL*8 RHO, TEMP, LAMBDA, PN, G05CAF, PREVAL(10), NOWVAL(10)
      REAL*8 CASH, REW1, PUN1, TRY, STADEV, TRYAV, PROPOR, CONFD, CONVAL
      REAL*8 MAXWGT, MINWGT
C
C     Integers;
C
      INTEGER X, Y, Z, NUMEL, RESULT, NUMIM, BAT
      INTEGER MAXCNT, XELEM, YELEM, ZELEM
      INTEGER LENGTH, WIDTH, DEPTH
      INTEGER LEN1, WID1
      INTEGER LEVEL, CHECK, ISLOT, ITER, PROG
      INTEGER MULT, MULT1, EXERC
      INTEGER RESFLG, CLIFLG, CLEFLG, GLOFLG, NEUFLG, DIRFLG, WGTFLG
      INTEGER TRAFLG, CLGFLG, NOIFLG, DUMFLG, TRYFLG, MULFLG
      INTEGER NULFLG, FINFLG, CONFLG, CSHFLG, ENEFLG, SYMFLG, HOPFLG
      INTEGER GRAFLG, STIFLG, INPFLG, PICFLG, BOLFLG, LINFLG, BRKFLG,
     &        STOFLG
      INTEGER TRACNT, EXECNT, RANCNT, GAUCNT, NOICNT, FDBCNT, IMCNT,
     &        GOCNT, HOPCNT
      INTEGER RANRES, ZERRES, GAURES, CORRES, IMARES, GOMAX
C
C     Define EXERES as an array to hold results for exercising up
C     to 10 input images;
C
      INTEGER EXERES(10)
C
C     Integer array to hold Rewards for each element - final frame only;
C
      INTEGER REWARD(0:4,0:4)
C
C     Alternative for Global Rewards;
C
      INTEGER GLOREW
C
C     Real array to hold Results for each element - final frame only;
C     This element holds the error value for each output- on each level
C     Visible elements are in DEPTH;
C     Maximum error +- 1.0
C
      REAL*8 CORECT(0:4,0:4,0:4)
C
C     Frame shift register - the 0th address holds the averaged value;
C
```

```
      REAL*8 SHIFT1(0:4,0:4,4,0:20)
C
C      Error shift register;
C
      REAL*8 SHIFT2(0:4,0:4,4,0:20)
C
C      Summer shift register;
C
      REAL*8 SHIFT3(0:4,0:4,4,0:20)
C
C      Weights shift register;
C
      REAL*8 SHIFT4(0:4,0:4,4,0:25)
C
C      Characters;
C
      CHARACTER*70 TITLE
      CHARACTER*30 TITLE1
      CHARACTER*30 TITLE2
      CHARACTER*30 TITLE3
      CHARACTER*30 TITLE4
      CHARACTER*10 NAME1, DUMMY
C
C      TITLE  = Title of job;
C      TITLE1 = Title of ARP input image plot;
C      TITLE2 = Title of ARP output answer image plot;
C      TITLE3 = Title of ARP corrupted image;
C      TITLE4 = Title of ARP uncorrupted output?;
C
C      Define Common Block One;
C
      COMMON /ONE/ FRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
     &        REWARD, GLOREW
C
C      Define Common Block Two;
C
      COMMON /TWO/ RHO, TEMP, LAMBDA, PN
C
C      Define Common Block Three;
C
      COMMON /THREE/ XIN
C
C      Define Common Block Four;
C
      COMMON /FOUR/ XELEM, YELEM, ZELEM, RESFLG, CLIFLG, CLEFLG, GLOFLG,
     &        NEUFLG, DIRFLG, WGTFLG, TRAFLG, CLGFLG, NOIFLG, DUMFLG,
     &        TRYFLG, NULFLG, FINFLG, CONFLG, MULFLG, CSHFLG, ENEFLG,
     &        SYMFLG, HOPFLG, GRAFLG, STIFLG, INPFLG, PICFLG, BOLFLG,
     &        LINFLG, BRKFLG, STOFLG, NUMIM, NOINUM, EXERC, GOMAX, LEN1,
     &        WID1, TITLE, TITLE1, TITLE2, TITLE3, TITLE4, NAME1, DUMMY
C
C      Define Common Block Five;
C
      COMMON /FIVE/ SHIFT1, SHIFT2, SHIFT3, SHIFT4, ISHIFT, IBCNT, BAT
C
C      Initialise Delta weights;
C
      DATA DWGTS /2500*0.0D0/
      DATA DWGTS1 /2500*0.0D0/
C
C      Initialise Delta stimulus;
C
```

```
      DATA DSTIM /100*0.0D0/
      DATA DSTIM1 /100*0.0D0/
C
C     Initialise Input image array;
C
      DATA IMAGE /250*0.0D0/
C
C     Initialise Error Averaging with nonzero value;
C
      DATA ERRCHK /500*1.0D0/
      DATA SPECIA /50*1.0D0/
C
C     Initialise Null image array;
C
      DATA ZERO /250*0.0D0/
C
C     Initialise Output required answer array;
C
      DATA ANSWER /250*0.0D0/
C
C     Initialise corrupted picture array;
C
      DATA BADPIC /250*0.0D0/
C
C     Initialise Correct result array;
C
      DATA CORECT /125*0.0D0/
C
C     Initialise PLNOI plot array;
C
      DATA PLNOI /10001*0.0/
C
C     Initialise CSHPLT plot array;
C
      DATA CSHPLT /10000*0.0/
C
C     Initialise Random Number Generator for non repeatable sequences;
C
      CALL G05CCF
C
C     Initialise Random Number Generator for repeatable sequences;
C
C     CALL G05CBF(1)
C
C     Disable limit checking on plots;
C
      CALL GPSTOP(0)
C
C     Call question subroutine to input most things;
C
      CALL QUEST(IMAGE, ANSWER, BADPIC, MAXCNT, REW1, PUN1, PROPOR,
     &     CONFD, ITER, BAT, ISHIFT)
C
C     Call Configuration routine to confirm;
C
      CALL CONFIG(IMAGE, ANSWER, BADPIC, MAXCNT, REW1, PUN1, PROPOR,
     &     CONFD, ITER, BAT, ISHIFT)
C
C     Define variable values;
C
```

```
      PN = 1.0D0
      NUM = 1
      MULT = MAXCNT / 4
      MULT1 = EXERC / 4
C
C     Define ISLOT as the results averaging interval;
C
      ISLOT = 500
C
C     Plot images;
C
C
      DO 1 IMCNT = 1, NUMIM
        PREVAL(IMCNT) = 0.0D0
        NOWVAL(IMCNT) = 0.0D0
        IF (PICFLG .EQ. 1) THEN
          CALL PICT(IMAGE, LEN1, WID1, TITLE1, IMCNT)
C
C     Plot answers;
C
          CALL PICT(ANSWER, LEN1, WID1, TITLE2, IMCNT)
        END IF
    1 CONTINUE
C
C
C     Compute the number of elements for one frame;
C
      NUMEL = LENGTH * WIDTH
C
C     Start Iterations from here;
C
      DO 43 PROG = 1, ITER
C
C     Reinitialise error check to non zero value;
C
        DO 2 II = 1, ISLOT
          ERRCHK(II) = 1.0D0
    2   CONTINUE
C
C     Check to see if weights recorded in weight file before;
C
        IF (DUMFLG .EQ. 1) THEN
          WRITE (6,*)'Opening file...', DUMMY
          OPEN (2,FILE='WEIGHTS')
          READ (2,*) (((((WEIGHT(I,J,K,L,M),I=0,LENGTH - 1),J=0,WIDTH -
     &          1),K=1,DEPTH),L=0,LENGTH - 1),M=0,WIDTH - 1)
          READ (2,*) (((STIM(I,J,K),I=0,LENGTH - 1),J=0,WIDTH - 1),K=1,
     &          DEPTH)
C
          WRITE (6,*)'Closing file...'
C
C     Close file;
C
          REWIND 2
          CLOSE (2)
        ELSE
C
          DO 7 I = 0, LENGTH - 1
C
            DO 6 J = 0, WIDTH - 1
C
```

```
                DO 5 K = 1, DEPTH
                  PN = G05CAF(PN)
                  IF (BRKFLG .EQ. 1) THEN
                    STIM(I,J,K) = (2.0D0*PN - 1.0D0) * 1.0D0
                    SUMS(I,J,K) = 0.0D0
                  ELSE
                    STIM(I,J,K) = 0.0D0
                    SUMS(I,J,K) = 0.0D0
                  END IF
C
                  DO 4 L = 0, LENGTH - 1
C
                    DO 3 M = 0, WIDTH - 1
                      PN = G05CAF(PN)
                      IF (BRKFLG .EQ. 1) THEN
                        WEIGHT(I,J,K,L,M) = (2.0D0*PN - 1.0D0) * 1.0D0
                      ELSE
                        WEIGHT(I,J,K,L,M) = 0.0D0
                      END IF
    3               CONTINUE
C
    4             CONTINUE
C
    5           CONTINUE
C
    6         CONTINUE
C
    7       CONTINUE
C
          END IF
C
C       Option - Draw Neural Net before doing anything;
C
          IF ((NEUFLG .EQ. 1) .AND. (TRYFLG .EQ. 1)) THEN
            WRITE (6,*)'Plotting initial weights...'
            IF (LINFLG .EQ. 1) THEN
              CALL LINK(0, BOLFLG)
            ELSE
              CALL LINE(0, BOLFLG)
            END IF
          END IF
C
C       Initialise GOCNT;
C
          GOCNT = 1
          IF (TRYFLG .EQ. 1) GO TO 33
C
C       Begin training;
C
    8     WRITE (6,*)'Training...', TITLE
C
C
          DO 26 TRACNT = 1, MAXCNT
C
C       Print out TRACNT in multiples of MULT;
C
            IF (MOD(TRACNT,MULT) .EQ. 0) THEN
              WRITE (6,74) TRACNT
C
C       Option - Draw Neural Net;
C
```

```
            IF (NEUFLG .EQ. 1) THEN
              IF (LINFLG .EQ. 1) THEN
                CALL LINK(TRACNT, BOLFLG)
              ELSE
                CALL LINE(TRACNT, BOLFLG)
              END IF
            END IF
C
C     Call Random Number Generator - If less than 0.5 then
C     Training pattern is Image, otherwise Random;
C
C     Set TRAFLG=1 if Image is selected for presentation;
C
            PN = G05CAF(PN)
C
C     If MULFLG=1 then only shuffle through the input progs;
C
            IF (MULFLG .EQ. 1) PN = PN / 2.0D0
            IF (PN .LE. 0.5D0) THEN
              TRAFLG = 1
C
C      Generate random selection of the image;
C      Defaults to 1 if NUMIM is 1;
C
              NUM = NINT(G05CAF(PN)*DFLOAT(NUMIM - 1)) + 1
C
              DO 10 X = 0, LEN1 - 1
C
                DO 9 Y = 0, WID1 - 1
                  FRAME(X,Y,1) = IMAGE(NUM,X,Y)
   9            CONTINUE
C
  10          CONTINUE
C
            ELSE
              TRAFLG = 0
C
              DO 12 X = 0, LEN1 - 1
C
                DO 11 Y = 0, WID1 - 1
                  PN = G05CAF(PN)
C
C     Generate Random 1's and 0's
C
C     If CLIFLG=1 then clip, otherwise not;
C
                  IF (CLIFLG .EQ. 1) THEN
                    FRAME(X,Y,1) = DNINT(PN)
                  ELSE
                    FRAME(X,Y,1) = PN
                  END IF
  11            CONTINUE
C
  12          CONTINUE
C
            END IF
C
C     Analyse image - is it the right pattern?
C     If Image is greater than 5 x 5 then
C     chance of random generation of the correct pattern
C     is small...use TRAFLG to tell if image is there;
C
```

```
         IF (NUMEL .LE. 25) THEN
C
C     Do Check loop;
C
           LEVEL = 1
           CALL COMP(IMAGE, LEVEL, RESULT, NUM)
         ELSE
C
C     Set Result = Flag ;
C
           RESULT = TRAFLG
         END IF
C
C     Commence batching here;
C
         DO 21 IBACNT = 1, BAT
           IBCNT = IBACNT
C
C     If HOPFIELD flag is set then introduce recurrent connection here;
C     Call array;
C
           IF (BOLFLG .EQ. 0) THEN
             CALL ARRAY(0, GLOFLG, STIFLG)
           END IF
           IF (BOLFLG .EQ. 1) THEN
             CALL HOPMOD(1, GLOFLG, STIFLG, LEN1, WID1, PROPOR)
           END IF
           IF (BOLFLG .EQ. 2) THEN
             CALL PROP(0, GLOFLG, STIFLG, CORECT, NUMIM, TRACNT, DWGTS,
     &            DSTIM, DWGTS1, DSTIM1, STOFLG)
           END IF
C
           DO 14 X = 0, LEN1 - 1
C
             DO 13 Y = 0, WID1 - 1
               FRAME(X,Y,1) = IMAGE(NUM,X,Y)
  13         CONTINUE
C
  14       CONTINUE
C
           LEVEL = DEPTH + 1
           IF (HOPFLG .EQ. 1) THEN
C
             DO 16 X = LEN1, LENGTH - 1
C
               DO 15 Y = WID1, WIDTH - 1
                 FRAME(X,Y,1) = FRAME(X,Y,LEVEL)
  15           CONTINUE
C
  16         CONTINUE
C
           END IF
C
C     If GLOFLG=1 then Global Reward otherwise Local;
C     For local case;
C     Check to see if correct image was presented
C     and examine each elements output using COMP1 for correct
C     result on an individual basis - CORECT array holds
C     each elements Real  'Marks' 0 = zero error, +-1=total error;
C     Note: if a NULL response is not required (NULFLG=0)
C     then the CHECK value will be set to a value 99
C     and the CASH value will be set to either REW1 or PUN1 with
C     probability 0.5 (heads or tails);
```

```
C
C     Initialise CHECK to avoid problems on Reward;
C
          CHECK = 0
C
          IF (RESULT .EQ. 1) THEN
            LEVEL = DEPTH + 1
            IF (GLOFLG .EQ. 1) THEN
              CALL COMP(ANSWER, LEVEL, CHECK, NUM)
            ELSE
              CALL COMP1(ANSWER, LEVEL, CORECT, NUM)
            END IF
          ELSE
C
C     Incorrect image presented;
C
            LEVEL = DEPTH + 1
            IF (GLOFLG .EQ. 1) THEN
C
C     Check NULFLG setting;
C
              IF (NULFLG .EQ. 0) THEN
                CHECK = 99
              ELSE
                CALL COMP(ZERO, LEVEL, CHECK, NUM)
              END IF
            ELSE
C
C     Check NULFLG setting - although CHECK is not used explicitly
C     here - take advantage of the variable as setting CORECT is
C     difficult (its an array);
C
              IF (NULFLG .EQ. 0) THEN
                CHECK = 99
              ELSE
                CALL COMP1(ZERO, LEVEL, CORECT, NUM)
              END IF
            END IF
          END IF
C
C     Encode Reward array for each element;
C
C     If GLOFLG=1 then Global Reward otherwise Local;
C     If DIRFLG=1 then Direct Reward otherwise Indirect;
C
          IF (GLOFLG .EQ. 1) THEN
            IF (DIRFLG .EQ. 1) THEN
C
C     Option - do gradient ascent;
C     Incorporating a top of maxima Reward;
C     NB. No isolation provided for multiple images;
C
              IF (GRAFLG .EQ. 1) THEN
                CALL MATCH(ANSWER, LEVEL, NOWVAL(NUM), NUM, ENEFLG)
                IF (NOWVAL(NUM) .GT. PREVAL(NUM)) THEN
                  IF (NOWVAL(NUM) .EQ. 1.0D0) THEN
C
C     Make definite REWARD here to ensure lock-on;
C
                    CASH = 1.0D0
                  ELSE
                    CASH = REW1
                  END IF
```

```
                    ELSE
                      IF (NOWVAL(NUM) .EQ. 1.0D0) THEN
                        CASH = 1.0D0
                      ELSE
                        CASH = PUN1
                      END IF
                    END IF
                    PREVAL(NUM) = NOWVAL(NUM)
                  ELSE
                    CALL MATCH(ANSWER, LEVEL, CASH, NUM, ENEFLG)
                  END IF
                ELSE
                  IF (CHECK .EQ. 1) THEN
                    CASH = REW1
                  ELSE
                    CASH = PUN1
                  END IF
C
C     Check here for CHECK=99;
C
                  IF (CHECK .EQ. 99) THEN
                    PN = G05CAF(PN)
                    IF (PN .GT. 0.5D0) THEN
                      CASH = REW1
                    ELSE
                      CASH = PUN1
                    END IF
                  END IF
                END IF
C
C      Begin Rewarding the array;- record these values;
C      Note 10000 point limit;
C
                IF (CSHFLG .EQ. 1) CSHPLT(TRACNT) = CASH
                PN = G05CAF(PN)
                IF (PN .LT. CASH) THEN
                  GLOREW = 1
                ELSE
                  GLOREW = 0
                END IF
              ELSE
C
                DO 18 I = 0, LEN1 - 1
C
                  DO 17 J = 0, WID1 - 1
C
C     Check here for CHECK=99;
C
                    IF (CHECK .EQ. 99) THEN
                      PN = G05CAF(PN)
                      IF (PN .GT. 0.5D0) THEN
                        CASH = REW1
                      ELSE
                        CASH = PUN1
                      END IF
                    ELSE
C
C      Use nearest integer to CORECT value;
C
                      IF (NINT(CORECT(DEPTH,I,J)) .EQ. 0) THEN
```

```
                    CASH = REW1
                  ELSE
                    CASH = PUN1
                  END IF
                END IF
                PN = G05CAF(PN)
                IF (PN .LT. CASH) THEN
                  REWARD(I,J) = 1
                ELSE
                  REWARD(I,J) = 0
                END IF
  17          CONTINUE
C
  18        CONTINUE
C
          END IF
C
C     Update Weights;
C
C     Call array;
C
          IF (BOLFLG .EQ. 0) THEN
            CALL ARRAY(1, GLOFLG, STIFLG)
          END IF
          IF (BOLFLG .EQ. 2) THEN
            CALL PROP(1, GLOFLG, STIFLG, CORECT, NUMIM, TRACNT, DWGTS,
     &           DSTIM, DWGTS1, DSTIM1, STOFLG)
          END IF
C
C     Option - Impose symmetry;
C
          IF (SYMFLG .EQ. 1) CALL SYMM(INPFLG)
C
C     Option - Put ARP element no. elem WEIGHT into storage array;
C     remembering that Y is Length and X is Width;
C
          IF (WGTFLG .EQ. 1) THEN
C
            DO 20 L = 0, LENGTH - 1
C
              DO 19 M = 0, WIDTH - 1
C               VALUES(L,M,TRACNT) = WEIGHT(YELEM,XELEM,ZELEM,L,M)
C               STIVAL(TRACNT) = STIM(YELEM,XELEM,ZELEM)
  19          CONTINUE
C
  20        CONTINUE
C
          END IF
C
C     Conclude batching;
C
  21    CONTINUE
C
C     Take average over ISLOT of the shifted average;
C
        CALL ERR(ANSWER, ERRVAL, ALTER, NUM)
C
C     Take moving average of ERRVAL to gauge relative performance;
C     Performed over 50 slots;
C
```

```
          DO 22 ICLOCK = (50 - 1), 1, -1
            SPECIA(ICLOCK + 1) = SPECIA(ICLOCK)
   22     CONTINUE
C
          SPECIA(1) = ERRVAL
C
C     Now average over ISLOT by computing average success rate;
C
          ERRVAL = 0
C
          DO 23 JCLOCK = 1, 50
            ERRVAL = ERRVAL + SPECIA(JCLOCK)
   23     CONTINUE
C
          ERRVAL = ERRVAL / 50.0D0
C
C     Option: now remove comment to enable this;
C
C          WRITE(99,*)ERRVAL
C
          DO 24 ICLOCK = (ISLOT - 1), 1, -1
            ERRCHK(ICLOCK + 1) = ERRCHK(ICLOCK)
   24     CONTINUE
C
          ERRCHK(1) = ALTER
C
C     Now average over ISLOT by computing average success rate;
C
          IVALER = 0
C
          DO 25 JCLOCK = 1, ISLOT
            IF (ERRCHK(JCLOCK) .GT. 0.0D0) THEN
              IVALER = IVALER + 0
            ELSE
              IVALER = IVALER + 1
            END IF
   25     CONTINUE
C
          VALERR = DFLOAT(IVALER) / DFLOAT(ISLOT) * 100.0D0
C
C     If VALERR is >CONFD then the array has learned the solution;
C
          IF (DABS(VALERR) .GE. CONFD) THEN
            WRITE (6,*)'Learning complete in ', TRACNT + (GOCNT - 1) *
     &           MAXCNT, ' cycles'
            WRITE (98,*) TRACNT + (GOCNT - 1) * MAXCNT
            GO TO 27
          END IF
   26    CONTINUE
C
         IF (ITER .NE. 1) WRITE (98,*) MAXCNT
C
C     Record Weights;
C
C     Open the Weight record file
C
   27    WRITE (6,*)'Opening file...', DUMMY
         OPEN (2,FILE='WEIGHTS')
         WRITE (6,*)'Recording weights...'
C
```

```
      WRITE (2,*) (((((WEIGHT(I,J,K,L,M),I=0,LENGTH - 1),J=0,WIDTH -
     &       1),K=1,DEPTH),L=0,LENGTH - 1),M=0,WIDTH - 1)
      WRITE (2,*) (((STIM(I,J,K),I=0,LENGTH - 1),J=0,WIDTH - 1),K=1,
     &       DEPTH)
C
      WRITE (6,*)'Closing file...'
C
C     Close file;
C
      REWIND 2
      CLOSE (2)
C
C     Compute Maximum and Minimum weights;
C
      MAXWGT = 0.0D0
      MINWGT = 0.0D0
C
      DO 32 IM = 0, LENGTH - 1
C
        DO 31 JM = 0, WIDTH - 1
C
          DO 30 KM = 1, DEPTH
C
            DO 29 LM = 0, LENGTH - 1
C
              DO 28 MM = 0, WIDTH - 1
                MAXWGT = DMAX1(WEIGHT(IM,JM,KM,LM,MM),MAXWGT,STIM(IM,
     &          JM,KM),STIM(LM,MM,KM))
                MINWGT = DMIN1(WEIGHT(IM,JM,KM,LM,MM),MINWGT,STIM(IM,
     &          JM,KM),STIM(LM,MM,KM))
   28         CONTINUE
C
   29       CONTINUE
C
   30     CONTINUE
C
   31   CONTINUE
C
   32 CONTINUE
C
C
C
C
C     Output results;
C
      WRITE (6,*)'Maximum Weight in array = ', MAXWGT
      WRITE (6,*)'Minimum Weight in array = ', MINWGT
C
C     Exercise array;- first with all Images;
C
   33 WRITE (6,*)'Exercising array with Image...'
C
      TRYAV = 0.0D0
C
      DO 42 IMCNT = 1, NUMIM
C
        DO 35 X = 0, LEN1 - 1
C
          DO 34 Y = 0, WID1 - 1
            EXERES(IMCNT) = 0
```

```
                FRAME(X,Y,1) = IMAGE(IMCNT,X,Y)
  34          CONTINUE
C
  35        CONTINUE
C
C
           DO 41 EXECNT = 1, EXERC
C
C       Print out EXECNT in multiples of MULT1;
C
             IF (MOD(EXECNT,MULT1) .EQ. 0) WRITE (6,74) EXECNT
C
C       Option - Do Hopfield Style model;
C
             IF (HOPFLG .EQ. 1) THEN
               LEVEL = DEPTH + 1
C
               DO 40 HOPCNT = 1, EXERC
C
C       Call array;
C
                  IF (BOLFLG .EQ. 0) THEN
                    CALL ARRAY(0, GLOFLG, STIFLG)
                  END IF
                  IF (BOLFLG .EQ. 1) THEN
                    CALL HOPMOD(0, GLOFLG, STIFLG, LEN1, WID1, PROPOR)
                  END IF
                  IF (BOLFLG .EQ. 2) THEN
                    CALL PROP(0, GLOFLG, STIFLG, CORECT, NUMIM, TRACNT,
     &                   DWGTS, DSTIM, DWGTS1, DSTIM1, STOFLG)
                  END IF
C
C       Feed back everything;
C
                  DO 37 X = 0, LENGTH - 1
C
                    DO 36 Y = 0, WIDTH - 1
                      FRAME(X,Y,1) = FRAME(X,Y,LEVEL)
  36                CONTINUE
C
  37                CONTINUE
C
C       Now overwrite - if PROPOR is greater than 0.0;
C
                  IF (PROPOR .GT. 0.00D0) THEN
C
                    DO 39 X = 0, NINT(DFLOAT(LEN1 - 1)*PROPOR)
C
                      DO 38 Y = 0, NINT(DFLOAT(WID1 - 1)*PROPOR)
                        FRAME(X,Y,1) = IMAGE(IMCNT,X,Y)
  38                    CONTINUE
C
  39                  CONTINUE
C
                  END IF
C
  40            CONTINUE
C
             ELSE
C
C       Call array;
C
```

```
              IF (BOLFLG .EQ. 0) THEN
                CALL ARRAY(0, GLOFLG, STIFLG)
              END IF
              IF (BOLFLG .EQ. 1) THEN
                CALL HOPMOD(0, GLOFLG, STIFLG, LEN1, WID1, PROPOR)
              END IF
              IF (BOLFLG .EQ. 2) THEN
                CALL PROP(0, GLOFLG, STIFLG, CORECT, NUMIM, TRACNT,
     &               DWGTS, DSTIM, DWGTS1, DSTIM1, STOFLG)
              END IF
            END IF
C
C     Check Answer;
C
            LEVEL = DEPTH + 1
            CALL COMP(ANSWER, LEVEL, CHECK, IMCNT)
            IF (CHECK .EQ. 1) EXERES(IMCNT) = EXERES(IMCNT) + 1
  41      CONTINUE
C
          TRY = DFLOAT(EXERES(IMCNT)) * 100.0D0 / DFLOAT(EXERC)
          WRITE (6,*)'Image number', ' ', IMCNT - 1, 'Recognition %
     &          TRY
          TRYAV = TRYAV + TRY
  42    CONTINUE
C
C       If this is less than CONFD% then program needs to try again
C
        CONVAL = TRYAV / DFLOAT(NUMIM)
        IF (ITER .EQ. 1) THEN
          IF (CONVAL .LT. CONFD) THEN
            WRITE (6,*)'Not right answer...trying again'
            GOCNT = GOCNT + 1
            IF (GOCNT .GT. GOMAX) THEN
              WRITE (6,*)'Have exceeded ', GOMAX, ' learning cycles...ex
     &iting..'
C
C     Record maximum learning cycle in f98 file;
C
              WRITE (98,*) DFLOAT(GOMAX) * DFLOAT(MAXCNT - 1)
            ELSE
              IF (ITER .EQ. 1) GO TO 8
            END IF
          ELSE
            WRITE (6,*)'Have learned correct response to ', CONVAL,
     &           ' >', CONFD, ' %'
            WRITE (6,*)'Maximum number of program cycles (x', MAXCNT,
     &           ') = ', GOCNT
          END IF
        END IF
  43 CONTINUE
C
C     Exercise array;- now with Random Image;
C
      WRITE (6,*)'Exercising array with random image...'
C
      RANRES = 0
      IMARES = 0
      CORRES = 0
      ZERRES = 0
C
```

```
      DO 53 RANCNT = 1, EXERC
C
C     Print out RANCNT in multiples of MULT1;
C
       IF (MOD(RANCNT,MULT1) .EQ. 0) WRITE (6,74) RANCNT
C
       DO 45 X = 0, LEN1 - 1
C
         DO 44 Y = 0, WID1 - 1
C
C     Generate Random 1's and 0's
C     If CLEFLG=1 then clip;
C
           PN = G05CAF(PN)
C
           IF (CLEFLG .EQ. 1) THEN
             FRAME(X,Y,1) = DNINT(PN)
           ELSE
             FRAME(X,Y,1) = PN
           END IF
  44     CONTINUE
C
  45   CONTINUE
C
C     Analyse image - is it the right pattern?
C     If Image is greater than 5 x 5 then
C     chance of random generation of the correct pattern
C     is small...and can be ignored;
C     Set Result default here;
C
       RESULT = 0
       IF (NUMEL .LE. 25) THEN
C
C     Do Check loop;
C
         LEVEL = 1
C
C     Check all images;
C
         DO 46 IMCNT = 1, NUMIM
           CALL COMP(IMAGE, LEVEL, RESULT, IMCNT)
C
C     This part carries IMCNT over;
C
           IF (RESULT .EQ. 1) GO TO 47
  46     CONTINUE
C
C       Compute number of occurrences of correct image;
C
  47     IF (RESULT .EQ. 1) IMARES = IMARES + 1
       END IF
C
C     Option - Do Hopfield Style model;
C
       IF (HOPFLG .EQ. 1) THEN
         LEVEL = DEPTH + 1
C
         DO 52 HOPCNT = 1, EXERC
C
C     Call array;
C
```

```
            IF (BOLFLG .EQ. 0) THEN
              CALL ARRAY(0, GLOFLG, STIFLG)
            END IF
            IF (BOLFLG .EQ. 1) THEN
              CALL HOPMOD(0, GLOFLG, STIFLG, LEN1, WID1, PROPOR)
            END IF
            IF (BOLFLG .EQ. 2) THEN
              CALL PROP(0, GLOFLG, STIFLG, CORECT, NUMIM, TRACNT, DWGTS,
     &             DSTIM, DWGTS1, DSTIM1, STOFLG)
            END IF
C
C     Feed back everything;
C
            DO 49 X = 0, LENGTH - 1
C
              DO 48 Y = 0, WIDTH - 1
                FRAME(X,Y,1) = FRAME(X,Y,LEVEL)
   48         CONTINUE
C
   49       CONTINUE
C
C     Now overwrite - if PROPOR is greater than 0.0;
C
            IF (PROPOR .GT. 0.00D0) THEN
C
              DO 51 X = 0, NINT(DFLOAT(LEN1 - 1)*PROPOR)
C
                DO 50 Y = 0, NINT(DFLOAT(WID1 - 1)*PROPOR)
                  FRAME(X,Y,1) = IMAGE(IMCNT,X,Y)
   50           CONTINUE
C
   51         CONTINUE
C
            END IF
   52     CONTINUE
C
        ELSE
C
C     Call array;
C
          IF (BOLFLG .EQ. 0) THEN
            CALL ARRAY(0, GLOFLG, STIFLG)
          END IF
          IF (BOLFLG .EQ. 1) THEN
            CALL HOPMOD(0, GLOFLG, STIFLG, LEN1, WID1, PROPOR)
          END IF
          IF (BOLFLG .EQ. 2) THEN
            CALL PROP(0, GLOFLG, STIFLG, CORECT, NUMIM, TRACNT, DWGTS,
     &           DSTIM, DWGTS1, DSTIM1, STOFLG)
          END IF
        END IF
C
C      Check Answer;  Two potential cases for False Random Recognition;
C      Is : Image present but Answer wrong;
C      Is : Image not present but Answer not zero ( if null response
C      selected)
C
C
        LEVEL = DEPTH + 1
        IF (RESULT .EQ. 1) THEN
C
C       Compute number of occurrences of correct answer for image;
C
```

```
          CALL COMP(ANSWER, LEVEL, ICHECK, IMCNT)
          IF (ICHECK .EQ. 1) THEN
            CORRES = CORRES + 1
          ELSE
            RANRES = RANRES + 1
          END IF
        ELSE
          CALL COMP(ZERO, LEVEL, CHECK, 1)
          IF ((CHECK .EQ. 0) .AND. (NULFLG .EQ. 0)) RANRES = RANRES + 1
C
C       Compute number of occurrences of zero answer
C
          CALL COMP(ZERO, LEVEL, ICHECK, 1)
          IF (ICHECK .EQ. 1) ZERRES = ZERRES + 1
        END IF
C
C       Continue major loop;
C
  53 CONTINUE
C
      WRITE (6,*)'Incorrect Output %  ', DFLOAT(RANRES) * 100.0D0 /
     &      DFLOAT(EXERC)
      WRITE (6,*)'Number of Images presented % ', DFLOAT(IMARES) * 100.
     &      0D0 / DFLOAT(EXERC)
      WRITE (6,*)'Number of correct answers %  ', DFLOAT(CORRES) * 100.
     &      0D0 / DFLOAT(EXERC)
      WRITE (6,*)'Number of zero answers %  ', DFLOAT(ZERRES) * 100.0D0
     &      / DFLOAT(EXERC)
C
C     If CSHFLG=1, plot CASH;
C
      IF (CSHFLG .EQ. 1) CALL CAPT(CSHPLT)
C
C     If NOIFLG=1, exercise with noisy image;
C
      IF (NOIFLG .EQ. 1) THEN
C
C     Exercise array;- now with noisy Image;
C
       WRITE (6,*)'Exercising array with noisy image...'
C
C     Do Noise Standard Deviation loop;
C
       DO 62 NOICNT = 0, 10000
         IF (MOD(NOICNT,1000) .EQ. 0) WRITE (6,74) NOICNT
         GAURES = 0
         STADEV = DFLOAT(NOICNT) / 10000.0D0
C
C     Begin loop;
C
         DO 61 GAUCNT = 1, EXERC
C
           DO 55 X = 0, LEN1 - 1
C
             DO 54 Y = 0, WID1 - 1
C
C     Generate noisy image - clip if CLGFLG=1;
C     Use selected noise image;
C
               PN = G05CAF(PN)
```

```
                IF (CLGFLG .EQ. 1) THEN
                  RANDNO = DNINT(G05DDF(IMAGE(NOINUM,X,Y),STADEV))
                ELSE
                  RANDNO = G05DDF(IMAGE(NOINUM,X,Y),STADEV)
                END IF
                IF (RANDNO .GT. 1.0D0) RANDNO = 1.0D0
                IF (RANDNO .LT. 0.0D0) RANDNO = 0.0D0
                FRAME(X,Y,1) = RANDNO
  54          CONTINUE
C
  55        CONTINUE
C
C     Option - Do Hopfield Style model;
C
            IF (HOPFLG .EQ. 1) THEN
              LEVEL = DEPTH + 1
C
              DO 60 HOPCNT = 1, EXERC
C
C     Call array;
C
                IF (BOLFLG .EQ. 0) THEN
                  CALL ARRAY(0, GLOFLG, STIFLG)
                END IF
                IF (BOLFLG .EQ. 1) THEN
                  CALL HOPMOD(0, GLOFLG, STIFLG, LEN1, WID1, PROPOR)
                END IF
                IF (BOLFLG .EQ. 2) THEN
                  CALL PROP(0, GLOFLG, STIFLG, CORECT, NUMIM, TRACNT,
     &                 DWGTS, DSTIM, DWGTS1, DSTIM1, STOFLG)
                END IF
C
C     Feed back everything;
C
                DO 57 X = 0, LENGTH - 1
C
                  DO 56 Y = 0, WIDTH - 1
                    FRAME(X,Y,1) = FRAME(X,Y,LEVEL)
  56                CONTINUE
C
  57              CONTINUE
C
C
C      Now overwrite with the clamped part  - if PROPOR is greater than
C      0.0D0
C
                IF (PROPOR .GT. 0.00D0) THEN
C
                  DO 59 X = 0, NINT(DFLOAT(LEN1 - 1)*PROPOR)
C
                    DO 58 Y = 0, NINT(DFLOAT(WID1 - 1)*PROPOR)
                      FRAME(X,Y,1) = IMAGE(IMCNT,X,Y)
  58                CONTINUE
C
  59                CONTINUE
C
                END IF
  60          CONTINUE
C
            ELSE
C
C     Call array;
C
```

```
              IF (BOLFLG .EQ. 0) THEN
                CALL ARRAY(0, GLOFLG, STIFLG)
              END IF
              IF (BOLFLG .EQ. 1) THEN
                CALL HOPMOD(0, GLOFLG, STIFLG, LEN1, WID1, PROPOR)
              END IF
              IF (BOLFLG .EQ. 2) THEN
                CALL PROP(0, GLOFLG, STIFLG, CORECT, NUMIM, TRACNT,
     &               DWGTS, DSTIM, DWGTS1, DSTIM1, STOFLG)
              END IF
            END IF
C
C       Compute number of occurrences of correct answer
C
            LEVEL = DEPTH + 1
            CALL COMP(ANSWER, LEVEL, ICHECK, NOINUM)
            IF (ICHECK .EQ. 1) GAURES = GAURES + 1
C
C       Continue major loop;
C
  61     CONTINUE
C
C     Write out to array;
C
         PLNOI(NOICNT) = DFLOAT(GAURES) * 100.0D0 / DFLOAT(EXERC)
  62    CONTINUE
C
C     Plot;
C
       CALL NPLOT(PLNOI, BOLFLG)
      END IF
C
C     Option - Plot weights for selected element;
C
      IF (WGTFLG .EQ. 1) THEN
        WRITE (6,*)'Plotting element weights...'
        CALL ELPLOT(VALUES, STIVAL, LENGTH, WIDTH, MAXCNT, XELEM, YELEM,
     &       ZELEM, BOLFLG)
      END IF
C
C     Option - exercise array with a corrupted image and connect its
C     output to its input - plot after EXERC iterations;
C
      IF (CONFLG .EQ. 1) THEN
C
        DO 72 IMCNT = 1, NUMIM
          WRITE (6,*)'Copying corrupted picture', IMCNT, 'into Frame...'
C
C     Copy BADPIC into FRAME;
C
          DO 64 X = 0, LEN1 - 1
C
            DO 63 Y = 0, WID1 - 1
              FRAME(X,Y,1) = BADPIC(IMCNT,X,Y)
  63        CONTINUE
C
  64      CONTINUE
C
C     Zero rest of FRAME;
C
```

```
          DO 66 X = LEN1, LENGTH - 1
C
            DO 65 Y = WID1, WIDTH - 1
              FRAME(X,Y,1) = 0.0D0
 65         CONTINUE
C
 66       CONTINUE
C
          WRITE (6,*)'Calling array...'
C
C       Now call array;
C
          DO 69 FDBCNT = 1, EXERC
            IF (MOD(FDBCNT,MULT1) .EQ. 0) WRITE (6,74) FDBCNT
            IF (BOLFLG .EQ. 0) THEN
              CALL ARRAY(0, GLOFLG, STIFLG)
            END IF
            IF (BOLFLG .EQ. 1) THEN
              CALL HOPMOD(0, GLOFLG, STIFLG, LEN1, WID1, PROPOR)
            END IF
            IF (BOLFLG .EQ. 2) THEN
              CALL PROP(0, GLOFLG, STIFLG, CORECT, NUMIM, TRACNT, DWGTS,
     &             DSTIM, DWGTS1, DSTIM1, STOFLG)
            END IF
C
C       Transfer output to input;
C
            DO 68 X = 0, LENGTH - 1
C
              DO 67 Y = 0, WIDTH - 1
                FRAME(X,Y,1) = FRAME(X,Y,DEPTH + 1)
 67           CONTINUE
C
 68         CONTINUE
C
 69       CONTINUE
C
C       Option - plot result;
C
          IF (FINFLG .EQ. 1) THEN
            WRITE (6,*)'Plotting corrupted picture', IMCNT, '...'
            CALL PICT(BADPIC, LEN1, WID1, TITLE3, IMCNT)
C
C       Copy FRAME into BADPIC;
C
            DO 71 X = 0, LEN1 - 1
C
              DO 70 Y = 0, WID1 - 1
                BADPIC(IMCNT,X,Y) = FRAME(X,Y,1)
 70           CONTINUE
C
 71         CONTINUE
C
            WRITE (6,*)'Plotting resultant picture...'
            CALL PICT(BADPIC, LEN1, WID1, TITLE4, IMCNT)
          END IF
 72     CONTINUE
C
      END IF
C
```

```
      CALL GREND
      STOP
   73 FORMAT (A)
   74 FORMAT (' ', I8)
      END
```

FORTRAN77 subroutine ARP.QUEST

– configuration input–question subroutine

```
C     ***********************************************
C     * Associative Reward-Punish Element (ARP)     *
C     * Image Recognition Input question subroutine; *
C     *                                             *
C     *                                             *
C     * - Multiple image version;                   *
C     * - The routine reads in the input parameters  *
C     *   and image from a file called NAME1;       *
C     * - The routine reads in general parameters    *
C     *   from a file called PARAM;                 *
C     * - The program also reads in the desired     *
C     *   output answer image from the file, NAME1; *
C     * - Also reads a corrupted image example;     *
C     * - ARP.PREP should be used to prepare a      *
C     *   suitable input file;                      *
C     * - Listed defaults are:                      *
C     *   1. Global Indirect ARP                    *
C     *   2. Global Direct Gradient ARP             *
C     *   3. Local ARP (single level only)          *
C     *   4. Deterministic Backpropagation          *
C     *   5. Hopfield Network (single level only)   *
C     *   6. Stochastic Backpropagation             *
C     * - Program uses DOUBLE PRECISION variables;  *
C     * Author: Richard Leaver                      *
C     * Created:16th June 1987                      *
C     * Update : 5th January 1988                   *
C     * Frozen :20th July 1988                      *
C     ***********************************************
      SUBROUTINE QUEST(IMAGE, ANSWER, BADPIC, MAXCNT, REW1, PUN1,
     &           PROPOR, CONFD, ITER, BAT, ISHIFT)
C
C     Define everything as implicit REAL*8
C
      IMPLICIT REAL*8(A - H,Q - Z)
C
C     Image to be recognised;
C
      REAL*8 IMAGE(10,0:4,0:4)
C
C     Required output;
C
      REAL*8 ANSWER(10,0:4,0:4)
C
C     Corrupted picture;
C
      REAL*8 BADPIC(10,0:4,0:4)
C
C     Define Frame - Max X x Y x Z ;
C
C     FRAME(0:X-1,0:Y-1,Z+1)
C
C     Final Frame N+1 is the output;
C
      REAL*8 FRAME(0:4,0:4,5)
C
C     Define Weights for X x Y ARP elements with Z  Frames;
C
      REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
C
C     Define Stimulus for X x Y ARP elements with Z  Frames;
C
      REAL*8 STIM(0:4,0:4,4)
C
C     Save SUMs for X x Y ARP elements, Z Frames ;
C
```

```
      REAL*8 SUMS(0:4,0:4,4)
C
C      Real *8 variables;
C
      REAL*8 REW1, PUN1, LAMBDA, RHO, TEMP, PN, PROPOR, CONFD
C
C      Integers;
C
      INTEGER MAXCNT, XELEM, YELEM, ZELEM, NUMIM, NOINUM
      INTEGER LENGTH, WIDTH, DEPTH, DUMDEP
      INTEGER LEN1, WID1
      INTEGER RESFLG, CLIFLG, CLEFLG, GLOFLG, NEUFLG, DIRFLG, WGTFLG
      INTEGER TRAFLG, CLGFLG, NOIFLG, DUMFLG, TRYFLG, NULFLG
      INTEGER FINFLG, CONFLG, MULFLG, CSHFLG, ENEFLG, SYMFLG, HOPFLG
      INTEGER GRAFLG, STIFLG, INPFLG, PICFLG, BOLFLG, LINFLG, BRKFLG,
     &        STOFLG
      INTEGER EXERC, GOMAX, ITER, BAT
C
C      Integer array to hold Rewards for each element - final frame only;
C
      INTEGER REWARD(0:4,0:4)
C
C      Alternative for Global Rewards;
C
      INTEGER GLOREW
C
C      Characters;
C
      CHARACTER*10 ANS, NAME1, DUMMY
      CHARACTER*70 TITLE
      CHARACTER*30 TITLE1
      CHARACTER*30 TITLE2
      CHARACTER*30 TITLE3
      CHARACTER*30 TITLE4
C
C      TITLE  = Title of job;
C      TITLE1 = Title of ARP input image plot;
C      TITLE2 = Title of ARP output answer image plot;
C      TITLE3 = Title of ARP corrupted image plot;
C      TITLE4 = Title of ARP uncorrupted image?;
C      NAME1 = Input image;
C
C      Define Common Block One;
C
      COMMON /ONE/ FRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
     &       REWARD, GLOREW
C
C      Define Common Block Two;
C
      COMMON /TWO/ RHO, TEMP, LAMBDA, PN
C
C      Define Common Block Four;
C
      COMMON /FOUR/ XELEM, YELEM, ZELEM, RESFLG, CLIFLG, CLEFLG, GLOFLG,
     &       NEUFLG, DIRFLG, WGTFLG, TRAFLG, CLGFLG, NOIFLG, DUMFLG,
     &       TRYFLG, NULFLG, FINFLG, CONFLG, MULFLG, CSHFLG, ENEFLG,
     &       SYMFLG, HOPFLG, GRAFLG, STIFLG, INPFLG, PICFLG, BOLFLG,
     &       LINFLG, BRKFLG, STOFLG, NUMIM, NOINUM, EXERC, GOMAX, LEN1,
     &       WID1, TITLE, TITLE1, TITLE2, TITLE3, TITLE4, NAME1, DUMMY
C
C      Initialise Flags;
C
```

```
      BRKFLG = 0
      RESFLG = 0
      LINFLG = 0
      PICFLG = 0
      CLIFLG = 0
      CLEFLG = 0
      CLGFLG = 0
      DIRFLG = 0
      GLOFLG = 0
      NEUFLG = 0
      WGTFLG = 0
      NOIFLG = 0
      TRYFLG = 0
      DUMFLG = 0
      TRAFLG = 0
      NULFLG = 0
      FINFLG = 0
      CONFLG = 0
      MULFLG = 0
      CSHFLG = 0
      ENEFLG = 0
      SYMFLG = 0
      HOPFLG = 0
      GRAFLG = 0
      STIFLG = 0
      INPFLG = 0
      BOLFLG = 0
      STOFLG = 0
C
C     Define input and output titles;
C
      TITLE1 = 'INPUT IMAGE'
      TITLE2 = 'REQUIRED ANSWER'
      TITLE3 = 'CORRUPTED IMAGE'
      TITLE4 = 'CORRECTED IMAGE'
C
C     Define default noise image number;
C
      NOINUM = 1
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     Read instructions and input image from file;
C
C     Format is;
C     TITLE
C     LEN1,              ..... Image length
C     WID1,              ..... Image width
C     DUMDEP,            .....allows for the old style data files;
C     NUMIM,
C     DUMMY
C     IMAGE DATA         .....repeated for multiple images;
C     ANSWER DATA        .....
C     BAD DATA           .....
C
      WRITE (6,*)'Input file name...'
      READ (5,58) NAME1
C
C     Open NAME1
C
```

```
      OPEN (1,FILE=NAME1)
      READ (1,58) TITLE
      READ (1,*) LEN1
      READ (1,*) WID1
      READ (1,*) DUMDEP
      READ (1,*) NUMIM
      IF (NUMIM .GT. 1) MULFLG = 1
C
C     Read DUMMY;
C
      READ (1,58) DUMMY
C
C     Read Image data;-
C
      WRITE (6,*)'Defining Image...'
C
      DO 1 IMCNT = 1, NUMIM
C
        READ (1,*) ((IMAGE(IMCNT,I,J),J=0,WID1 - 1),I=0,LEN1 - 1)
C
    1 CONTINUE
C
C     Read Answer data;
C
      WRITE (6,*)'Defining Answer...'
C
      DO 2 IMCNT = 1, NUMIM
C
        READ (1,*) ((ANSWER(IMCNT,I,J),J=0,WID1 - 1),I=0,LEN1 - 1)
C
    2 CONTINUE
C
C     Read Corrupted image data;
C
      WRITE (6,*)'Defining Corrupted Image...'
C
C
      DO 3 IMCNT = 1, NUMIM
        READ (1,*) ((BADPIC(IMCNT,I,J),J=0,WID1 - 1),I=0,LEN1 - 1)
    3 CONTINUE
C
C
      WRITE (6,*)'Closing file..f1..'
C
C     Close 1;
C
      REWIND 1
      CLOSE (1)
C
C     Open Parameter file;
C
      OPEN (3,FILE='PARAM')
      READ (3,*) LAMBDA
      READ (3,*) LENGTH
      READ (3,*) WIDTH
      READ (3,*) DEPTH
      READ (3,*) RHO
      READ (3,*) TEMP
      READ (3,*) REW1
      READ (3,*) PUN1
```

```
      READ (3,*) MAXCNT
      READ (3,*) EXERC
      READ (3,*) GOMAX
      READ (3,*) PROPOR
      READ (3,*) CONFD
      READ (3,*) ITER
      READ (3,*) BAT
      READ (3,*) ISHIFT
C
C      Close (3)
C
      WRITE (6,*)'Closing file..f3..'
      CLOSE (3)
C
C      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C
      WRITE (6,*)'Do you want to run on Defaults?...'
    4 READ (5,58) ANS
      IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
        GO TO 6
      END IF
      IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
        WRITE (6,*)'Which set of defaults?...'
    5   READ (5,*) IDEF
        IF ((IDEF .LT. 1) .OR. (IDEF .GT. 6)) GO TO 5
        GO TO 56
      END IF
      GO TO 4
    6 WRITE (6,*)'Do you want to plot the Neural Net connections?...'
    7 READ (5,58) ANS
      IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
        NEUFLG = 0
        GO TO 11
      END IF
      IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
        NEUFLG = 1
    8   WRITE (6,*)'Do you want a Block plot or Line Plot?...'
    9   READ (5,58) ANS
        IF ((ANS .EQ. 'B') .OR. (ANS .EQ. 'b')) THEN
          LINFLG = 0
          GO TO 10
        END IF
        IF ((ANS .EQ. 'L') .OR. (ANS .EQ. 'l')) THEN
          LINFLG = 1
          GO TO 10
        END IF
        GO TO 9
   10   GO TO 11
      END IF
      GO TO 7
C
C      Do you want to plot the Weights of the ARP elements?
C
   11 WRITE (6,*)'Do you want to plot any ARP weights?...'
   12 READ (5,58) ANS
      IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
        WGTFLG = 0
        GO TO 13
      END IF
```

```
      IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
        WGTFLG = 1
C
C     Which element to be plotted;
C
        WRITE (6,*)'Which element would you like to plot?...'
        WRITE (6,*)'... please enter X,Y,Z coordinates?...'
        READ (5,*) XELEM, YELEM, ZELEM
        GO TO 13
      END IF
      GO TO 12
   13 WRITE (6,*)'Do you want Global or Local REWARD for the run?...'
   14 READ (5,58) ANS
      IF ((ANS .EQ. 'G') .OR. (ANS .EQ. 'g')) THEN
        GLOFLG = 1
        WRITE (6,*)'Do you want Direct or Indirect REWARD for the run?..
     &.'
   15   READ (5,58) ANS
        IF ((ANS .EQ. 'D') .OR. (ANS .EQ. 'd')) THEN
          DIRFLG = 1
          GO TO 16
        END IF
        IF ((ANS .EQ. 'I') .OR. (ANS .EQ. 'i')) THEN
          DIRFLG = 0
          GO TO 16
        END IF
        GO TO 15
      END IF
      IF ((ANS .EQ. 'L') .OR. (ANS .EQ. 'l')) THEN
        GLOFLG = 0
        GO TO 16
      END IF
      GO TO 14
   16 WRITE (6,*)'Do you want Clipped or Unclipped training noise?...'
   17 READ (5,58) ANS
      IF ((ANS .EQ. 'C') .OR. (ANS .EQ. 'c')) THEN
        CLIFLG = 1
        GO TO 18
      END IF
      IF ((ANS .EQ. 'U') .OR. (ANS .EQ. 'u')) THEN
        CLIFLG = 0
        GO TO 18
      END IF
      GO TO 17
   18 WRITE (6,*)'Do you want Clipped or Unclipped exercising noise?...'
   19 READ (5,58) ANS
      IF ((ANS .EQ. 'C') .OR. (ANS .EQ. 'c')) THEN
        CLEFLG = 1
        GO TO 20
      END IF
      IF ((ANS .EQ. 'U') .OR. (ANS .EQ. 'u')) THEN
        CLEFLG = 0
        GO TO 20
      END IF
      GO TO 19
   20 WRITE (6,*)'Do you want Clipped or Unclipped Gaussian noise?...'
   21 READ (5,58) ANS
      IF ((ANS .EQ. 'C') .OR. (ANS .EQ. 'c')) THEN
        CLGFLG = 1
        GO TO 22
```

```
      END IF
      IF ((ANS .EQ. 'U') .OR. (ANS .EQ. 'u')) THEN
        CLGFLG = 0
        GO TO 22
      END IF
      GO TO 21
C
C     Do you want to examine the noisy input image response?...
C
   22 WRITE (6,*)'Do you want to plot the noisy image response?...'
   23 READ (5,58) ANS
      IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
        NOIFLG = 0
        GO TO 24
      END IF
      IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
        NOIFLG = 1
C
C     Input noise image number;
C
        WRITE (6,*)'Please input noise image number...'
        READ (5,*) NOINUM
        GO TO 24
      END IF
      GO TO 23
   24 CONTINUE
C
C     Check to see if weights recorded in weight file before;
C
      WRITE (6,*)'Does file  ', DUMMY, 'contain previous weights for thi
     &s problem?...'
   25 READ (5,58) ANS
      IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
        DUMFLG = 1
        WRITE (6,*)'Do you just want to exercise the program?...'
   26   READ (5,58) ANS
        IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
          TRYFLG = 1
          GO TO 27
        END IF
        IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
          TRYFLG = 0
          GO TO 27
        END IF
        GO TO 26
      END IF
      IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
        DUMFLG = 0
        GO TO 27
      END IF
      GO TO 25
   27 CONTINUE
C
C
      IF (NUMIM .LT. 2) THEN
        WRITE (6,*)'Do you want to use random inputs?...'
   28   READ (5,58) ANS
        IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
          MULFLG = 0
          WRITE (6,*)'Do you want to use a null response method?...'
```

```
 29     READ (5,58) ANS
        IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
          NULFLG = 1
          GO TO 30
        END IF
        IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
          NULFLG = 0
          GO TO 30
        END IF
        GO TO 29
      END IF
      IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
        MULFLG = 1
        GO TO 30
      END IF
      GO TO 28
C
 30   CONTINUE
    ELSE
      MULFLG = 1
    END IF
C
    WRITE (6,*)'Do you want to plot the final answer?...'
 31 READ (5,58) ANS
    IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
      FINFLG = 1
      GO TO 32
    END IF
    IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
      FINFLG = 0
      GO TO 32
    END IF
    GO TO 31
C
 32 CONTINUE
    WRITE (6,*)'Do you want to plot the Cash values?...'
 33 READ (5,58) ANS
    IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
      CSHFLG = 1
      GO TO 34
    END IF
    IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
      CSHFLG = 0
      GO TO 34
    END IF
    GO TO 33
C
 34 CONTINUE
    WRITE (6,*)'Do you want to use the final feedback setting?...'
 35 READ (5,58) ANS
    IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
      CONFLG = 1
      GO TO 36
    END IF
    IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
      CONFLG = 0
      GO TO 36
    END IF
    GO TO 35
C
```

```
 36 CONTINUE
    WRITE (6,*)'Do you want to use the Error function?...'
 37 READ (5,58) ANS
    IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
      ENEFLG = 1
      GO TO 38
    END IF
    IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
      ENEFLG = 0
      GO TO 38
    END IF
    GO TO 37
C
 38 CONTINUE
    WRITE (6,*)'Do you want to impose symmetry on the weights?...'
 39 READ (5,58) ANS
    IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
      SYMFLG = 1
      GO TO 40
    END IF
    IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
      SYMFLG = 0
      GO TO 40
    END IF
    GO TO 39
C
 40 CONTINUE
    WRITE (6,*)'Do you want to do a Hopfield style test?...'
 41 READ (5,58) ANS
    IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
      HOPFLG = 1
      GO TO 42
    END IF
    IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
      HOPFLG = 0
      GO TO 42
    END IF
    GO TO 41
C
 42 CONTINUE
    WRITE (6,*)'Do you want use a Gradient method?...'
 43 READ (5,58) ANS
    IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
      GRAFLG = 1
      GO TO 44
    END IF
    IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
      GRAFLG = 0
      GO TO 44
    END IF
    GO TO 43
C
 44 CONTINUE
    WRITE (6,*)'Do you want to use a stimulus?...'
 45 READ (5,58) ANS
    IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
      STIFLG = 1
      GO TO 46
    END IF
    IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
```

```
          STIFLG = 0
          GO TO 46
        END IF
        GO TO 45
C
   46 CONTINUE
        WRITE (6,*)'Do you want to zero Wii ?...'
   47 READ (5,58) ANS
        IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
          INPFLG = 1
          GO TO 48
        END IF
        IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
          INPFLG = 0
          GO TO 48
        END IF
        GO TO 47
C
   48 CONTINUE
        WRITE (6,*)'Do you want to plot the input/output training pictures
       &?...'
   49 READ (5,58) ANS
        IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
          PICFLG = 1
          GO TO 50
        END IF
        IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
          PICFLG = 0
          GO TO 50
        END IF
        GO TO 49
   50 CONTINUE
        WRITE (6,*)'Do you want to use a Hopfield, Arp or Back model?...'
   51 READ (5,58) ANS
        IF ((ANS .EQ. 'B') .OR. (ANS .EQ. 'b')) THEN
          BOLFLG = 2
          GO TO 52
        END IF
        IF ((ANS .EQ. 'H') .OR. (ANS .EQ. 'h')) THEN
          BOLFLG = 1
          GO TO 52
        END IF
        IF ((ANS .EQ. 'A') .OR. (ANS .EQ. 'a')) THEN
          BOLFLG = 0
          GO TO 52
        END IF
        GO TO 51
   52 CONTINUE
        WRITE (6,*)'Do you want to randomise the weight matrix?...'
   53 READ (5,58) ANS
        IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
          BRKFLG = 1
          GO TO 54
        END IF
        IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
          BRKFLG = 0
          GO TO 54
        END IF
        GO TO 53
   54 CONTINUE
```

```
      WRITE (6,*)'Do you want to use a stochastic method?...'
   55 READ (5,58) ANS
      IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
        STOFLG = 1
        GO TO 57
      END IF
      IF ((ANS .EQ. 'N') .OR. (ANS .EQ. 'n')) THEN
        STOFLG = 0
        GO TO 57
      END IF
      GO TO 55
C
C     Set all FLAGS to defaults; (1)
C
   56 IF (IDEF .EQ. 1) THEN
        BRKFLG = 1
        RESFLG = 0
        LINFLG = 0
        PICFLG = 0
        CLIFLG = 1
        CLEFLG = 1
        CLGFLG = 1
        DIRFLG = 0
        GLOFLG = 1
        NEUFLG = 0
        WGTFLG = 0
        NOIFLG = 0
        TRYFLG = 0
        DUMFLG = 0
        TRAFLG = 0
        NULFLG = 0
        FINFLG = 0
        CONFLG = 0
        MULFLG = 1
        CSHFLG = 0
        ENEFLG = 1
        SYMFLG = 0
        HOPFLG = 0
        GRAFLG = 0
        STIFLG = 1
        INPFLG = 0
        BOLFLG = 0
        STOFLG = 0
      END IF
C
C     Set all FLAGS to defaults; (1)
C
      IF (IDEF .EQ. 2) THEN
        BRKFLG = 1
        LINFLG = 0
        RESFLG = 0
        PICFLG = 0
        CLIFLG = 1
        CLEFLG = 1
        CLGFLG = 1
        DIRFLG = 1
        GLOFLG = 1
        NEUFLG = 0
        WGTFLG = 0
        NOIFLG = 0
```

```
    TRYFLG = 0
    DUMFLG = 0
    TRAFLG = 0
    NULFLG = 0
    FINFLG = 0
    CONFLG = 0
    MULFLG = 1
    CSHFLG = 0
    ENEFLG = 1
    SYMFLG = 0
    HOPFLG = 0
    GRAFLG = 1
    STIFLG = 1
    INPFLG = 0
    BOLFLG = 0
    STOFLG = 0
  END IF
  IF (IDEF .EQ. 3) THEN
    BRKFLG = 1
    RESFLG = 0
    LINFLG = 0
    PICFLG = 0
    CLIFLG = 1
    CLEFLG = 1
    CLGFLG = 1
    DIRFLG = 0
    GLOFLG = 0
    NEUFLG = 0
    WGTFLG = 0
    NOIFLG = 0
    TRYFLG = 0
    DUMFLG = 0
    TRAFLG = 0
    NULFLG = 0
    FINFLG = 0
    CONFLG = 0
    MULFLG = 1
    CSHFLG = 0
    ENEFLG = 1
    SYMFLG = 0
    HOPFLG = 0
    GRAFLG = 0
    STIFLG = 1
    INPFLG = 0
    BOLFLG = 0
    STOFLG = 0
  END IF
  IF (IDEF .EQ. 4) THEN
    BRKFLG = 1
    RESFLG = 0
    LINFLG = 0
    PICFLG = 0
    CLIFLG = 1
    CLEFLG = 1
    CLGFLG = 1
    DIRFLG = 0
    GLOFLG = 0
    NEUFLG = 0
    WGTFLG = 0
    NOIFLG = 0
```


```
    TRYFLG = 0
    DUMFLG = 0
    TRAFLG = 0
    NULFLG = 0
    FINFLG = 0
    CONFLG = 0
    MULFLG = 1
    CSHFLG = 0
    ENEFLG = 1
    SYMFLG = 0
    HOPFLG = 0
    GRAFLG = 0
    STIFLG = 1
    INPFLG = 0
    BOLFLG = 2
    STOFLG = 0
  END IF
  IF (IDEF .EQ. 5) THEN
    BRKFLG = 1
    RESFLG = 0
    LINFLG = 0
    PICFLG = 0
    CLIFLG = 1
    CLEFLG = 1
    CLGFLG = 1
    DIRFLG = 0
    GLOFLG = 0
    NEUFLG = 0
    WGTFLG = 0
    NOIFLG = 0
    TRYFLG = 0
    DUMFLG = 0
    TRAFLG = 0
    NULFLG = 0
    FINFLG = 0
    CONFLG = 0
    MULFLG = 1
    CSHFLG = 0
    ENEFLG = 1
    SYMFLG = 1
    HOPFLG = 0
    GRAFLG = 0
    STIFLG = 1
    INPFLG = 1
    BOLFLG = 1
    STOFLG = 0
  END IF
  IF (IDEF .EQ. 6) THEN
    BRKFLG = 1
    RESFLG = 0
    LINFLG = 0
    PICFLG = 0
    CLIFLG = 1
    CLEFLG = 1
    CLGFLG = 1
    DIRFLG = 0
    GLOFLG = 0
    NEUFLG = 0
    WGTFLG = 0
    NOIFLG = 0
```

```
      TRYFLG = 0
      DUMFLG = 0
      TRAFLG = 0
      NULFLG = 0
      FINFLG = 0
      CONFLG = 0
      MULFLG = 1
      CSHFLG = 0
      ENEFLG = 1
      SYMFLG = 0
      HOPFLG = 0
      GRAFLG = 0
      STIFLG = 1
      INPFLG = 0
      BOLFLG = 2
      STOFLG = 1
    END IF
57  RETURN
58  FORMAT (A)
    END
```

FORTRAN77 subroutine ARP.CONFIG

– configuration output subroutine

```
C     ************************************************
C     * Associative Reward-Punish Element (ARP)      *
C     * Image Recognition Configuration subroutine;  *
C     *                                              *
C     *                                              *
C     * - Multiple image version;                    *
C     * - The routine writes out the simulation      *
C     *   program parameters;                        *
C     * - Program uses DOUBLE PRECISION variables;   *
C     * Author: Richard Leaver                       *
C     * Created:19th July 1987                       *
C     * Update :17th November 1987                   *
C     * Frozen :20th July 1988                       *
C     ************************************************
      SUBROUTINE CONFIG(IMAGE, ANSWER, BADPIC, MAXCNT, REW1, PUN1,
     &           PROPOR, CONFD, ITER, BAT, ISHIFT)
C
C     Define everything as implicit REAL*8
C
      IMPLICIT REAL*8(A - H,O - Z)
C
C     Image to be recognised;
C
      REAL*8 IMAGE(10,0:4,0:4)
C
C     Required output;
C
      REAL*8 ANSWER(10,0:4,0:4)
C
C     Corrupted picture;
C
      REAL*8 BADPIC(10,0:4,0:4)
C
C     Define Frame - Max X x Y x Z ;
C
C     FRAME(0:X-1,0:Y-1,Z+1)
C
C     Final Frame N+1 is the output;
C
      REAL*8 FRAME(0:4,0:4,5)
C
C     Define Weights for X x Y ARP elements with Z  Frames;
C
      REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
C
C     Define Stimulus for X x Y ARP elements with Z  Frames;
C
      REAL*8 STIM(0:4,0:4,4)
C
C     Save SUMs for X x Y ARP elements, Z Frames ;
C
      REAL*8 SUMS(0:4,0:4,4)
C
C     Real *8 variables;
C
      REAL*8 REW1, PUN1, LAMBDA, RHO, TEMP, PN, PROPOR, CONFD
C
C     Integers;
C
      INTEGER MAXCNT, XELEM, YELEM, ZELEM, NUMIM, NOINUM
      INTEGER LENGTH, WIDTH, DEPTH
      INTEGER LEN1, WID1
```

```
      INTEGER RESFLG, CLIFLG, CLEFLG, GLOFLG, NEUFLG, DIRFLG, WGTFLG
      INTEGER TRAFLG, CLGFLG, NOIFLG, DUMFLG, TRYFLG, NULFLG
      INTEGER FINFLG, CONFLG, MULFLG, CSHFLG, ENEFLG, SYMFLG, HOPFLG
      INTEGER GRAFLG, STIFLG, INPFLG, PICFLG, BOLFLG, LINFLG, BRKFLG,
     &        STOFLG
      INTEGER EXERC, GOMAX, ITER, BAT
C
C     Integer array to hold Rewards for each element - final frame only;
C
      INTEGER REWARD(0:4,0:4)
C
C     Alternative for Global Rewards;
C
      INTEGER GLOREW
C
C     Characters;
C
      CHARACTER*10 ANS, NAME1, DUMMY
      CHARACTER*70 TITLE
      CHARACTER*30 TITLE1
      CHARACTER*30 TITLE2
      CHARACTER*30 TITLE3
      CHARACTER*30 TITLE4
C
C     TITLE  = Title of job;
C     TITLE1 = Title of ARP input image plot;
C     TITLE2 = Title of ARP output answer image plot;
C     TITLE3 = Title of ARP corrupted image plot;
C     TITLE4 = Title of ARP uncorrupted image?;
C     NAME1 = Input image;
C
C     Define Common Block One;
C
      COMMON /ONE/ FRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
     &       REWARD, GLOREW
C
C     Define Common Block Two;
C
      COMMON /TWO/ RHO, TEMP, LAMBDA, PN
C
C     Define Common Block Four;
C
      COMMON /FOUR/ XELEM, YELEM, ZELEM, RESFLG, CLIFLG, CLEFLG, GLOFLG,
     &       NEUFLG, DIRFLG, WGTFLG, TRAFLG, CLGFLG, NOIFLG, DUMFLG,
     &       TRYFLG, NULFLG, FINFLG, CONFLG, MULFLG, CSHFLG, ENEFLG,
     &       SYMFLG, HOPFLG, GRAFLG, STIFLG, INPFLG, PICFLG, BOLFLG,
     &       LINFLG, BRKFLG, STOFLG, NUMIM, NOINUM, EXERC, GOMAX, LEN1,
     &       WID1, TITLE, TITLE1, TITLE2, TITLE3, TITLE4, NAME1, DUMMY
      WRITE (6,*)'Input title = ', NAME1
      WRITE (6,*)'File title = ', TITLE
      WRITE (6,*)'Array Length = ', LENGTH
      WRITE (6,*)'Array Width = ', WIDTH
      WRITE (6,*)'Array Depth = ', DEPTH
      WRITE (6,*)'Image Length = ', LEN1
      WRITE (6,*)'Image Width = ', WID1
      WRITE (6,*)'Number of images = ', NUMIM
      WRITE (6,*)'Dummy variable  = ', DUMMY
      WRITE (6,*)'Rho  = ', RHO
      WRITE (6,*)'Temp = ', TEMP
      WRITE (6,*)'Lambda = ', LAMBDA
```

```
      WRITE (6,*)'Reward probability = ', REW1
      WRITE (6,*)'Punish probability = ', PUN1
      WRITE (6,*)'Number of teaching trials = ', MAXCNT
      WRITE (6,*)'Number of post learning trials = ', EXERC
      WRITE (6,*)'Maximum bound on number of attempts = ', GOMAX
      WRITE (6,*)'Proportion clamped in Hopfield case = ', PROPOR
      WRITE (6,*)'Length proportion clamped = 0,', NINT(DFLOAT(LEN1 - 1)
     &       *PROPOR)
      WRITE (6,*)'Width proportion clamped = 0,', NINT(DFLOAT(WID1 - 1)*
     &       PROPOR)
      WRITE (6,*)'Confidence level (%) = ', CONFD
      WRITE (6,*)'No of program loops  = ', ITER
      WRITE (6,*)'Degree of Batching  = ', BAT
      WRITE (6,*)'Shift Register Length = ', ISHIFT
C
C     Write Image data;-
C
      WRITE (6,*)'Defining Image...'
C
      DO 1 IMCNT = 1, NUMIM
C
        WRITE (6,*) ((IMAGE(IMCNT,I,J),J=0,WIDTH - 1),I=0,LENGTH - 1)
C
    1 CONTINUE
C
      WRITE (6,*)'Defining Answer...'
C
C     Read Answer data;
C
      DO 2 IMCNT = 1, NUMIM
C
        WRITE (6,*) ((ANSWER(IMCNT,I,J),J=0,WIDTH - 1),I=0,LENGTH - 1)
C
    2 CONTINUE
C
C     Read Corrupted image data;
C
      WRITE (6,*)'Defining Corrupted Image...'
C
C
      DO 3 IMCNT = 1, NUMIM
        WRITE (6,*) ((BADPIC(IMCNT,I,J),J=0,WIDTH - 1),I=0,LENGTH - 1)
    3 CONTINUE
C
      IF (NEUFLG .EQ. 1) THEN
        WRITE (6,*)' Plotting neural net connections ENABLED'
      ELSE
        WRITE (6,*)' Plotting neural net connections DISABLED'
      END IF
      IF (LINFLG .EQ. 1) THEN
        WRITE (6,*)' Link Plot ENABLED'
      ELSE
        WRITE (6,*)' Block Plot ENABLED'
      END IF
      IF (PICFLG .EQ. 1) THEN
        WRITE (6,*)' Plotting input/output pictures ENABLED'
      ELSE
        WRITE (6,*)' Plotting input/output pictures DISABLED'
      END IF
      IF (WGTFLG .EQ. 1) THEN
```

```
  WRITE (6,*)' Plotting weights ENABLED'
  WRITE (6,*)' Weights are ', XELEM, YELEM, ZELEM
ELSE
  WRITE (6,*)' Plotting weights DISABLED'
END IF
IF (GLOFLG .EQ. 1) THEN
  WRITE (6,*)' Global Reward scheme ENABLED'
ELSE
  WRITE (6,*)' Local Reward scheme ENABLED'
END IF
IF (DIRFLG .EQ. 1) THEN
  WRITE (6,*)' Direct Reward ENABLED'
ELSE
  WRITE (6,*)' Indirect Reward ENABLED'
END IF
IF (CLIFLG .EQ. 1) THEN
  WRITE (6,*)' Clipped Training Noise ENABLED'
ELSE
  WRITE (6,*)' Clipped Training Noise DISABLED'
END IF
IF (CLEFLG .EQ. 1) THEN
  WRITE (6,*)' Clipped Exercising Noise ENABLED'
ELSE
  WRITE (6,*)' Clipped Exercising Noise DISABLED'
END IF
IF (CLGFLG .EQ. 1) THEN
  WRITE (6,*)' Clipped Gaussian Noise ENABLED'
ELSE
  WRITE (6,*)' Clipped Gaussian Noise DISABLED'
END IF
IF (NOIFLG .EQ. 1) THEN
  WRITE (6,*)' Plotting Noisy Image ENABLED'
  WRITE (6,*)' Image number ', NOINUM
ELSE
  WRITE (6,*)' Plotting Noisy Image DISABLED'
END IF
IF (DUMFLG .EQ. 1) THEN
  WRITE (6,*)' Using weights in file', DUMMY
ELSE
  WRITE (6,*)' Setting all initial weights to zero'
END IF
IF (TRYFLG .EQ. 1) THEN
  WRITE (6,*)' Initial learning DISABLED'
ELSE
  WRITE (6,*)' Normal Training'
END IF
IF (MULFLG .EQ. 0) THEN
  WRITE (6,*)' Use of Random Inputs ENABLED'
ELSE
  WRITE (6,*)' Use of Random Inputs DISABLED'
END IF
IF (NULFLG .EQ. 1) THEN
  WRITE (6,*)' Use of Null Output ENABLED'
ELSE
  WRITE (6,*)' Use of Null Output DISABLED'
END IF
IF (FINFLG .EQ. 1) THEN
  WRITE (6,*)' Plotting Final Answer ENABLED'
ELSE
  WRITE (6,*)' Plotting Final Answer DISABLED'
```

```
END IF
IF (CSHFLG .EQ. 1) THEN
  WRITE (6,*)' Plotting Cash Values ENABLED'
ELSE
  WRITE (6,*)' Plotting Cash Values DISABLED'
END IF
IF (CONFLG .EQ. 1) THEN
  WRITE (6,*)' Final Feedback ENABLED'
ELSE
  WRITE (6,*)' Final Feedback DISABLED'
END IF
IF (ENEFLG .EQ. 1) THEN
  WRITE (6,*)' Use of Backpropagation Error Function ENABLED'
  WRITE (6,*)' Use of Checksum Error Function DISABLED'
ELSE
  WRITE (6,*)' Use of Backpropagation Error Function DISABLED'
  WRITE (6,*)' Use of Checksum Error Function ENABLED'
END IF
IF (SYMFLG .EQ. 1) THEN
  WRITE (6,*)' Symmetric Weights ENABLED'
ELSE
  WRITE (6,*)' Asymmetric Weights ENABLED'
END IF
IF (HOPFLG .EQ. 1) THEN
  WRITE (6,*)' Hopfield Connection ENABLED'
ELSE
  WRITE (6,*)' Normal Connection ENABLED'
END IF
IF (GRAFLG .EQ. 1) THEN
  WRITE (6,*)' Differential Error ENABLED'
ELSE
  WRITE (6,*)' Normal Error ENABLED'
END IF
IF (STIFLG .EQ. 1) THEN
  WRITE (6,*)' Stimulus value = 1.0'
ELSE
  WRITE (6,*)' Stimulus = 0.0'
END IF
IF (INPFLG .EQ. 1) THEN
  WRITE (6,*)' Zeroing Wii'
ELSE
  WRITE (6,*)' Normal Wii'
END IF
IF (BOLFLG .EQ. 2) THEN
  WRITE (6,*)' Back Propagation Model ENABLED'
END IF
IF (BOLFLG .EQ. 1) THEN
  WRITE (6,*)' Hopfield Model ENABLED'
END IF
IF (BOLFLG .EQ. 0) THEN
  WRITE (6,*)' ARP Model ENABLED'
END IF
IF (BRKFLG .EQ. 1) THEN
  WRITE (6,*)' Randomising Weights ENABLED'
ELSE
  WRITE (6,*)' Non Randomised Weights ENABLED'
END IF
IF (STOFLG .EQ. 1) THEN
  WRITE (6,*)' Stochastic model ENABLED'
ELSE
```

```
  WRITE (6,*)' Deterministic model ENABLED'
END IF
RETURN
END
```

FORTRAN77 subroutine ARP:COMP

– compares two images and outputs a global correct/incorrect flag

```
C     ***********************************************
C     * Associative Reward-Punish Element (ARP)     *
C     * Subroutine to compare two images;           *
C     *                                             *
C     * - Primarily for use by ARP.MAIN programs;   *
C     * - Multiple image version;                   *
C     * - Routine uses a Checksum method rather     *
C     *   than Point x Point examination for speed; *
C     * - Adds one to the value in the array to     *
C     *   allow for zero images;                    *
C     * - Common block entry for array              *
C     * - CHECK = 1 for IDENTICAL                   *
C     * - CHECK = 0 for NOT IDENTICAL               *
C     *   FRAME(0:L-1,0:W-1,Z+1) and associated     *
C     *   variables;                                *
C     *                                             *
C     * Author: Richard Leaver                      *
C     * Created: 2nd June 1987                      *
C     * Update :1st October 1987                    *
C     * Frozen :20th July 1988                      *
C     ***********************************************
      SUBROUTINE COMP(IMAGE, LEVEL, CHECK, IMCNT)
C
C     Define everything as implicit REAL*8
C
      IMPLICIT REAL*8(A - H,Q - Z)
C
C     Bring in COMMON block;
C
C
C     Define Common Block One;
C
      COMMON /ONE/ FRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
     &       REWARD, GLOREW
C
C     Define Frame - Max X x Y x Z ;
C
C     FRAME(0:X-1,0:Y-1,Z+1)
C
C     Final Frame N+1 is the output;
C
      REAL*8 FRAME(0:4,0:4,5)
C
C     Define the variables in the COMMON block;
C
      REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
      REAL*8 STIM(0:4,0:4,4)
      REAL*8 SUMS(0:4,0:4,4)
      INTEGER LENGTH, WIDTH, DEPTH, LEVEL, CHECK, GLOREW
C
C     Define Integer Common block;
C
      INTEGER REWARD(0:4,0:4)
C
C     Image to be recognised;
C
      REAL*8 IMAGE(10,0:4,0:4)
C
C     Define 'zero' value ;
C
      ZERO = 1.0D0
C
C     Do Check;
C
```

```
      DO 3 I = 0, LENGTH - 1
C
C      Do Checksum over width of array...multiplying by
C      K=J+1 to stop symmetry problem;
C
        VSUM = 0
C
        DO 2 J = 0, WIDTH - 1
          K = J + 1
C
C      Incorporate correction routine for back propagation;
C      since exponential cannot go to 1 or 0 except with infinitely large
C      weights;
C
           IF (FRAME(I,J,LEVEL) .GT. 0.95D0) THEN
             VALX = 1.0D0
             GO TO 1
           END IF
           IF (FRAME(I,J,LEVEL) .LT. 0.05D0) THEN
             VALX = 0.0D0
             GO TO 1
           END IF
           VALX = FRAME(I,J,LEVEL)
    1      VSUM = VSUM + DABS((VALX + ZERO)*K - (IMAGE(IMCNT,I,J) + ZERO)
     &     *K)
    2   CONTINUE
C
C      If this is zero then array rows are identical;
C
        IF (DABS(VSUM) .GT. 0.000D0) THEN
C
C      Not identical;
C
           CHECK = 0
           RETURN
         END IF
    3 CONTINUE
C
C      Identical;
C
      CHECK = 1
      RETURN
      END
```

FORTRAN77 subroutine ARP.COMP1

– compares two images and outputs local correct/incorrect flags for each coordinate

```
C     ***********************************************
C     * Associative Reward-Punish Element (ARP)     *
C     * Subroutine to ' mark' the ARP's answers     *
C     * during the training period and puts the     *
C     * error value in the CORECT array;            *
C     *                                             *
C     * - NB; array used for back propag errors     *
C     *   so DEPTH level holds visible element      *
C     *   errors; (differential error = y-d)        *
C     * - Multiple image version;                   *
C     * - Common block entry for array              *
C     * - Corect = 0.0 for right answer;            *
C     * - Corect = +-1.0 for wrong answer;          *
C     *                                             *
C     * Author: Richard Leaver                      *
C     * Created:11th June 1987                      *
C     * Update :11th September 1987                 *
C     * Frozen :24th July 1988                      *
C     ***********************************************
      SUBROUTINE COMP1(IMAGE, LEVEL, CORECT, IMCNT)
C
C     Define everything as implicit REAL*8
C
      IMPLICIT REAL*8(A - H,O - Z)
C
C     Bring in COMMON block;
C
C
C     Define Common Block One;
C
      COMMON /ONE/ FRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
     &        REWARD, GLOREW
C
C     Define Frame - Max X x Y x Z ;
C
C     FRAME(0:X-1,0:Y-1,Z+1)
C
C     Final Frame N+1 is the output;
C
      REAL*8 FRAME(0:4,0:4,5)
C
C     Define the variables in the COMMON block;
C
      REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
      REAL*8 STIM(0:4,0:4,4)
      REAL*8 SUMS(0:4,0:4,4)
      INTEGER LENGTH, WIDTH, DEPTH, LEVEL, X, Y
C
C     Define the Reward array;
C
      INTEGER REWARD(0:4,0:4), GLOREW
C
C     Define the 'marks' array
C
      REAL*8 CORECT(0:4,0:4,0:4)
C
C     Image to be recognised;
C
      REAL*8 IMAGE(10,0:4,0:4)
C
C     Clear CORECT array;
C
```

```
      DO 3 I = 0, DEPTH
C
        DO 2 J = 0, LENGTH - 1
C
          DO 1 K = 0, WIDTH - 1
            CORECT(I,J,K) = 0.0D0
    1     CONTINUE
C
    2   CONTINUE
C
    3 CONTINUE
C
C      Do Check - This is differential error at output;
C
      DO 5 X = 0, LENGTH - 1
C
        DO 4 Y = 0, WIDTH - 1

          CORECT(DEPTH,X,Y) = FRAME(X,Y,LEVEL) - IMAGE(IMCNT,X,Y)
    4   CONTINUE
C
    5 CONTINUE
C
      RETURN
      END
```

FORTRAN77 subroutine ARP.BACKP

– simulates an error backpropagation element

```
C     ***********************************************
C     * Back Propagation Element Simulation         *
C     * Subroutine.                                 *
C     *                                             *
C     * - This subroutine simulates one element     *
C     *   with 25  possible inputs (plus stimulus); *
C     * - The inputs are -                          *
C     *   X(0,25) with the X(25) input = 'Stimulus': *
C     * - ACTION - This defines what the subroutine *
C     *   does on entry. ;                          *
C     * - If ACTION=0 then the element calculates   *
C     *   the summation  and encodes this using the *
C     *   Psi distribution;                         *
C     * - The output is Y (Real*8 +1./0.)           *
C     * - If ACTION=1 then the element calculates   *
C     *   the backpropagation error function in much *
C     *   the same way and outputs this as Y instead *
C     *   plus the weight update;                   *
C     * - The weights for each element, W0 etc      *
C     *   are passed through to the main program    *
C     *   plus the updates in order to preserve     *
C     *   their value;                              *
C     * - Updates operate after twice the number    *
C     *   of different images;                      *
C     * - Error is accumulated over 2*NUMIM trials  *
C     *   and then weights are updated;             *
C     * - Note that RHO is used to define the rate  *
C     *   of acceleration in this routine;          *
C     * - Stochastic Backpropagation is performed   *
C     *   using a moving average for the weights.   *
C     *   Element outputs are premultiplied         *
C     *   internally to backpropagation mode        *
C     *   to y(1-y).                                *
C     * - Program uses DOUBLE PRECISION variables;  *
C     * - Subroutines called;                       *
C     *   *NAG                                      *
C     *                                             *
C     * Author: Richard Leaver                      *
C     * Created: 5th September 1987                 *
C     * Update : 21st April 1988                    *
C     * Frozen : 20th July 1988                     *
C     ***********************************************
      SUBROUTINE ELEM(Y, ICX, ICY, Z, SUM, W, XPRIN, WPRIN, DELWGT,
     &           DELPRE, TRACNT, NUMIM, NUMEL, YPREV, STOFLG, ACTION)
C
C     Define everything as implicit REAL*8;
C
      IMPLICIT REAL*8(A - H,Q - Z)
C
C     Define COMMON blocks;
C     DUMR1 and DUMR2 are the previous REWARD and GLOREW blocks;
C
      COMMON /ONE/ FRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
     &       DUMR1, DUMR2
      COMMON /TWO/ RHO, TEMP, LAMBDA, PN
      COMMON /THREE/ X
      COMMON /FIVE/ SHIFT1, SHIFT2, SHIFT3, SHIFT4, ISHIFT, IBCNT, BAT
C
C     Define all variables as DOUBLE PRECISION;
C
      REAL*8 W(0:25), INC(0:25), X(0:25,4)
      REAL*8 XPRIN(0:25,4), WPRIN(0:25), DELWGT(0:25), DELPRE(0:25)
```

```
      REAL*8 PN, TEMP, RHO, G05CAF, VALUE
      REAL*8 THRESH, SUM, LAMBDA, Y, YPREV
C
C     Define Frame - Max X x Y x Z ;
C
C     FRAME(0:X-1,0:Y-1,Z+1)
C
C     Final Frame N+1 is the output;
C
      REAL*8 FRAME(0:4,0:4,5)
C
C     Define the variables in the COMMON block;
C
      REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
      REAL*8 STIM(0:4,0:4,4)
      REAL*8 SUMS(0:4,0:4,4)
      REAL*8 SHIFT1(0:4,0:4,4,0:20)
      REAL*8 SHIFT2(0:4,0:4,4,0:20)
      REAL*8 SHIFT3(0:4,0:4,4,0:20)
      REAL*8 SHIFT4(0:4,0:4,4,0:25)
C
C     Define Integer variables;
C
      INTEGER Z, REWARD, ACTION, NUMEL, BAT, TRACNT, NUMIM

      INTEGER LENGTH, WIDTH, DEPTH, LEVEL, CHECK, DUMR2, STOFLG
C
C     Define Integer Common block;
C
      INTEGER DUMR1(0:4,0:4)
C
C     Calculate Summation;
C
      SUM = 0.0D0
C
      DO 1 ICOUNT = 0, NUMEL
        SUM = SUM + X(ICOUNT,Z) * W(ICOUNT)
    1 CONTINUE
C
C     If ACTION is 0 then do summation and encoding;
C
      IF (ACTION .EQ. 0) THEN
C
C     Encode this summation with a special distribution;
C     Psi=(1+exp(-s/t))^-1
C
        IF (TEMP .EQ. 0.0D0) TEMP = 0.001D0
        IF (SUM .GT. 20.0D0) THEN
          Y = 1.0D0
          GO TO 2
        END IF
        IF (SUM .LT. - 20.0D0) THEN
          Y = 0.0D0
          GO TO 2
        END IF
        Y = 1.0D0 / (1.0D0+DEXP(-1.0D0*SUM/TEMP))
C
C     If STOFLG is 1 then stochastically threshold;
C
    2   IF (STOFLG .EQ. 1) THEN
```

```
            PN = G05CAF(PN)
            IF (PN .LT. Y) THEN
              Y = 1.0D0
            ELSE
              Y = 0.0D0
            END IF
          END IF
C
C     Return to main program;
C
        ELSE
C
C     Compute backpropagation function;
C
          IF (STOFLG .EQ. 1) THEN
C
C     Retrieve old summation from forward pass from before;
C     This is so the Y(1-Y) may be recomputed deterministically;
C
            SUMOLD = SUMS(ICX,ICY,Z)
C
C     Overwrite Yprevious with deterministic value;
C
            IF (SUMOLD .GT. 20.0D0) THEN
              YPREV = 1.0D0
              GO TO 3
            END IF
            IF (SUMOLD .LT. - 20.0D0) THEN
              YPREV = 0.0D0
              GO TO 3
            END IF
            YPREV = 1.0D0 / (1.0D0+DEXP(-1.0D0*SUMOLD/TEMP))
C
C     Compute Y(1-Y) deterministically and scale up by 4;
C
   3        YPREV1 = YPREV * (1.0D0-YPREV) * 4.0D0
            IF (PN .LT. YPREV1) THEN
              YPREV = 1.0D0
            ELSE
              YPREV = 0.0D0
            END IF

C
C     Now encode summation for dE/dY;
C
C     Stochastically threshold - note this is dual polarity;
C
            PN = G05CAF(PN)
            IF (PN .LT. DABS(SUM)) THEN
              THRESH = 1.0D0
              SUM = DSIGN(THRESH,SUM)
            ELSE
              SUM = 0.0D0
            END IF
C
C     AND operation to multiply dE/dY and YPREV;
C
            IF ((YPREV .GE. 0.99D0) .AND. (SUM .GE. 0.99D0))
     &           THEN
              VAL = 1.0D0
```

```
            GO TO 4
          END IF
          IF ((YPREV .GE. 0.99D0) .AND. (SUM .LE. - 0.99D0))
     &          THEN
            VAL = -1.0D0
            GO TO 4
          END IF
          VAL = 0.0D0
C
C     Y is now output to next stages;
C
    4     Y = VAL
C
C     AND operation to multiply dE/ds and XPRIN;
C
          DO 6 MCOUNT = 0, NUMEL
            IF ((Y .GE. 0.99D0) .AND. (XPRIN(MCOUNT,Z) .GE. 0.99D0))
     &          THEN
              VAL = 1.0D0
              GO TO 5
            END IF
            IF ((Y .LE. - 0.99D0) .AND. (XPRIN(MCOUNT,Z) .GE. 0.99D0))
     &          THEN
              VAL = -1.0D0
              GO TO 5
            END IF
            VAL = 0.0D0
    5       CONTINUE
C
C     Accumulate values;
C     Use Shift4 bin;
C
            SHIFT4(ICX,ICY,Z,MCOUNT) = SHIFT4(ICX,ICY,Z,MCOUNT) + VAL
C
C
C     Compute weight update and update weights if equal to the required
C     accumulation length represented by ISHIFT;
C
            IF (MOD(TRACNT,ISHIFT) .EQ. 0) THEN
C
C     Average bin contents - note that BAT does not change TRACNT,
C     therefore need to amend average to cope;
C
              WGTNEW = SHIFT4(ICX,ICY,Z,MCOUNT) / DFLOAT(ISHIFT*BAT)
C
C     Update weights - note TEMP scaling factor as required by equation;
C
              DELWGT(MCOUNT) = DELWGT(MCOUNT) + WGTNEW / (4.0D0*TEMP)
C
C     Zero bin;
C
              SHIFT4(ICX,ICY,Z,MCOUNT) = 0.0D0
            END IF
    6     CONTINUE
C
C     Change weights if twice the length of the shift register;
C
          IF (MOD(TRACNT,2*ISHIFT) .EQ. 0) THEN
C
            DO 7 J = 0, NUMEL
```

```
                VALUE = -DELWGT(J) * LAMBDA + RHO * DELPRE(J)
                WPRIN(J) = WPRIN(J) + VALUE
                DELPRE(J) = VALUE
C
C       Now zero DELWGTS;
C
                DELWGT(J) = 0.0D0
    7         CONTINUE
C
            END IF
C
C        This is where the deterministic error backpropagation is carried
C  out;
C
          ELSE
            Y = SUM * 1.0D0 / TEMP * YPREV * (1.0D0-YPREV)
C
C       Compute weight update and update weights if twice the number of
C       images;
C
            DO 8 I = 0, NUMEL
              DELWGT(I) = DELWGT(I) + Y * XPRIN(I,Z)
    8       CONTINUE
C
C       Test number of images and cycle count;
C
            IF (MOD(TRACNT,2*NUMIM) .EQ. 0) THEN
C
              DO 9 J = 0, NUMEL
                VALUE = -DELWGT(J) * LAMBDA + RHO * DELPRE(J)
                WPRIN(J) = WPRIN(J) + VALUE
                DELPRE(J) = VALUE
C
C       Now zero DELWGTS;
C
                DELWGT(J) = 0.0D0
    9         CONTINUE
C
            END IF
          END IF
        END IF
        RETURN
        END
```

FORTRAN77 subroutine ARP.PROP

– simulates an array of error backpropagation elements

```
C     ************************************************
C     * Back Propagation Array Connection            *
C     * Simulation Subroutine.                       *
C     * - Connectionist model.                       *
C     * - This subroutine simulates a 3D array of    *
C     *   Backpropagation elements with dimensions   *
C     *   Length x Width x Depth.                    *
C     * - Each element connects to all               *
C     *   neighbouring inputs on the same level;     *
C     * - Each element output then becomes the       *
C     *   centre input to the next plane of elements *
C     *   in exactly the same way;                   *
C     * - Variables; -                               *
C     *   INPUT  - input picture  (0:L-1,0:W-1)      *
C     *   OUTPUT - output picture (0:L-1,0:W-1)      *
C     *   LENGTH - Vertical length (Integer);        *
C     *   WIDTH  - Horizontal length (Integer);      *
C     *   DEPTH  - Array depth (Integer);            *
C     * - Program uses DOUBLE PRECISION variables;   *
C     * - Subroutines called;                        *
C     *   ELEM in ARP.BACKP (25 input)               *
C     *   *NAG                                       *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created: 5th September 1987                  *
C     * Update : 6th January  1988                   *
C     * Frozen :20th July 1988                       *
C     ************************************************
      SUBROUTINE PROP(ACTION, GLOFLG, STIFLG, CORECT, NUMIM, TRACNT,
     &           DWGTS, DSTIM, DWGTS1, DSTIM1, STOFLG)
C
C     Define everything as implicit REAL*8
C
      IMPLICIT REAL*8(A - H,O - Z)
C
C     Define X,Y from 0 to maximum-1
C     Define Z from 1 to maximum;
C
C     Define Frame - Max X x Y x Z ;
C
C     FRAME(0:X-1,0:Y-1,N+1)
C
C     Final Frame N+1 is the output;
C
      REAL*8 FRAME(0:4,0:4,5)
C
C     Define Error Array;
C
      REAL*8 CORECT(0:4,0:4,0:4)
C
C     Define Xinputs for the elements with N  Frames
C
      REAL*8 XIN(0:25,4), XPRIN(0:25,4)
C
C     Define Weights for N x N elements with N  Frames;
C
      REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
C
C     Define Delta weights;
C
      REAL*8 DWGTS(0:4,0:4,4,0:4,0:4)
C
C     Define Delta weights - which saves values for acceleration;
C
```

```
      REAL*8 DWGTS1(0:4,0:4,4,0:4,0:4)
      REAL*8 DELWGT(0:25)
C
C     Define Stimulus for N x N elements with N  Frames;
C
      REAL*8 STIM(0:4,0:4,4)
C
C     Define Delta Stimulus;
C
      REAL*8 DSTIM(0:4,0:4,4)
C
C     Define Delta Stimuli - which save values for acceleration;
C
      REAL*8 DSTIM1(0:4,0:4,4)
      REAL*8 DELPRE(0:25)
C
C     Define Weight input for an element;
C
      REAL*8 WIN(0:25), WPRIN(0:25)
C
C     Save SUMs for N x N elements, N Frames ;
C
      REAL*8 SUMS(0:4,0:4,4)
C
C     Frame shift register;
C
      REAL*8 SHIFT1(0:4,0:4,4,0:20)
C
C     Sum shift register;
C
      REAL*8 SHIFT2(0:4,0:4,4,0:20)
C
C     Error shift register;
C
      REAL*8 SHIFT3(0:4,0:4,4,0:20)
C
C     Error shift register;
C
      REAL*8 SHIFT4(0:4,0:4,4,0:25)
C
C     Integers;
C
      INTEGER LENGTH, WIDTH, DEPTH, ACTION, X, Y, Z, BAT
      INTEGER NUMEL, GLOFLG, STIFLG, BOLFLG, NUMIM, TRACNT, STOFLG
C
C     Define Reward array;
C
      INTEGER REWARD(0:4,0:4), GLOREW
C
C     Real *8 variables;
C
      REAL*8 RHO, TEMP, LAMBDA, PN, SUM, G05CAF, YOUT, YPREV
C
C     Define common blocks;
C
      COMMON /ONE/ FRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
     &       REWARD, GLOREW
      COMMON /TWO/ RHO, TEMP, LAMBDA, PN
      COMMON /THREE/ XIN
      COMMON /FIVE/ SHIFT1, SHIFT2, SHIFT3, SHIFT4, ISHIFT, IBCNT, BAT
C
C     Define number of elements;
C
```

```
      NUMEL = LENGTH * WIDTH
C
C     Connect elements to the Image;
C
      IF (ACTION .EQ. 0) THEN
C
        DO 7 Z = 1, DEPTH
C
C     All are connected irrespective of the position in the Array;
C
          ITOT = 0
C
          DO 2 IX = 0, LENGTH - 1
C
            DO 1 IY = 0, WIDTH - 1
              XIN(ITOT,Z) = FRAME(IX,IY,Z)
              ITOT = ITOT + 1
    1       CONTINUE
C
    2     CONTINUE
C
C     Set stimulus input;
C
          IF (STIFLG .EQ. 1) THEN
            XIN(NUMEL,Z) = 1.0D0
          ELSE
            XIN(NUMEL,Z) = 0.0D0
          END IF
C
C     Continue
C
          DO 6 X = 0, LENGTH - 1
C
            DO 5 Y = 0, WIDTH - 1
C
C     Transfer W weights into input arrays;
C     Connect to the Image;
C
              DO 4 L = 0, LENGTH - 1
C
                DO 3 M = 0, WIDTH - 1
                  IXIN = L * WIDTH + M
                  WIN(IXIN) = WEIGHT(X,Y,Z,L,M)
    3             CONTINUE
C
    4         CONTINUE
C
C     Stimulus weight input;
C
              WIN(NUMEL) = STIM(X,Y,Z)

C
C     Set YPREV to prevent it being undefined;
C
              YPREV = 0.0D0
C
C     Save the outputs;
C
              CALL ELEM(YOUT, X, Y, Z, SUM, WIN, XPRIN, WPRIN, DELWGT,
     &             DELPRE, TRACNT, NUMIM, NUMEL, YPREV, STOFLG, ACTION)
C
C     Save results;
C
```

```
                FRAME(X,Y,Z + 1) = YOUT
                SUMS(X,Y,Z) = SUM
C
    5         CONTINUE
C
    6       CONTINUE
C
    7     CONTINUE
C
        ELSE
C
C         This is where the back propagation will happen as this is the
C         learning phase;
C
          DO 16 Z = DEPTH, 1, -1
C
            DO 15 X = 0, LENGTH - 1
C
              DO 14 Y = 0, WIDTH - 1
C
C       For connecting error:-
C       If Z=DEPTH then the Xinput is only the error from the environment;
C       therefore only one input and all the weights are 1;
C       Test to make sure the array doesn't overlearn;
C       For connecting old inputs - as normal;
C
                ITOT = 0
C
                DO 9 IX = 0, LENGTH - 1
C
                  DO 8 IY = 0, WIDTH - 1
                    IF (Z .EQ. DEPTH) THEN
                      XPRIN(ITOT,Z) = FRAME(IX,IY,Z)
                      XIN(ITOT,Z) = 0.0D0
                    ELSE
                      XIN(ITOT,Z) = CORECT(Z,IX,IY)
                      XPRIN(ITOT,Z) = FRAME(IX,IY,Z)
                    END IF
                    ITOT = ITOT + 1
C
C       Rewrite this so its ok for final level case;
C
                    IF (Z .EQ. DEPTH) THEN
                      IF (DABS(CORECT(Z,X,Y)) .LT. 0.01D0) THEN
                        XIN(0,Z) = 0.0D0
                      ELSE
                        XIN(0,Z) = CORECT(Z,X,Y)
                      END IF
                    END IF
    8             CONTINUE
C
    9           CONTINUE
C
C         Set stimuli inputs as appropriate;
C
                IF (STIFLG .EQ. 1) THEN
                  XPRIN(NUMEL,Z) = 1.0D0
                ELSE
                  XPRIN(NUMEL,Z) = 0.0D0
                END IF
```

```
              XIN(NUMEL,Z) = 0.0D0
C
C     Transfer lower level W weights into input arrays and
C     connect to the error;
C     Transfer upper level W weights into input arrays and
C     connect to the inputs;
C
              DO 11 L = 0, LENGTH - 1
C
                DO 10 M = 0, WIDTH - 1
                  IXIN = L * WIDTH + M
                  IF (Z .EQ. DEPTH) THEN
                    WIN(IXIN) = 1.0D0
                    WPRIN(IXIN) = WEIGHT(X,Y,Z,L,M)
                    DELWGT(IXIN) = DWGTS(X,Y,Z,L,M)
                    DELPRE(IXIN) = DWGTS1(X,Y,Z,L,M)
                  ELSE
                    WIN(IXIN) = WEIGHT(L,M,Z + 1,X,Y)
                    WPRIN(IXIN) = WEIGHT(X,Y,Z,L,M)
                    DELWGT(IXIN) = DWGTS(X,Y,Z,L,M)
                    DELPRE(IXIN) = DWGTS1(X,Y,Z,L,M)
                  END IF
  10            CONTINUE
C
  11          CONTINUE
C
C      Set stimuli inputs as appropriate;
C
              WIN(NUMEL) = 0.0D0
              WPRIN(NUMEL) = STIM(X,Y,Z)
C
C     Input stimulus updates;
C
              DELWGT(NUMEL) = DSTIM(X,Y,Z)
              DELPRE(NUMEL) = DSTIM1(X,Y,Z)
C
C     Input Yprevious;
C
              YPREV = FRAME(X,Y,Z + 1)
C
C     Call element;
C
              CALL ELEM(YOUT, X, Y, Z, SUM, WIN, XPRIN, WPRIN, DELWGT,
     &             DELPRE, TRACNT, NUMIM, NUMEL, YPREV, STOFLG, ACTION)
C
C     Save output error;
C
              CORECT(Z - 1,X,Y) = YOUT
C
C     Save weight outputs;
C
              DO 13 L = 0, LENGTH - 1
C
                DO 12 M = 0, WIDTH - 1
                  IXIN = L * WIDTH + M
                  WEIGHT(X,Y,Z,L,M) = WPRIN(IXIN)
                  DWGTS(X,Y,Z,L,M) = DELWGT(IXIN)
                  DWGTS1(X,Y,Z,L,M) = DELPRE(IXIN)
  12            CONTINUE
C
```

```
  13          CONTINUE
C
C     Save stimulus weights;
C
              STIM(X,Y,Z) = WPRIN(NUMEL)
              DSTIM(X,Y,Z) = DELWGT(NUMEL)
              DSTIM1(X,Y,Z) = DELPRE(NUMEL)
  14        CONTINUE
C
  15      CONTINUE
C
C
C
C     Useful debugging routines;
C
C               WRITE (99,*)'Z=', Z - 1
C
C               DO 22 I = 0, LENGTH - 1
C                 WRITE (99,26) (STIM(I,J,Z),J=0,WIDTH - 1)
C                 WRITE (99,26) (CORECT(Z - 1,I,J),J=0,WIDTH - 1)
C                 WRITE (99,26) (CORECT(Z,I,J),J=0,WIDTH - 1)
C                 WRITE (99,26) (WEIGHT(0,0,Z,I,J),J=0,WIDTH - 1)
C                 WRITE (99,26) (DWGTS(0,0,Z,I,J),J=0,WIDTH - 1)
C                 WRITE (99,26) (FRAME(I,J,Z + 1),J=0,WIDTH - 1)
C   22          CONTINUE
C
C
  16    CONTINUE
C
      END IF
      RETURN
C    26 FORMAT (' ', 5(E10.4,2X))
      END
```

FORTRAN77 subroutine ARP.NEURON

– simulates an A_{R-P} element

```
C     ************************************************
C     * Associative Reward-Punish Element (ARP)      *
C     * Simulation Subroutine.                       *
C     *                                              *
C     * - This subroutine simulates one ARP element  *
C     *   with 25  possible inputs; (+ stimulus)     *
C     * - The inputs are -                           *
C     *   X(0,25) with the X(25) input = 'Stimulus': *
C     *   NB This input is different from previous   *
C     *   ARP subroutines in that it is via a        *
C     *   COMMON block. The plane Z is input instead;*
C     * - LAMBDA (Real*8) - Learning Rate ;          *
C     * - ACTION - This defines what the subroutine  *
C     *   does on entry. ;                           *
C     * - If ACTION=0 then the element calculates    *
C     *   the summation, and encodes this using the  *
C     *   Psi distribution;                          *
C     * - Note that for values of ( ) greater than   *
C     *   +- 10, the E distribution converges to     *
C     *   +- 1;                                      *
C     * - If ACTION=1 then the element calculates    *
C     *   weights according to the reward input;     *
C     * - The output is - Y (Real*8 +1./0.)          *
C     * - The weights for each element, W0 etc       *
C     *   are passed through to the main program     *
C     *   in order to preserve their values;         *
C     * - The NAG routine G05CAF is used to provide  *
C     *   uniform random numbers between 0 and 1.    *
C     * - The NAG routine G05CCF must be called in   *
C     *   the Main program to ensure non-repeatable  *
C     *   sequences.                                 *
C     * - A non uniform distribution is generated    *
C     *   by using the Inversion method.             *
C     * - COMMON blocks included for ease of         *
C     *   adaption to other methods;                 *
C     * - Program uses DOUBLE PRECISION variables;   *
C     * - Subroutines called;                        *
C     *   *NAG                                       *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created: 17th January 1987                   *
C     * Update : 30th March   1988                   *
C     * Frozen : 20th July 1988                      *
C     ************************************************
      SUBROUTINE ARP(Y, Z, SUM, REWARD, W, NUMEL, ACTION)
C
C     Define everything as implicit REAL*8;
C
      IMPLICIT REAL*8(A - H,Q - Z)
C
C     Define COMMON blocks;
C     DUMR1 and DUMR2 are the previous REWARD and GLOREW blocks;
C
      COMMON /ONE/ FRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
     &       DUMR1, DUMR2
      COMMON /TWO/ RHO, TEMP, LAMBDA, PN
      COMMON /THREE/ X
C
C     Define all variables as DOUBLE PRECISION;
C
      REAL*8 W(0:25), INC(0:25), X(0:25,4)
      REAL*8 PN, TEMP, RHO, G05CAF, VALUE
```

```
      REAL*8 THRESH, SUM, E, LAMBDA, Y
C
C      Define Frame - Max X x Y x Z ;
C
C      FRAME(0:X-1,0:Y-1,Z+1)
C
C      Final Frame N+1 is the output;
C
      REAL*8 FRAME(0:4,0:4,5)
C
C      Define the variables in the COMMON block;
C
      REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
      REAL*8 STIM(0:4,0:4,4)
      REAL*8 SUMS(0:4,0:4,4)
C
C      Define Integer variables;
C
      INTEGER Z, REWARD, ACTION, NUMEL
      INTEGER LENGTH, WIDTH, DEPTH, LEVEL, CHECK, DUMR2
C
C      Define Integer Common block;
C
      INTEGER DUMR1(0:4,0:4)
C
C
C      If ACTION is 0 then do summation and encoding;
C
      IF (ACTION .EQ. 0) THEN
C
C      Calculate Summation;
C
        SUM = 0.0D0
C
        DO 1 ICOUNT = 0, NUMEL
          SUM = SUM + X(ICOUNT,Z) * W(ICOUNT)
    1   CONTINUE
C
C      Encode this summation with a special distribution;
C      Psi=(1+exp(-s/t))^-1
C
C      This is implemented using a equiprobable random number u;
C      using the Inversion Method;
C      s=-t.ln[1/u -1]
C
        PN = G05CAF(PN)
        THRESH = -1.0D0 * TEMP * DLOG(1.0D0/PN - 1.0D0)
        IF (SUM .GT. (-1.0D0*THRESH)) THEN
          Y = 1.0D0
        ELSE
          Y = 0.0D0
        END IF
      ELSE
C
C      If ACTION is 1 then calculate the weight update;
C
C
C      Calculate Distribution Function E;
C      Note that for values of the () function greater than
C      +- 10, the E function converges to +-1
C
C      Provide protection for low temp;
C
```

```
        IF (TEMP .EQ. 0.0D0) TEMP = 0.001D0
        VALUE = SUM / TEMP
        IF (VALUE .GT. 10.0D0) THEN
          E = 1.0D0
        ELSE
          IF (VALUE .LT. (-1.0D0*10.0D0)) THEN
            E = -1.0D0
          ELSE
            E = (DEXP(VALUE) - 1.0D0) / (DEXP(VALUE) + 1.0D0)
          END IF
        END IF
C       
C       Amend weights - change in weight = INC1,INC2 ;
C
C       Precalculate -1*RHO, -1*RHO*LAMBDA, EPLU1 and EMIN1
C
        EPLU1 = E + 1.0D0
        EMIN1 = E - 1.0D0
        RHOM = -1.0D0 * RHO
        RHOL = -1.0D0 * RHO * LAMBDA
C
C        Update weights;
C
        DO 2 ICOUNT = 0, NUMEL
          IF (REWARD .EQ. 1) THEN
            IF (NINT(Y) .EQ. 1) THEN
              INC(ICOUNT) = RHOM * EMIN1 * X(ICOUNT,Z)
            ELSE
              INC(ICOUNT) = RHOM * EPLU1 * X(ICOUNT,Z)
            END IF
          ELSE
            IF (NINT(Y) .EQ. 1) THEN
              INC(ICOUNT) = RHOL * EPLU1 * X(ICOUNT,Z)
            ELSE
              INC(ICOUNT) = RHOL * EMIN1 * X(ICOUNT,Z)
            END IF
          END IF
          W(ICOUNT) = W(ICOUNT) + INC(ICOUNT)
    2   CONTINUE
C
C       End ACTION IF ;
C
      END IF
      RETURN
      END
```

FORTRAN77 subroutine ARP.CONNECT

– simulates an array of A_{R-P} feedforward elements

```
C     ***********************************************
C     * Associative Reward-Punish Element (ARP)     *
C     * Simulation Subroutine.                      *
C     * - Connectionist model.                      *
C     * - This subroutine simulates a 3D array of   *
C     *   ARP elements with dimensions              *
C     *   Length x Width x Depth.                   *
C     * - Each ARP element connects to all          *
C     *   neighbouring inputs on the same level;    *
C     * - Each ARP element output then becomes the  *
C     *   centre input to the next plane of ARPs    *
C     *   in exactly the same way;                  *
C     * - Variables; -                              *
C     *   INPUT  - input picture  (0:L-1,0:W-1)     *
C     *   OUTPUT - output picture (0:L-1,0:W-1)     *
C     *   LENGTH - Vertical length (Integer);       *
C     *   WIDTH  - Horizontal length (Integer);     *
C     *   DEPTH  - Array depth (Integer);           *
C     * - Program uses DOUBLE PRECISION variables;  *
C     * - Subroutines called;                       *
C     *   ARP in ARP.NEURON (25 input)              *
C     *   *NAG                                      *
C     *                                             *
C     * Author: Richard Leaver                      *
C     * Created: 4th May  1987                      *
C     * Update :25th September 1987                 *
C     * Frozen :20th July 1988                      *
C     ***********************************************
      SUBROUTINE ARRAY(ACTION, GLOFLG, STIFLG)
C
C     Define everything as implicit REAL*8
C
      IMPLICIT REAL*8(A - H,Q - Z)
C
C     Define X,Y from 0 to maximum-1
C     Define Z from 1 to maximum;
C
C     Define Frame - Max X x Y x Z ;
C
C     FRAME(0:X-1,0:Y-1,N+1)
C
C     Final Frame N+1 is the output;
C
      REAL*8 FRAME(0:4,0:4,5)
C
C     Define Xinputs for the ARP elements with N  Frames
C
      REAL*8 XIN(0:25,4)
C
C     Define Weights for N x N ARP elements with N  Frames;
C
      REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
C
C     Define Stimulus for N x N ARP elements with N  Frames;
C
      REAL*8 STIM(0:4,0:4,4)
C
C     Define Weight input for an ARP element;
C
      REAL*8 WIN(0:25)
C
C     Save SUMs for N x N ARP elements, N Frames ;
C
```

```
      REAL*8 SUMS(0:4,0:4,4)
C
C     Integers;
C
      INTEGER LENGTH, WIDTH, DEPTH, ACTION, X, Y, Z
      INTEGER NUMEL, GLOFLG, STIFLG
C
C     Define Reward array;
C
      INTEGER REWARD(0:4,0:4), GLOREW
C
C     Real *8 variables;
C
      REAL*8 RHO, TEMP, LAMBDA, PN, SUM, G05CAF, YOUT
C
C     Define common blocks;
C
      COMMON /ONE/ FRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
     &       REWARD, GLOREW
      COMMON /TWO/ RHO, TEMP, LAMBDA, PN
      COMMON /THREE/ XIN
C
C     Define number of elements;
C
      NUMEL = LENGTH * WIDTH
C
C     Connect ARPs to the Image;
C
      DO 9 Z = 1, DEPTH
C
C     All ARPs are connected irrespective of the position in the Array;
C
        ITOT = 0
C
        DO 2 IX = 0, LENGTH - 1
C
          DO 1 IY = 0, WIDTH - 1
            XIN(ITOT,Z) = FRAME(IX,IY,Z)
            ITOT = ITOT + 1
    1     CONTINUE
C
    2   CONTINUE
C
C     Set stimulus input;
C
        IF (STIFLG .EQ. 1) THEN
          XIN(NUMEL,Z) = 1.0D0
        ELSE
          XIN(NUMEL,Z) = 0.0D0
        END IF
C
C     Continue
C
        DO 8 X = 0, LENGTH - 1
C
          DO 7 Y = 0, WIDTH - 1
C
C     Transfer W weights into ARP input arrays;
C     Connect ARPs to the Image;
C
```

```
            DO 4 L = 0, LENGTH - 1
C
              DO 3 M = 0, WIDTH - 1
                IXIN = L * WIDTH + M
                WIN(IXIN) = WEIGHT(X,Y,Z,L,M)
    3         CONTINUE
C
    4       CONTINUE
C
C     Stimulus weight input;
C
            WIN(NUMEL) = STIM(X,Y,Z)
C
C     Transfer YOUT,SUM to array if ACTION=1; otherwise SAVE them
C     afterwards;
C
            IF (ACTION .EQ. 1) THEN
              YOUT = FRAME(X,Y,Z + 1)
              SUM = SUMS(X,Y,Z)
C
C     (If GLOFLG=1 then use Global Reward)
C
              IF (GLOFLG .EQ. 1) THEN
                IVAL = GLOREW
              ELSE
                IVAL = REWARD(X,Y)
              END IF
            END IF
            CALL ARP(YOUT, Z, SUM, IVAL, WIN, NUMEL, ACTION)
C
C     Save results if ACTION=0;
C
            IF (ACTION .EQ. 0) THEN
              FRAME(X,Y,Z + 1) = YOUT
              SUMS(X,Y,Z) = SUM
            ELSE
C
C     Save new ARP weights;
C
              DO 6 L = 0, LENGTH - 1
C
                DO 5 M = 0, WIDTH - 1
                  IXIN = L * WIDTH + M
                  WEIGHT(X,Y,Z,L,M) = WIN(IXIN)
    5           CONTINUE
C
    6         CONTINUE
C
C     Stimulus weight output;
C
              STIM(X,Y,Z) = WIN(NUMEL)
C
            END IF
    7     CONTINUE
C
    8   CONTINUE
C
    9 CONTINUE
C
      RETURN
```

END

FORTRAN77 subroutine ARP.HOP

– simulates a Hopfield element

```
C     ***********************************************
C     * Hopfield Model Simulation Subroutine;       *
C     *                                             *
C     * - This subroutine simulates one HOP element *
C     *   with 25 possible inputs; (+ stimulus)     *
C     * - The inputs are -                          *
C     *   X(0,25) with the X(25) input = 'Stimulus': *
C     * - ACTION - This defines what the subroutine *
C     *   does on entry. ;                          *
C     * - If ACTION=0 then the element calculates   *
C     *   the summation, and encodes this using a   *
C     *   binary element, with zero threshold;      *
C     * - If ACTION=1 then the element calculates   *
C     *   weights according to the inv exor rule;   *
C     * - The output is -                           *
C     *   Y (Real*8 +1./0.)                         *
C     * - The weights for each element, W0 etc      *
C     *   are passed through to the main program    *
C     *   in order to preserve their values;        *
C     * - The stimulus is doubly updated because of *
C     * - COMMON blocks included for ease of        *
C     *   adaption to other methods;                *
C     * - Program uses DOUBLE PRECISION variables;  *
C     * - Subroutines called;                       *
C     *   *NAG                                      *
C     *                                             *
C     * Author: Richard Leaver                      *
C     * Created: 17th January 1987                  *
C     * Update :5th October 1987                    *
C     * Frozen :20th July 1988                      *
C     ***********************************************
      SUBROUTINE HOP(Y, Z, SUM, REWARD, W, NUMEL, XX, YY, ACTION)
C
C     Define everything as implicit REAL*8;
C
      IMPLICIT REAL*8(A - H,O - Z)
C
C     Define COMMON blocks;
C     DUMR1 and DUMR2 are the previous REWARD and GLOREW blocks;
C
      COMMON /ONE/ FRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
     &       DUMR1, DUMR2
      COMMON /TWO/ RHO, TEMP, LAMBDA, PN
      COMMON /THREE/ X
C
C     Define all variables as DOUBLE PRECISION;
C
      REAL*8 W(0:25), INC(0:25), X(0:25,4)
      REAL*8 PN, TEMP, RHO, G05CAF, VALUE
      REAL*8 THRESH, SUM, E, LAMBDA, Y
C
C     Define Frame - Max X x Y x Z ;
C
C     FRAME(0:X-1,0:Y-1,Z+1)
C
C     Final Frame N+1 is the output;
C
      REAL*8 FRAME(0:4,0:4,5)
C
C     Define the variables in the COMMON block;
C
      REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
```

```
      REAL*8 STIM(0:4,0:4,4)
      REAL*8 SUMS(0:4,0:4,4)
C
C     Define Integer variables;
C
      INTEGER Z, REWARD, ACTION, NUMEL, XX, YY
      INTEGER LENGTH, WIDTH, DEPTH, LEVEL, CHECK, DUMR2
C
C     Define Integer Common block;
C
      INTEGER DUMR1(0:4,0:4)
C
C
C     If ACTION is 0 then do summation and encoding;
C
      IF (ACTION .EQ. 0) THEN
C
C     Calculate Summation;
C
       SUM = 0.0D0
C
       DO 1 ICOUNT = 0, NUMEL
         SUM = SUM + X(ICOUNT,Z) * W(ICOUNT)
    1  CONTINUE
C
C     Encode this summation with a binary threshold technique;
       IF (SUM .GT. 0.0D0) THEN
         Y = 1.0D0
       ELSE
         Y = 0.0D0
       END IF
      ELSE
C
C     If ACTION is 1 then calculate the weight update;
C     NB. INV EXOR Rule is calculated using a safety method;
C     Amend weights - change in weight = INC1,INC2 ;
C
       DO 2 ICOUNT = 0, NUMEL
         IF (DABS(FRAME(XX,YY,Z) - X(ICOUNT,Z)) .LT. 0.01D0)
     &           THEN
C
C     Double up for the stimulus;
C
           IF (ICOUNT .EQ. NUMEL) THEN
             INC(ICOUNT) = 2.0D0
           ELSE
             INC(ICOUNT) = 1.0D0
           END IF
         ELSE
           IF (ICOUNT .EQ. NUMEL) THEN
             INC(ICOUNT) = -2.0D0
           ELSE
             INC(ICOUNT) = -1.0D0
           END IF
         END IF
         W(ICOUNT) = W(ICOUNT) + INC(ICOUNT)
    2  CONTINUE
C
C     End ACTION IF ;
C
```

```
END IF
RETURN
END
```

FORTRAN77 subroutine ARP.CONHOP

– simulates a Hopfield Network

```
C     ************************************************
C     * Hopfield Model Simulation Subroutine         *
C     * - Model has random updating;                 *
C     * - Connectionist model.                       *
C     * - If ACTION = 1 then update and do weights;  *
C     * - Clamp over all in this case;               *
C     * - If ACTION = 0 then just update;            *
C     * - Clamp over PROPOR in this case;            *
C     * - This subroutine simulates a 3D array of    *
C     *   Hopfield elements with dimensions          *
C     *   Length x Width,Depth = 1 for Hopfield net  *
C     * - Each element connects to all               *
C     *   neighbouring inputs on the same level;     *
C     * - Each element output then becomes the       *
C     *   centre input to the next plane of elements *
C     *   in exactly the same way;                   *
C     * - Note operation of this model requires that *
C     *   the HOPFLG is switched off;                *
C     * - The flags switching on symmetry and        *
C     *   zeroing wii are redundant in this mode;    *
C     * - Variables; -                               *
C     *   INPUT  - input picture  (0:L-1,0:W-1)      *
C     *   OUTPUT - output picture (0:L-1,0:W-1)      *
C     *   LENGTH - Vertical length (Integer);        *
C     *   WIDTH  - Horizontal length (Integer);      *
C     *   DEPTH  - Array depth (Integer);            *
C     * - Program uses DOUBLE PRECISION variables;   *
C     * - Subroutines called;                        *
C     *   HOP in ARP.HOP    (25  input)              *
C     *   *NAG                                       *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created: 4th May  1987                       *
C     * Update :2nd October 1987                     *
C     * Frozen :20th July 1988                       *
C     ************************************************
      SUBROUTINE HOPMOD(ACTION, GLOFLG, STIFLG, LEN1, WID1, PROPOR)
C
C     Define everything as implicit REAL*8
C
      IMPLICIT REAL*8(A - H,Q - Z)
C
C     Define X,Y from 0 to maximum-1
C     Define Z from 1 to maximum;
C
C     Define Frame - Max X x Y x Z ;
C
C     FRAME(0:X-1,0:Y-1,N+1)
C
C     Final Frame N+1 is the output;
C
      REAL*8 FRAME(0:4,0:4,5)
C
C     Define Xinputs for the elements with N  Frames
C
      REAL*8 XIN(0:25,4)
C
C     Define Weights for N x N elements with N  Frames;
C
      REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
C
C     Define Stimulus for N x N elements with N  Frames;
C
```

```
      REAL*8 STIM(0:4,0:4,4)
C
C      Define Weight input for an element;
C
      REAL*8 WIN(0:25)
C
C      Save SUMs for N x N elements, N Frames ;
C
      REAL*8 SUMS(0:4,0:4,4)
C
C      Integers;
C
      INTEGER LENGTH, WIDTH, DEPTH, ACTION, X, Y, Z
      INTEGER NUMEL, GLOFLG, STIFLG, LEN1, WID1
C
C      Define Reward array;
C
      INTEGER REWARD(0:4,0:4), GLOREW
C
C      Real *8 variables;
C
      REAL*8 RHO, TEMP, LAMBDA, PN, SUM, G05CAF, YOUT, PROPOR
C
C      Define common blocks;
C
      COMMON /ONE/ FRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
     &        REWARD, GLOREW
      COMMON /TWO/ RHO, TEMP, LAMBDA, PN
      COMMON /THREE/ XIN
C
C      Define number of elements;
C
      NUMEL = LENGTH * WIDTH
C
C      Connect HOPs to the Image; (Main routine sets DEPTH=1)
C
      DO 10 Z = 1, DEPTH
C
C      All HOPs are connected irrespective of the position in the Array;
C
        ITOT = 0
C
        DO 2 IX = 0, LENGTH - 1
C
          DO 1 IY = 0, WIDTH - 1
            XIN(ITOT,Z) = FRAME(IX,IY,Z)
            ITOT = ITOT + 1
    1     CONTINUE
C
    2   CONTINUE
C
C      Set stimulus input;
C
        IF (STIFLG .EQ. 1) THEN
          XIN(NUMEL,Z) = 1.0D0
        ELSE
          XIN(NUMEL,Z) = 0.0D0
        END IF
C
C      Continue
C      Select X,Y position randomly;
C      Do this at least twice the number of elements to ensure at least
C      one update;
```

```
C
       DO 9 IRANNO = 1, 2 * NUMEL
         PN = G05CAF(PN)
         X = NINT(PN*(LENGTH - 1))
         PN = G05CAF(PN)
         Y = NINT(PN*(WIDTH - 1))
C
C      Transfer W weights into HOP input arrays;
C      Connect HOPs to the Image;
C
         DO 4 L = 0, LENGTH - 1
C
           DO 3 M = 0, WIDTH - 1
             IXIN = L * WIDTH + M
C
C      Zero Wii;
C
             IF ((X .EQ. L) .AND. (Y .EQ. M)) THEN
               WEIGHT(X,Y,Z,L,M) = 0.0D0
             END IF
             WIN(IXIN) = WEIGHT(X,Y,Z,L,M)
   3       CONTINUE
C
   4     CONTINUE
C
C      Stimulus weight input;
C
         WIN(NUMEL) = STIM(X,Y,Z)
C
C      (If GLOFLG=1 then use Global Reward - this is a relic and not
C      used);
C
         IF (GLOFLG .EQ. 1) THEN
           IVAL = GLOREW
         ELSE
           IVAL = REWARD(X,Y)
         END IF
C
C      Call Hopfield subroutine;
C
         CALL HOP(YOUT, Z, SUM, IVAL, WIN, NUMEL, X, Y, 0)
C
C      Save results if ACTION is 0 (ie. not training);
C      Only save results if outside clamping range;
C
         IF (ACTION .EQ. 0) THEN
           IF (PROPOR .GT. 0.00D0) THEN
             IF ((X .GT. (NINT(DFLOAT(LEN1 - 1)*PROPOR))) .AND. (Y .GT.
     &               (NINT(DFLOAT(WID1 - 1)*PROPOR)))) THEN
               FRAME(X,Y,Z) = YOUT
             END IF
           ELSE
             FRAME(X,Y,Z) = YOUT
           END IF
         END IF
C
C      Duplicate the first level as the second so that the
C      standard program checks can take place;
C
         DO 6 I = 0, LENGTH - 1
C
           DO 5 J = 0, WIDTH - 1
             FRAME(I,J,Z + 1) = FRAME(I,J,Z)
```

```
    5       CONTINUE
C
    6     CONTINUE
C
          IF (ACTION .EQ. 1) THEN
            CALL HOP(YOUT, Z, SUM, IVAL, WIN, NUMEL, X, Y, 1)
C
C     Save new HOP weights;
C
            DO 8 L = 0, LENGTH - 1
C
              DO 7 M = 0, WIDTH - 1
                IXIN = L * WIDTH + M
                WEIGHT(X,Y,Z,L,M) = WIN(IXIN)
C
C     Include Symmetry;
C
                WEIGHT(L,M,Z,X,Y) = WIN(IXIN)
    7         CONTINUE
C
    8       CONTINUE
C
C     Stimulus weight output;
C
            STIM(X,Y,Z) = WIN(NUMEL)
C
          END IF
    9   CONTINUE
C
C
   10 CONTINUE
C
      RETURN
      END
```

FORTRAN77 subroutine ARP.ERROR

– computes the error function between two images

```
C     ***********************************************
C     * Associative Reward-Punish Element (ARP)     *
C     * Subroutine to compute absolute error        *
C     * between two images;                         *
C     *                                             *
C     * - Computes error function as used in Back   *
C     *   Propagation;                              *
C     * - Two values returned...one is absolute     *
C     *   error, the other is the amended error     *
C     *   assuming Back Prop is corrected;          *
C     * - Multiple image version;                   *
C     * - Common block entry for array              *
C     *   FRAME(0:L-1,0:W-1,Z+1) and associated     *
C     *   variables;                                *
C     *                                             *
C     * Author: Richard Leaver                      *
C     * Created: 2nd June 1987                      *
C     * Update :19th October 1987                   *
C     * Frozen :20th July 1988                      *
C     ***********************************************
      SUBROUTINE ERR(IMAGE, ERRVAL, ALTER, IMCNT)
C
C     Define everything as implicit REAL*8
C
      IMPLICIT REAL*8(A - H,Q - Z)
C
C     Bring in COMMON block;
C
C
C     Define Common Block One;
C
      COMMON /ONE/ FRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
     &       REWARD, GLOREW
C
C     Define Frame - Max X x Y x Z ;
C
C     FRAME(0:X-1,0:Y-1,Z+1)
C
C     Final Frame N+1 is the output;
C
      REAL*8 FRAME(0:4,0:4,5)
C
C     Define the variables in the COMMON block;
C
      REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
      REAL*8 STIM(0:4,0:4,4)
      REAL*8 SUMS(0:4,0:4,4)
      INTEGER LENGTH, WIDTH, DEPTH, GLOREW
C
C     Define Integer Common block;
C
      INTEGER REWARD(0:4,0:4)
C
C     Image to be recognised;
C
      REAL*8 IMAGE(10,0:4,0:4)
C
C     Do Check;
C
      ERRVAL = 0.0D0
      ALTER = 0.0D0
C
```

```
      DO 2 I = 0, LENGTH - 1
C
        DO 1 J = 0, WIDTH - 1
          ERRVAL = ERRVAL + (FRAME(I,J,DEPTH + 1) - IMAGE(IMCNT,I,J)) **
     &      2
          XVAL = (FRAME(I,J,DEPTH + 1) - IMAGE(IMCNT,I,J)) ** 2
C
C       Correct for the Backpropagation;
C
          IF (XVAL .LT. 0.0025D0) XVAL = 0.0D0
          ALTER = ALTER + XVAL
    1   CONTINUE
C
    2 CONTINUE
C
      ERRVAL = ERRVAL * 0.5D0
      ALTER = ALTER * 0.5D0
      RETURN
      END
```

FORTRAN77 subroutine ARP.MATCH

– computes the error/checksum function between two images

```
C     ***********************************************
C     * Associative Reward-Punish Element (ARP)     *
C     * Subroutine to provide an Error Measure for  *
C     * the ARP output and a reference Image input;  *
C     *                                             *
C     *- If ENEFLG=1 then the standard Backprop     *
C     *  error function is computed and CHSUM       *
C     *  computed from the reciprocal;              *
C     *  Otherwise, the Checksum difference is      *
C     *  used;                                      *
C     *- Variable CHSUM returned is the probability *
C     *  that the image matches;                    *
C     *                                             *
C     * - Multiple image version;                   *
C     * - Common block entry for array              *
C     *   FRAME(0:L-1,0:W-1,Z+1) and associated     *
C     *   variables;                                *
C     *                                             *
C     * Author: Richard Leaver                      *
C     * Created: 2nd June 1987                      *
C     * Update :29th October 1987                   *
C     * Frozen :20th July 1988                      *
C     ***********************************************
      SUBROUTINE MATCH(IMAGE, LEVEL, CHSUM, IMCNT, ENEFLG)
C
C     Define everything as implicit REAL*8
C
      IMPLICIT REAL*8(A - H,O - Z)
C
C     Bring in COMMON block;
C
C     Define Common Block One;
C
      COMMON /ONE/ FRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
     &       REWARD, GLOREW
C
C     Define Frame - Max X x Y x Z ;
C
C     FRAME(0:X-1,0:Y-1,Z+1)
C
C     Final Frame N+1 is the output;
C
      REAL*8 FRAME(0:4,0:4,5)
C
C     Define the variables in the COMMON block;
C
      REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
      REAL*8 STIM(0:4,0:4,4)
      REAL*8 SUMS(0:4,0:4,4)
      INTEGER REWARD(0:4,0:4)
      INTEGER LENGTH, WIDTH, DEPTH, LEVEL, GLOREW, ENEFLG
C
C     Image to be referenced;
C
      REAL*8 IMAGE(10,0:4,0:4)
      CHSUM = 0.0D0
      KOUNT = 0
C
C     Compute Backpropagation error function;
C
      IF (ENEFLG .EQ. 1) THEN
C
```

```
      DO 2 I = 0, LENGTH - 1
C
        DO 1 J = 0, WIDTH - 1
C
          CHSUM = CHSUM + (FRAME(I,J,LEVEL) - IMAGE(IMCNT,I,J)) ** 2
C
1       CONTINUE
C
2     CONTINUE
C
        CHSUM = CHSUM * 0.5D0
      ELSE
C
C      Do Checksum over width of array...multiplying by
C      K=K+1 to stop symmetry problem;
C
C
        DO 4 I = 0, LENGTH - 1
C
          DO 3 J = 0, WIDTH - 1
            KOUNT = KOUNT + 1
            CHSUM = CHSUM + DABS(KOUNT*(FRAME(I,J,LEVEL) + 1.0D0) -
     &      KOUNT*(IMAGE(IMCNT,I,J) + 1.0D0))
3       CONTINUE
C
4     CONTINUE
C
      END IF
C
C      Compute probability
C
      IF (CHSUM .GT. 0.0D0) THEN
C
C      Multiply denominator by 2.5 to avoid problems with 1/1 case;
C
        CHSUM = 1.0D0 / (CHSUM*2.5D0)
      ELSE
        CHSUM = 1.0D0
      END IF
      RETURN
      END
```

FORTRAN77 subroutine ARP.SYMM

– imposes symmetry on weight matrix

```
C     ***********************************************
C      * Associative Reward-Punish Element (ARP)      *
C      * Symmetry Imposition Subroutine;              *
C      *                                              *
C      * - Routine imposes symmetry on Weight matrix; *
C      * - ie. Wij=Wji;                               *
C      * - Routine uses DOUBLE PRECISION variables;   *
C      *                                              *
C      * Author: Richard Leaver                       *
C      * Created:17th May  1987                       *
C      * Update :29th September 1987                  *
C      * Frozen :20th July 1988                       *
C      ***********************************************
       SUBROUTINE SYMM(INPFLG)
C
C      Define everything as implicit REAL*8
C
       IMPLICIT REAL*8(A - H,Q - Z)
C
C      Define X,Y from 0 to maximum-1;
C      Define Z from 1 to maximum;
C
C      Define Frame - Max X x Y x Z ;
C
C      FRAME(0:X-1,0:Y-1,Z+1)
C
C      Final Frame N+1 is the output;
C
       REAL*8 FRAME(0:4,0:4,5)
C
C      Define Weights for X x Y ARP elements with Z  Frames;
C
       REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
C
C      Define Stimulus for X x Y ARP elements with Z  Frames;
C
       REAL*8 STIM(0:4,0:4,4)
C
C      Save SUMs for X x Y ARP elements, Z Frames ;
C
       REAL*8 SUMS(0:4,0:4,4)
C
C      Integers;
C
       INTEGER X, Y, X1, Y1, Z
       INTEGER LENGTH, WIDTH, DEPTH
C
C      Integer array to hold Rewards for each element - final frame only;
C
       INTEGER REWARD(0:4,0:4)
C
C      Alternative for Global Rewards;
C
       INTEGER GLOREW, INPFLG
C
C      Define Common Block One;
C
       COMMON /ONE/ FRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
      &        REWARD, GLOREW
C
C      Impose symmetry; First select element;
C
```

```
      DO 5 Z = 1, DEPTH
C
        DO 4 X = 0, LENGTH - 1
C
          DO 3 Y = 0, WIDTH - 1
C
C       Now select second element;
C
            DO 2 X1 = 0, LENGTH - 1
C
              DO 1 Y1 = 0, WIDTH - 1
                IF ((X .EQ. X1) .AND. (Y .EQ. Y1)) THEN
C
C       This is only for Hopfield Model - Feedback ;
C
                  IF (INPFLG .EQ. 1) THEN
                    WEIGHT(X,Y,Z,X1,Y1) = 0.0D0
                  ELSE
                    CONTINUE
                  END IF
                ELSE
                  WEIGHT(X,Y,Z,X1,Y1) = WEIGHT(X1,Y1,Z,X,Y)
                END IF
    1         CONTINUE
C
    2       CONTINUE
C
    3     CONTINUE
C
    4   CONTINUE
C
    5 CONTINUE
C
      RETURN
      END
```

FORTRAN77 subroutine ARP.LINK

– plots connection weights between elements using links

```
C     ***********************************************
C     * Associative Reward-Punish Element (ARP)     *
C     * Subroutine to plot connection weights       *
C     * between ARP elements;                       *
C     *                                             *
C     * - Plots blue circle representing ARP        *
C     *   elements;                                 *
C     * - Plots line of variable thickness          *
C     *   representing the connection weights;      *
C     * - Local Reward version;                     *
C     * - RED   = +ve weight                        *
C     * - BLACK = -ve weight                        *
C     *                                             *
C     * - Subroutine uses *GHOST80                  *
C     *                                             *
C     * Author: Richard Leaver                      *
C     * Created: 6th June 1987                      *
C     * Update :12th May 1988                       *
C     * Frozen :20th July 1988                      *
C     ***********************************************
      SUBROUTINE LINK(ICOUNT, BOLFLG)
C
C     Define Implicit variable types;
C
      IMPLICIT REAL*4(A - H,Q - Z)
C
C     Define variables used in the Common block;
C     N.B. Have to call FRAME sonething different here;
C
      REAL*8 XFRAME(0:4,0:4,5)
      REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
      REAL*8 STIM(0:4,0:4,4)
      REAL*8 SUMS(0:4,0:4,4)
      INTEGER LENGTH, WIDTH, DEPTH, ICOUNT, BOLFLG
C
C     Define Integer Reward array;
C
      INTEGER REWARD(0:4,0:4), GLOREW
C
C     Define titles;
C
      CHARACTER*70 TITLE
      CHARACTER*30 TITLE1, TITLE2, TITLE3, TITLE4, TITLE5
C
C     Define WGTMAX as a special case REAL*8
C
      REAL*8 WGTMAX
C
C     Define INC as the thickness increment;
C
      REAL*4 INC
C
C     Define PI
C
      REAL*4 PI
C
C     Import Common block;
C
      COMMON /ONE/ XFRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
     &       REWARD, GLOREW
      PI = 3.14159
      INC = 0.0005
C
C     Define titles;
C
```

```
      IF (BOLFLG .EQ. 0) THEN
        TITLE = 'ARP INTERCONNECTION WEIGHTS'
      END IF
      IF (BOLFLG .EQ. 1) THEN
        TITLE = 'HOPFIELD INTERCONNECTION WEIGHTS'
      END IF
      IF (BOLFLG .EQ. 2) THEN
        TITLE = 'BACKPROPAGATION INTERCONNECTION WEIGHTS'
      END IF
      TITLE1 = 'LENGTH= '
      TITLE2 = 'WIDTH= '
      TITLE3 = 'LEVEL= '
      TITLE4 = 'MAX ABS WEIGHT= '
      TITLE5 = 'TRIAL NUMBER ='
C
C     Define max line thickness and circle radius;
C
      STREN = 5.0
      CIRC = 0.25
C
C     Define ARC ratio;
C
      RATIO = 1.5
C
C     Begin plots;
C
      DO 16 K = 1, DEPTH
C
C     Reset plot parameters;
C
       CALL BLUPEN
C
C     Compute maximum absolute weight on this level;
C
       WGTMAX = 0.0D0
C
       DO 4 I = 0, LENGTH - 1
C
         DO 3 J = 0, WIDTH - 1
C
           DO 2 L = 0, LENGTH - 1
C
             DO 1 M = 0, WIDTH - 1
               WGTMAX = DMAX1(DABS(WEIGHT(I,J,K,L,M)),WGTMAX)
               WGTMAX = DMAX1(DABS(STIM(I,J,K)),WGTMAX)
     1       CONTINUE
C
     2     CONTINUE
C
     3   CONTINUE
C
     4   CONTINUE
C
C     Define normalisation factor;
C
       FACTOR = STREN / WGTMAX
C
C     Begin plot routine
C
       CALL PAPER(1)
C
C     Define use of radians;
C
```

```
      CALL RADIAN
C
C     Set up plotting space for laser printer;
C     0 < X < 1 ; 0 < Y < 0.8 ar the maximum allowed values;
C     Set up for one plot;
C
      CALL PSPACE(0.1, 0.8, 0.1, 0.8)
C
C     Define mapping space onto PSPACE
C     This is for the titles;
C
      CALL MAP(0.0, 1.0, 0.0, 1.0)
C
C     Define character size;
C
      CALL CTRMAG(7)
C
C     Put titles on the plot;
C
      CALL PLOTCS(0.1, 0.9, TITLE)
      CALL PLOTCS(0.1, 0.1, TITLE1)
      CALL PLOTNI(0.2, 0.1, LENGTH)
      CALL PLOTCS(0.4, 0.1, TITLE2)
      CALL PLOTNI(0.5, 0.1, WIDTH)
      CALL PLOTCS(0.7, 0.1, TITLE3)
      CALL PLOTNI(0.8, 0.1, K)
      CALL PLOTCS(0.1, 0.04, TITLE4)
      CALL PLOTNE(0.3, 0.04, WGTMAX, 2)
      CALL PLOTCS(0.1, 0.85, TITLE5)
      CALL PLOTNI(0.3, 0.85, ICOUNT)
C
C     Draw a border round the area;
C
      CALL BORDER
C
C     Compute array size for plot;
C
      XMIN = -4.0
      XMAX = FLOAT(WIDTH) + 3.0
      YMIN = -4.0
      YMAX = FLOAT(LENGTH) + 3.0
C
C     Redefine PSPACE for the plot;
C
      CALL PSPACE(0.15, 0.75, 0.15, 0.75)
C
C     Map new limits onto this;
C
      CALL MAP(XMIN, XMAX, YMIN, YMAX)
C
C     Draw blue circles at each ARP position;
C     and a vertical line representing the stimulus weight;
C
      DO 6 I = 0, LENGTH - 1
C
        DO 5 J = 0, WIDTH - 1
          X = FLOAT(J)
          Y = FLOAT(I)
          CALL POSITN(X, Y)
          CALL BLUPEN
```

```
            CALL CIRCLE(CIRC)
    5     CONTINUE
C
    6   CONTINUE
C
C     Plot stimulus line;
C     Set thickness and colour;
C
C
        DO 9 I = 0, LENGTH - 1
C
          DO 8 J = 0, WIDTH - 1

            IFACT = NINT(FACTOR*DABS(STIM(I,J,K)))
            IF (STIM(I,J,K) .LT. 0.0D0) THEN
              CALL BLKPEN
            ELSE
              CALL REDPEN
            END IF
C
C     Draw vertical line representing the stimulus weight;
C
            X = FLOAT(J)
            Y = FLOAT(I)
C
            DO 7 ITER = 0, IFACT
              X = X + INC
              Y = Y + INC

              CALL POSITN(X, Y)
              CALL JOIN(X, Y + 0.5)
    7       CONTINUE
C
    8     CONTINUE
C
    9   CONTINUE
C
C     Now draw weights;
C
C     Compute centre of arc between current and projected position;
C
        DO 15 I = 0, LENGTH - 1
C
          DO 14 J = 0, WIDTH - 1
C
            DO 13 M = 0, LENGTH - 1
C
              DO 12 N = 0, WIDTH - 1
C
C     Set thickness and colour;
C
                IFACT = NINT(FACTOR*DABS(WEIGHT(I,J,K,M,N)))
                IF (WEIGHT(I,J,K,M,N) .LT. 0.0D0) THEN
                  CALL BLKPEN
                ELSE
                  CALL REDPEN
                END IF
                IF ((I .EQ. M) .AND. (J .EQ. N)) THEN
C
C     Source and destination coordinates are the same;
C
```

```
                XNOW = FLOAT(J)
                YNOW = FLOAT(I)
C
                DO 10 ITER = 0, IFACT
                  XNOW = XNOW + INC
                  YNOW = YNOW + INC
C
C     Draw a horizontal line down representing weight attached
C     to input;
C
                  CALL POSITN(XNOW, YNOW)
                  CALL JOIN((XNOW - 0.5), YNOW)
  10              CONTINUE
C
              ELSE
C
C     Connect an ARC between the two coordinates;
C     Compute ARC centre first;
C
                XNOW = FLOAT(J)
                YNOW = FLOAT(I)
                XTHEN = FLOAT(N)
                YTHEN = FLOAT(M)
C
                DO 11 ITER = 0, IFACT
                  XTHEN = XTHEN + INC
                  YTHEN = YTHEN + INC
                  XNOW = XNOW + INC
                  YNOW = YNOW + INC
                  XCENT = (XTHEN - XNOW) / 2.0 + XNOW
                  YCENT = (YTHEN - YNOW) / 2.0 + YNOW
C
C     Plot ARC
C
                  CALL POSITN(XCENT, YCENT)
                  CALL ARCELL(XNOW, YNOW, PI, RATIO)
  11              CONTINUE
C
              END IF
  12          CONTINUE
C
  13        CONTINUE
C
  14      CONTINUE
C
  15    CONTINUE
C
C     Close picture on this level;
C
       CALL FRAME
  16 CONTINUE
C
     RETURN
     END
```

FORTRAN77 subroutine ARP.LINE

– plots connection weights between elements using blocks

```
C     ************************************************
C     * Associative Reward-Punish Element (ARP)      *
C     * Subroutine to plot connection weights        *
C     * between ARP elements;                        *
C     *                                              *
C     * - Plots L x W weights with stimulus;         *
C     *                                              *
C     * - Draws a block representing an ARP element  *
C     * - Horizontal shading across block is         *
C     *   red for positive weights and black for     *
C     *   negative weights;                          *
C     *                                              *
C     * - Subroutine uses *GHOST80                   *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created:11th August 1987                     *
C     * Update :1st October 1987                     *
C     * Frozen :20th July 1988                       *
C     ************************************************
      SUBROUTINE LINE(ICOUNT, BOLFLG)
C
C     Define Implicit variable types;
C
      IMPLICIT REAL*4(A - H,Q - Z)
C
C     Define variables used in the Common block;
C     N.B. Have to call FRAME sonething different here;
C
      REAL*8 XFRAME(0:4,0:4,5)
      REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
      REAL*8 STIM(0:4,0:4,4)
      REAL*8 SUMS(0:4,0:4,4)
      INTEGER LENGTH, WIDTH, DEPTH, ICOUNT, BOLFLG
C
C     Define Integer Reward array;
C
      INTEGER REWARD(0:4,0:4), GLOREW
C
C     Define titles;
C
      CHARACTER*70 TITLE
      CHARACTER*30 TITLE1, TITLE2, TITLE3
C
C     Define WGTMAX as a special case REAL*8
C
      REAL*8 WGTMAX
C
C     Define Polarity array;
C
      INTEGER POL(0:25)
C
C     Define Xminus arrays:
C
      REAL*4 XMINUS(0:25)
      REAL*4 XMINU1(0:25)
C
C     Define Xplus arrays:
C
      REAL*4 XPLUS(0:25)
      REAL*4 XPLUS1(0:25)
C
C     Define Yminus arrays:
C
```

```
      REAL*4 YMINUS(0:25)
      REAL*4 YMINU1(0:25)
C
C     Define Yplus arrays:
C
      REAL*4 YPLUS(0:25)
      REAL*4 YPLUS1(0:25)
C
C     Define INC as the vertical line step;
C
      REAL*4 INC
C
C     Define MAXSPC as the Maximum permitted weight space;
C
      REAL*4 MAXSPC
C
C     Define LSTEP and WSTEP as the increments across the block;
C
      REAL*4 LSTEP, WSTEP
C
C     Define PI
C
      REAL*4 PI
C
C     Import Common block;
C
      COMMON /ONE/ XFRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
     &       REWARD, GLOREW
C
C     Define variables;
C
      IF (LENGTH .GT. 1) THEN
        LSTEP = 90.0 / FLOAT(LENGTH - 1)
      ELSE
        LSTEP = 100.0
      END IF
      IF (WIDTH .GT. 1) THEN
        WSTEP = 90.0 / FLOAT(WIDTH - 1)
      ELSE
        WSTEP = 100.0
      END IF
      PI = 3.14159
      INC = 2.0
      IFLAG = 0
C
C     Set Flag here for blank space or not;
C
      IBLANK = 1
C
C     Define titles;
C
      IF (BOLFLG .EQ. 0) THEN
        TITLE = 'ARP INTERCONNECTION WEIGHTS'
      END IF
      IF (BOLFLG .EQ. 1) THEN
        TITLE = 'HOPFIELD INTERCONNECTION WEIGHTS'
      END IF
      IF (BOLFLG .EQ. 2) THEN
        TITLE = 'BACKPROPAGATION INTERCONNECTION WEIGHTS'
      END IF
```

```
      TITLE1 = 'LEVEL= '
      TITLE2 = 'MAX ABS WEIGHT= '
      TITLE3 = 'TRIAL NUMBER ='
C
C     Define MAXSPC;
C
      MAXSPC = 8.5
C
C     Begin plots;
C
      DO 15 K = 1, DEPTH
C
C     Reset plot parameters;
C
       CALL BLUPEN
C
C     Compute maximum absolute weight on this level;
C
       WGTMAX = 0.0D0
C
       DO 4 I = 0, LENGTH - 1
C
         DO 3 J = 0, WIDTH - 1
C
           DO 2 L = 0, LENGTH - 1
C
             DO 1 M = 0, WIDTH - 1
               WGTMAX = DMAX1(DABS(WEIGHT(I,J,K,L,M)),WGTMAX)
               WGTMAX = DMAX1(DABS(STIM(I,J,K)),WGTMAX)
    1        CONTINUE
C
    2      CONTINUE
C
    3    CONTINUE
C
    4   CONTINUE
C
C     Define normalisation factor;
C
       FACTOR = MAXSPC / WGTMAX
C
C     Begin plot routine
C
       CALL PAPER(1)
C
C     Define use of radians;
C
       CALL RADIAN
C
C     Set up plotting space for laser printer;
C     0 < X < 1 ; 0 < Y < 0.8 are the maximum allowed values;
C     Set up for one plot;
C
       CALL PSPACE(0.1, 0.8, 0.1, 0.8)
C
C     Define mapping space onto PSPACE
C     This is for the titles;
C
       CALL MAP(0.0, 100.0, 0.0, 100.0)
C
C     Define character size;
C
```

```
        CALL CTRMAG(7)
C
C     Put titles on the plot;
C
        CALL PLOTCS(50., 95., TITLE)
        CALL PLOTCS(70., 4., TITLE1)
        CALL PLOTNI(80., 4., K)
        CALL PLOTCS(10., 4., TITLE2)
        CALL PLOTNE(30., 4., WGTMAX, 2)
        CALL PLOTCS(50., 90., TITLE3)
        CALL PLOTNI(70., 90., ICOUNT)
C
C     Draw a border round the area;
C
        CALL BORDER
C
C     Compute array size for plot;
C
        XMIN = -11.0
        XMAX = 100.0
        YMIN = -10.0
        YMAX = 120.0
C
C     Plot each square representing an element;
C     Initialise;
C
        YYN = 0.2
C
        DO 14 IX = 0, LENGTH - 1
C
          XXN = 0.2
C
          DO 13 IY = 0, WIDTH - 1
C
C     Max PSPACE available is PSPACE(0.20, 0.70, 0.20, 0.70)
C     Divide as;
            IF ((LENGTH + WIDTH) .LT. 6) THEN
              XDIV = 0.6 / FLOAT(LENGTH + WIDTH)
              YDIV = 0.6 / FLOAT(LENGTH + WIDTH)
            ELSE
              XDIV = 1.0 / FLOAT(LENGTH + WIDTH)
              YDIV = 1.0 / FLOAT(LENGTH + WIDTH)
            END IF
C
C     Compute pspace coords;
C
            XXX = XXN + XDIV
            YYY = YYN + YDIV
C
C     Redefine PSPACE for the plot;
C
            CALL PSPACE(XXN, XXX, YYN, YYY)
C
C     Map new limits onto this;
C
            CALL MAP(XMIN, XMAX, YMIN, YMAX)
C
C     Draw Element outline;
C
            CALL POSITN(-11.0, -10.0)
```

```
            CALL JOIN(-11.0, 120.0)
            CALL JOIN(10.0, 120.0)
            CALL JOIN(10.0, 100.0)
            CALL JOIN(100.0, 100.0)
            CALL JOIN(100.0, -10.0)
            CALL JOIN(-11.0, -10.0)
C
C       Define Number of Elements;
C
            NUMEL = LENGTH * WIDTH
C
C     Compute limits for the stimulus weight;
C
            XCOORD = 0.0
            YCOORD = 110.0
            XMINUS(NUMEL) = XCOORD - FACTOR * ABS(STIM(IX,IY,K))
            XPLUS(NUMEL) = XCOORD + FACTOR * ABS(STIM(IX,IY,K))
            YMINUS(NUMEL) = YCOORD - FACTOR * ABS(STIM(IX,IY,K))
            YPLUS(NUMEL) = YCOORD + FACTOR * ABS(STIM(IX,IY,K))
            XMINU1(NUMEL) = XCOORD - FACTOR * WGTMAX
            XPLUS1(NUMEL) = XCOORD + FACTOR * WGTMAX
            YMINU1(NUMEL) = YCOORD - FACTOR * WGTMAX
            YPLUS1(NUMEL) = YCOORD + FACTOR * WGTMAX
C
C     To Blank or not to Blank;
C
            IF (IBLANK .NE. 1) THEN
              XMINU1(NUMEL) = XMINUS(NUMEL)
              XPLUS1(NUMEL) = XPLUS(NUMEL)
              YMINU1(NUMEL) = YMINUS(NUMEL)
              YPLUS1(NUMEL) = YPLUS(NUMEL)
            END IF
C
C     Compute polarity;- Use Polflg here;
C
            IF (STIM(IX,IY,K) .GE. 0.0) THEN
              POL(NUMEL) = 1
            ELSE
              POL(NUMEL) = 0
            END IF
C
C     Compute limits for weights given that IX,IY is this element
C     and IIX, IIY are the destination elements;
C
            IIX = 0
            KCOUNT = 0
C
            DO 6 I = 0, 90, LSTEP
              IIY = 0
C
              DO 5 J = 0, 90, WSTEP
                XCOORD = FLOAT(J)
                YCOORD = FLOAT(I)
                XMINUS(KCOUNT) = XCOORD - FACTOR * ABS(WEIGHT(IX,IY,K,
     &          IIX,IIY))
                XPLUS(KCOUNT) = XCOORD + FACTOR * ABS(WEIGHT(IX,IY,K,
     &          IIX,IIY))
                YMINUS(KCOUNT) = YCOORD - FACTOR * ABS(WEIGHT(IX,IY,K,
     &          IIX,IIY))
                YPLUS(KCOUNT) = YCOORD + FACTOR * ABS(WEIGHT(IX,IY,K,
```

```
     &          IIX,IIY))
                XMINU1(KCOUNT) = XCOORD - FACTOR * WGTMAX
                XPLUS1(KCOUNT) = XCOORD + FACTOR * WGTMAX
                YMINU1(KCOUNT) = YCOORD - FACTOR * WGTMAX
                YPLUS1(KCOUNT) = YCOORD + FACTOR * WGTMAX
C
C     To Blank or not to Blank;
C
                IF (IBLANK .NE. 1) THEN
                  XMINU1(KCOUNT) = XMINUS(KCOUNT)
                  XPLUS1(KCOUNT) = XPLUS(KCOUNT)
                  YMINU1(KCOUNT) = YMINUS(KCOUNT)
                  YPLUS1(KCOUNT) = YPLUS(KCOUNT)
                END IF
C
C     Compute polarity;
C
                IF (WEIGHT(IX,IY,K,IIX,IIY) .GE. 0.0D0) THEN
                  POL(KCOUNT) = 1
                ELSE
                  POL(KCOUNT) = 0
                END IF
                IIY = IIY + 1
                KCOUNT = KCOUNT + 1
    5         CONTINUE
C
              IIX = IIX + 1
C
    6       CONTINUE

C
C     Draw Horizontal Lines;
C
            CALL POSITN(XMIN, YMIN)
C
            ID = -1
C
            DO 12 MCOUNT = 1, 2
              ID = -1 * ID
C
              DO 11 Y = YMIN, YMAX, INC
                CALL POSITN(XMIN, Y)
C
                DO 10 X = XMIN, XMAX
C
C       Examine X and Y boundary files;
C
                  IF (ID .EQ. 1) THEN
C
C     Draw lines leaving space for squares;
C
                    IP1FLG = 0
C
                    DO 7 LL = 0, NUMEL
                      IF ((X .GT. XMINU1(LL)) .AND. (X .LT. XPLUS1(LL))
     &                      .AND. (Y .GT. YMINU1(LL)) .AND. (Y .LT.
     &                      YPLUS1(LL))) THEN
                        IFLAG = 1
                        IF (IFLAG .NE. IP1FLG) THEN
                          IP1FLG = IFLAG
```

```
                        GO TO 9
                      END IF
                    ELSE
                      IFLAG = 0
                      IF (IFLAG .NE. IP1FLG) THEN
                        IP1FLG = IFLAG
                        GO TO 9
                      END IF
                    END IF
  7               CONTINUE
C
                ELSE
C
C     Call polarity pens & plot in squares
C
                  DO 8 LL = 0, NUMEL
                    IF (POL(LL) .EQ. 1) THEN
                      CALL REDPEN
                    ELSE
                      CALL BLKPEN
                    END IF
                    IP2FLG = 1
C
                    IF ((X .GT. XMINUS(LL)) .AND. (X .LT. XPLUS(LL))
     &                    .AND. (Y .GT. YMINUS(LL)) .AND. (Y .LT.
     &                    YPLUS(LL))) THEN
                      IFLAG = 0
                      IF (IFLAG .NE. IP2FLG) THEN
                        IP2FLG = IFLAG
                        GO TO 9
                      END IF
                    ELSE
                      IFLAG = 1
                      IF (IFLAG .NE. IP2FLG) THEN
                        IP2FLG = IFLAG
                        GO TO 9
                      END IF
                    END IF
  8               CONTINUE
C
                END IF
C
C       Plot;
C
  9             IF (IFLAG .EQ. 0) THEN
C
C       Put boundaries in;
C
                  IF ((X .GT. 10.0) .AND. (Y .GT. 100.0))
     &                THEN
                    CALL POSITN(X, Y)
                  ELSE
                    CALL JOIN(X, Y)
                  END IF
                ELSE
                  CALL POSITN(X, Y)
                END IF
                CALL BLUPEN
 10           CONTINUE
C
```

```
  11          CONTINUE
C
  12        CONTINUE
C
C     Update x coords;
C
            IF ((LENGTH + WIDTH) .LT. 5) THEN
              XXN = XXX + 0.05
            ELSE
              XXN = XXX + 0.005
            END IF
  13      CONTINUE
C
C     Update y coords;
C
           IF ((LENGTH + WIDTH) .LT. 5) THEN
             YYN = YYY + 0.05
           ELSE
             YYN = YYY + 0.005
           END IF
  14    CONTINUE
C
C     Close picture on this level;
C
       CALL FRAME
  15 CONTINUE
C
      RETURN
      END
```

FORTRAN77 subroutine ARP.PICT

– plots input–output vectors

```
C     ***********************************************
C     * Associative Reward-Punish Element (ARP)     *
C     * Subroutine to plot ARP images;              *
C     *                                             *
C     * - Plots integer in ARP image arrays;        *
C     * - Multiple image version;                   *
C     *                                             *
C     * Author: Richard Leaver                      *
C     * Created: 8th June 1987                      *
C     * Update :11th September 1987                 *
C     * Frozen :20th July 1988                      *
C     ***********************************************
      SUBROUTINE PICT(IMAGE, LENGTH, WIDTH, TITLE4, IMCNT)
C
C     Define Implicit variable types;
C
      IMPLICIT REAL*4(A - H,O - Z)
C
C     Define Image array;
C
      REAL*8 IMAGE(10,0:4,0:4)
      INTEGER LENGTH, WIDTH
C
C     Define titles;
C
      CHARACTER*70 TITLE
      CHARACTER*30 TITLE1, TITLE2, TITLE3, TITLE4
C
C     Define titles;
C
      TITLE = 'ARP ARRAY INPUT/OUTPUT'
      TITLE1 = 'LENGTH= '
      TITLE2 = 'WIDTH= '
      TITLE3 = 'IMAGE= '
C
C     Set plot parameters;
C
      CALL BLUPEN
C
C     Begin plot routine
C
      CALL PAPER(1)
C
C     Set up plotting space for laser printer;
C     0 < X < 1 ; 0 < Y < 0.8 ar the maximum allowed values;
C     Set up for one plot;
C
      CALL PSPACE(0.1, 0.8, 0.1, 0.8)
C
C     Define mapping space onto PSPACE
C     This is for the titles;
C
      CALL MAP(0.0, 1.0, 0.0, 1.0)
C
C     Define character size;
C
      CALL CTRMAG(7)
C
C     Put titles on the plot;
C
      CALL PLOTCS(0.1, 0.9, TITLE)
      CALL PLOTCS(0.1, 0.1, TITLE1)
```

```
      CALL PLOTNI(0.2, 0.1, LENGTH)
      CALL PLOTCS(0.4, 0.1, TITLE2)
      CALL PLOTNI(0.5, 0.1, WIDTH)
      CALL PLOTCS(0.4, 0.05, TITLE3)
      CALL PLOTNI(0.5, 0.05, IMCNT - 1)
      CALL PLOTCS(0.1, 0.85, TITLE4)
C
C     Draw a border round the area;
C
      CALL BLKPEN
      CALL BORDER
C
C     Compute array size for plot;
C
      XMIN = -1.0
      XMAX = FLOAT(WIDTH) + 1.0
      YMIN = -1.0
      YMAX = FLOAT(LENGTH) + 1.0
C
C     Redefine PSPACE for the plot;
C
      CALL PSPACE(0.15, 0.75, 0.15, 0.75)
C
C     Map new limits onto this;
C
      CALL MAP(XMIN, XMAX, YMIN, YMAX)
C
C     Draw blue character at each ARP position;
C     and a vertical line representing the stimulus weight;
C
C     Increase character size & call a red pen;
C
      CALL CTRMAG(20)
      CALL REDPEN
C
      DO 2 I = 0, LENGTH - 1
C
        DO 1 J = 0, WIDTH - 1
          X = FLOAT(J)
          Y = FLOAT(I)
          CALL POSITN(X, Y)
          VAL = IMAGE(IMCNT,I,J)
C
C     Plot real number with ldp.
C
          CALL PLOTNF(X, Y, VAL, 1)
    1   CONTINUE
C
    2 CONTINUE
C
C     Close picture on this level;
C
      CALL FRAME
      RETURN
      END
```

FORTRAN77 subroutine ARP.NPLOT

– plots noise performance

```
C     ************************************************
C     * Special Purpose GHOST plotting subroutine;   *
C     *                                              *
C     * - Routine generates plots of ARP simulations.*
C     * - Plot is for selected ARP element (X,Y,Z)   *
C     *   and draws the element performance in terms *
C     *   of %Performance against Standard Deviation;*
C     * - max N allowed for = 10000                  *
C     * - Routine uses *GHOST80                      *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created:2nd April 1986                       *
C     * Update :1st October 1987                     *
C     * Frozen :20th July 1988                       *
C     ************************************************
C
      SUBROUTINE NPLOT(PLNOI, BOLFLG)
C
C     Define Implicit variable type;
C
      IMPLICIT REAL*4(A - H,Q - Z)
C
C     Define array;
C
      REAL*4 PLNOI(0:10000)
C
C     Define integers;
C
      INTEGER BOLFLG
C
C     Define Character variables;
C
      CHARACTER*70 TITLE, TITLE1, TITLE2
C
C     Enter titles;
C
      IF (BOLFLG .EQ. 0) THEN
        TITLE = ' ARP SIMULATION - ELEMENT NOISE PERFORMANCE'
      END IF
      IF (BOLFLG .EQ. 1) THEN
        TITLE = ' HOPFIELD SIMULATION - ELEMENT NOISE PERFORMANCE'
      END IF
      IF (BOLFLG .EQ. 2) THEN
        TITLE = ' BACKPROPAGATION SIMULATION - ELEMENT NOISE PERFORMANCE
     &'
      END IF
      TITLE1 = 'STANDARD DEVIATION'
      TITLE2 = '% RECOGNITION '
C
      XMIN = 0.0
      XMAX = 1.0
      YMIN = 0.0
      YMAX = 100.0
C
C     Start plotting routine ;
C
      CALL PAPER(1)
C
C     Define use of degrees;
C
      CALL DEGREE
C
C     Set up plotting space for laser printer ;
C     0 < X < 1  ; 0 < Y < 0.8 are the maximum allowed values ;
C     Set up for one plot;
```

```
C
      CALL PSPACE(0.1, 0.8, 0.1, 0.8)
C
C     Define mapping space onto PSPACE
C     This is for the titles  ;
C
      CALL MAP(0.0, 1.0, 0.0, 1.0)
C
C     Define character size ;
C
      CALL CTRMAG(7)
C
C     Use Blue Pen;
C
      CALL BLUPEN
C
C     Put titles on the plot ;
C
      CALL PLOTCS(0.1, 0.97, TITLE)
      CALL PLOTCS(0.4, 0.01, TITLE1)
      CALL CTRORI(90.0)
      CALL PLOTCS(0.02, 0.50, TITLE2)
      CALL CTRORI(0.0)
C
C     Draw a border around the PSPACE defined area ;
C
      CALL BORDER
C
C     Redefine PSPACE for the plot ;
C
      CALL PSPACE(0.15, 0.75, 0.15, 0.75)
C
C     Define maximum and minimum values for X and Y ;
C
      CALL MAP(XMIN, XMAX, YMIN, YMAX)
C
C     Draw axes in black;
C
      CALL BLKPEN
      CALL AXORIG(XMIN, YMIN)
      CALL AXORIG(XMIN, YMIN)
      CALL BORDER
      CALL SCALES
      CALL BORDER
C
C     Plot weights;
C
      CALL REDPEN
C
      CALL POSITN(0., 0.)
C
      DO 1 I = 0, 10000
        X = FLOAT(I) / 10000.0
        Y = PLNOI(I)
        CALL JOIN(X, Y)
    1 CONTINUE
C
C     Stop picture on this level;
C
      CALL FRAME
      RETURN
      END
```

FORTRAN77 subroutine ARP.ELPLOT

– plots selected element weights

```
C     ************************************************
C     * Special Purpose GHOST plotting subroutine;   *
C     *                                              *
C     * - Routine generates plots of ARP simulations.*
C     * - Plot is for selected ARP element (X,Y,Z)   *
C     *   and draws the element weights on a self    *
C     *   scaled graph;                              *
C     * - max N allowed for = 10000                  *
C     * - Routine uses *GHOST80                      *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created:2nd April 1986                       *
C     * Update :26th October 1987                    *
C     * Frozen :20th July 1988                       *
C     ************************************************
C
      SUBROUTINE ELPLOT(VALUES, STIVAL, LENGTH, WIDTH, MAXCNT, XELEM,
     &           YELEM, ZELEM, BOLFLG)
C
C     Define Implicit variable type;
C
      IMPLICIT REAL*4(A - H,Q - Z)
C
C     Define arrays;
C
      REAL*4 VALUES(0:4,0:4,10000)
      REAL*4 STIVAL(10000)
C
C     Define Real*8 factor;
C
      REAL*8 FACT
C
C     Define Integer values;
C
      INTEGER XELEM, YELEM, ZELEM, MAXCNT, LENGTH, WIDTH, BOLFLG
C
C     Define Character variables;
C
      CHARACTER*70 TITLE, TITLE1, TITLE2, TITLE3
      CHARACTER*5 TITLE4, TITLE5, TITLE6
C
C     Enter titles;
C
      IF (BOLFLG .EQ. 0) THEN
        TITLE = ' ARP SIMULATION - ELEMENT ADAPTIVE WEIGHTS'
      END IF
      IF (BOLFLG .EQ. 1) THEN
        TITLE = ' HOPFIELD SIMULATION - ELEMENT ADAPTIVE WEIGHTS'
      END IF
      IF (BOLFLG .EQ. 2) THEN
        TITLE = ' BACKPROPAGATION SIMULATION - ELEMENT ADAPTIVE WEIGHTS'
      END IF
      TITLE1 = 'NO OF TRIALS'
      TITLE2 = 'ADAPTIVE WEIGHT VALUES '
      TITLE3 = 'ELEMENT COORDINATES'
      TITLE4 = 'X = '
      TITLE5 = 'Y = '
      TITLE6 = 'Z = '
C
C     Initialise X and Y coordinates to suitable values
C     Self scale;
C
```

```
      WGTMAX = 1.0E-8
C
      DO 3 L = 0, LENGTH - 1
C
        DO 2 M = 0, WIDTH - 1
C
          DO 1 J = 1, MAXCNT
            WGTMAX = AMAX1(ABS(VALUES(L,M,J)),WGTMAX)
            WGTMAX = AMAX1(ABS(STIVAL(J)),WGTMAX)
    1     CONTINUE
C
    2   CONTINUE
C
    3 CONTINUE
C
C     Scale up WGTMAX to be a 'nice' plot value;
C
C
      DO 5 I = -7, 7
        FACT = 10.0D0 ** I
C
        DO 4 J = 0, 9
          FACTOR = FACT + DFLOAT(J) * FACT
          IF (FACTOR/WGTMAX .GE. 1.0) GO TO 6
    4   CONTINUE
C
    5 CONTINUE
C
    6 XMIN = 0.0
      XMAX = FLOAT(MAXCNT)
      YMIN = -1 * FACTOR
      YMAX = FACTOR
C
C     Start plotting routine ;
C
      CALL PAPER(1)
C
C     Define use of degrees;
C
      CALL DEGREE
C
C     Set up plotting space for laser printer ;
C     0 < X < 1  ; 0 < Y < 0.8 are the maximum allowed values ;
C     Set up for one plot;
C
      CALL PSPACE(0.1, 0.8, 0.1, 0.8)
C
C     Define mapping space onto PSPACE
C     This is for the titles  ;
C
      CALL MAP(0.0, 1.0, 0.0, 1.0)
C
C     Define character size ;
C
      CALL CTRMAG(7)
C
C     Use Blue Pen;
C
      CALL BLUPEN
C
C     Put titles on the plot ;
C
```

```
      CALL PLOTCS(0.1, 0.97, TITLE)
      CALL PLOTCS(0.4, 0.020, TITLE1)
      CALL CTRORI(90.0)
      CALL PLOTCS(0.02, 0.50, TITLE2)
      CALL CTRORI(0.0)
      CALL PLOTCS(0.1, 0.020, TITLE3)
      CALL PLOTCS(0.1, 0.01, TITLE4)
      CALL PLOTNI(0.3, 0.01, XELEM)
      CALL PLOTCS(0.4, 0.01, TITLE5)
      CALL PLOTNI(0.6, 0.01, YELEM)
      CALL PLOTCS(0.7, 0.01, TITLE6)
      CALL PLOTNI(0.9, 0.01, ZELEM)
C
C     Draw a border around the PSPACE defined area ;
C
      CALL BORDER
C
C     Redefine PSPACE for the plot ;
C
      CALL PSPACE(0.15, 0.75, 0.15, 0.75)
C
C     Define maximum and minimum values for X and Y ;
C
      CALL MAP(XMIN, XMAX, YMIN, YMAX)
C
C     Draw axes in black;
C
      CALL BLKPEN
      CALL AXORIG(XMIN, YMIN)
      CALL SCALES
      CALL BORDER
C
C     Plot weights;
C
      CALL REDPEN
C
      DO 9 L = 0, LENGTH - 1
C
        DO 8 M = 0, WIDTH - 1
          CALL POSITN(0., 0.)
C
          DO 7 J = 1, MAXCNT
C
C     Plot weights - Red for +ve and Black for -ve ;
C
            IF (VALUES(L,M,J) .LT. 0.0) THEN
              CALL BLKPEN
            ELSE
              CALL REDPEN
            END IF
            CALL JOIN(FLOAT(J), VALUES(L,M,J))
    7     CONTINUE
C
    8   CONTINUE
C
    9 CONTINUE
C
C     Plot stimulus weight in green;
C
      CALL GRNPEN
```

```
      CALL POSITN(0., 0.)
C
      DO 10 J = 1, MAXCNT
        CALL JOIN(FLOAT(J), STIVAL(J))
   10 CONTINUE
C
C      Stop picture on this level;
C
      CALL FRAME
      RETURN
      END
```

FORTRAN77 subroutine ARP.CAPT

– plots global reinforcement

```
C     ************************************************
C     * Special Purpose GHOST plotting subroutine;   *
C     *                                              *
C     * - Routine generates plots of ARP simulation  *
C     *   CASH reward values;                        *
C     * - max N allowed for = 100000                 *
C     * - Routine uses *GHOST80                      *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created:2nd April 1986                       *
C     * Update :20th October 1987                    *
C     * Frozen :20th July 1988                       *
C     ************************************************
C
      SUBROUTINE CAPT(CSHPLT)
C
C     Define Implicit variable type;
C
      IMPLICIT REAL*4(A - H,Q - Z)
C
C     Define array;
C
      REAL*4 CSHPLT(10000)
C
C     Define Character variables;
C
      CHARACTER*70 TITLE, TITLE1, TITLE2
C
C     Enter titles;
C
      TITLE = ' ARP SIMULATION - ARRAY REWARD VALUES'
      TITLE1 = 'NO OF TRIALS'
      TITLE2 = 'REWARD VALUES'
C
      XMIN = 0.0
      XMAX = 10000.0
      YMIN = 0.0
      YMAX = 1.0
C
C     Start plotting routine ;
C
      CALL PAPER(1)
C
C     Define use of degrees;
C
      CALL DEGREE
C
C     Set up plotting space for laser printer ;
C     0 < X < 1  ; 0 < Y < 0.8 are the maximum allowed values ;
C     Set up for one plot;
C
      CALL PSPACE(0.1, 0.8, 0.1, 0.8)
C
C     Define mapping space onto PSPACE
C     This is for the titles  ;
C
      CALL MAP(0.0, 1.0, 0.0, 1.0)
C
C     Define character size ;
C
      CALL CTRMAG(7)
C
C     Use Blue Pen;
C
```

```
      CALL BLUPEN
C
C     Put titles on the plot ;
C
      CALL PLOTCS(0.1, 0.97, TITLE)
      CALL PLOTCS(0.4, 0.1, TITLE1)
      CALL CTRORI(90.0)
      CALL PLOTCS(0.02, 0.50, TITLE2)
      CALL CTRORI(0.0)
C
C     Draw a border around the PSPACE defined area ;
C
      CALL BORDER
C
C     Redefine PSPACE for the plot ;
C
      CALL PSPACE(0.15, 0.75, 0.15, 0.75)
C
C     Define maximum and minimum values for X and Y ;
C
      CALL MAP(XMIN, XMAX, YMIN, YMAX)
C
C     Draw axes in black;
C
      CALL BLKPEN
      CALL AXORIG(XMIN, YMIN)
      CALL SCALES
      CALL BORDER
C
C     Plot weights;
C
      CALL REDPEN
C
      CALL POSITN(0., 0.)
C
      DO 1 I = 1, 10000
        X = FLOAT(I)
        Y = CSHPLT(I)
        CALL JOIN(X, Y)
    1 CONTINUE
C
C     Stop picture on this level;
C
      CALL FRAME
      RETURN
      END
```

FORTRAN77 subroutine ARP.ZERO

– initialisation routine

```
C     ***********************************************
C     * Associative Reward-Punish Element (ARP)     *
C     * Data initialisation subroutine;             *
C     *                                             *
C     * - This routine initialises the BLOCK data   *
C     *   for the ARP.NET type program;             *
C     * - Routine uses DOUBLE PRECISION variables;  *
C     *                                             *
C     * Author: Richard Leaver                      *
C     * Created:20th May  1987                      *
C     * Update : 6th January 1988                   *
C     * Frozen :20th July 1988                      *
C     ***********************************************
      BLOCK DATA INIT
C
C     Define everything as implicit REAL*8
C
      IMPLICIT REAL*8(A - H,Q - Z)
C
C     Define Common Blocks and initialise using a BLOCK data statement;
C
C     Define Frame - Max X x Y x Z
C
C     FRAME(0:X-1,0:Y-1,N+1)
C
C     Final Frame N+1 is the output;
C
      REAL*8 FRAME(0:4,0:4,5)
C
C     Define X inputs the  ARP elements with Z  Frames
C
      REAL*8 XIN(0:25,4)
C
C     Define Weights for X x Y ARP elements with Z  Frames;
C
      REAL*8 WEIGHT(0:4,0:4,4,0:4,0:4)
C
C     Define Stimulus for X x Y ARP elements with Z  Frames;
C
      REAL*8 STIM(0:4,0:4,4)
C
C     Save SUMs for X x Y ARP elements, Z Frames ;
C
      REAL*8 SUMS(0:4,0:4,4)
C
C     Define Frame shift register;
C
      REAL*8 SHIFT1(0:4,0:4,4,0:20)
C
C     Define Error shift register;
C
      REAL*8 SHIFT2(0:4,0:4,4,0:20)
C
C     Define Sum shift register;
C
      REAL*8 SHIFT3(0:4,0:4,4,0:20)
C
C     Define Weights shift register;
C
      REAL*8 SHIFT4(0:4,0:4,4,0:25)
C
C     Define Reward block;
C
```

```
      INTEGER REWARD(0:4,0:4), GLOREW
C
C     Integers;
C
      INTEGER LENGTH, WIDTH, DEPTH, BAT
C
C     Real *8 variables;
C
      REAL*8 RHO, TEMP, LAMBDA, PN
C
C     Define Common block One;
C
      COMMON /ONE/ FRAME, WEIGHT, SUMS, STIM, LENGTH, WIDTH, DEPTH,
     &       REWARD, GLOREW
C
C     Define Common Block Two;
C
      COMMON /TWO/ RHO, TEMP, LAMBDA, PN
C
C     Define Common Block Three;
C
      COMMON /THREE/ XIN
C
C     Define Common Block Five;
C
      COMMON /FIVE/ SHIFT1, SHIFT2, SHIFT3, SHIFT4, ISHIFT, IBCNT, BAT
C
C     Initialise Frame;
C
      DATA FRAME /125*0.0D0/
C
C     Initialise Xinput;
C
      DATA XIN /104*0.0D0/
C
C     Initialise Weight;
C
      DATA WEIGHT /2500*0.0D0/
C
C     Initialise Stimulus;
C
      DATA STIM /100*0.0D0/
C
C     Initialise Sums;
C
      DATA SUMS /100*0.0D0/
C
C     Initialise Reward block;
C
      DATA REWARD /25*0/
C
C     Initialise Frame shift register;
C
      DATA SHIFT1 /2100*0.0D0/
C
C     Initialise sum shift register;
C
      DATA SHIFT2 /2100*0.0D0/
C
C     Initialise error shift register;
C
```

```
      DATA SHIFT3 /2100*1.0D0/
C
C      Initialise weights shift register;
C
      DATA SHIFT4 /2600*0.0D0/
C
C      Return;
C
      END
```

Simulation programs for Appendix 5 – simple A_{R-P} networks

FORTRAN77 program ARP.1EL

– one A_{R-P} element simulation

```
C     ************************************************
C     * Associative Reward-Punish Element (ARP)      *
C     * Simulation Linking Program.                  *
C     *                                              *
C     *                                              *
C     * - This program simulates one ARP element     *
C     *   and outputs the performance average        *
C     *   Mk over n sequences of up to 10000         *
C     *   trials;                                    *
C     * - Three cases for Lambda are taken -         *
C     *   0.01,0.05,0.25                             *
C     * - The NAG routine G05CBF(1) initialises the  *
C     *   random number generator to                 *
C     *   repeatable sequences.                      *
C     * - Subroutines called;                        *
C     *   ARP in ARP.SUB                             *
C     *   PSI in ARP.FUNC                            *
C     *   *NAG                                       *
C     *                                              *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created: 17th January 1987                   *
C     * Update : 19th March 1987                     *
C     * Frozen : 23rd March 1987                     *
C     ************************************************
      PROGRAM ONE
C
C     Define averaging array for Reward line;
C     (for three cases of Lambda)
C
      IMPLICIT REAL*8(A - H,O - Z)
      REAL*8 SAVE(10000,3), WGT1(10000,3), WGT2(10000,3)
C
C     Define Lambda array;
C
      REAL*8 LAM(3)
C
C     Define all variables as DOUBLE PRECISION;
C
C     Define Real variables element 1;
C
      REAL*8 PN, CASH, RHO, TEMP, MK, LAMBDA, PSI
      REAL*8 W1(10), X1(10), G05CAF
      REAL*8 PK1(2)
C
C     Define Integer variables element 1;
C
      INTEGER Y1, TRIAL, REWARD, YINT1
C
C     Initialise Random Number Generator (repeatable sequences);
C
      CALL G05CBF(1)
      PN = 1.0D0
C
C     Set max time;
C
      MAXTIM = 10000
C
C     Initialise SAVE & Weight arrays;
C
      DO 2 I = 1, MAXTIM
C
```

```
      DO 1 J = 1, 3
        SAVE(I,J) = 0.0D0
        WGT1(I,J) = 0.0D0
        WGT2(I,J) = 0.0D0
1   CONTINUE
C
2 CONTINUE
C
C     Put values in Lambda array;
C
     LAM(1) = 0.01D0
     LAM(2) = 0.05D0
     LAM(3) = 0.25D0
C
C     Set initial parameters  Element 1;
C
C     X11 is set to be a constant input of 1;
C     X12 is set randomly;
C
     X1(1) = 1.0D0
C
     DO 3 K = 3, 10
       X1(K) = 0.0D0
3 CONTINUE
C
C     Read variables;
C
     WRITE (6,*)'Please enter values for Rho,Temp and Maxcnt'
     READ (5,*) RHO, TEMP, MAXCNT
C
C     Do Lambda loop
C
     DO 7 LCOUNT = 1, 3
       LAMBDA = LAM(LCOUNT)
C
C     Do Averaging loop;
C
      DO 6 ICOUNT = 1, MAXCNT
        IF (MOD(ICOUNT,5) .EQ. 0) WRITE (6,*) ICOUNT
C
C     Set initial weights Element 1 to zero;
C
        DO 4 K = 1, 10
          W1(K) = 0.0D0
4       CONTINUE
C
C     Begin loop;
C
        DO 5 TRIAL = 1, MAXTIM
C
C     Generate Random X12 (either 0 or 1 with probability 0.5);
C
          PN = G05CAF(PN)
          IF (PN .LT. (0.5D0)) THEN
            X1(2) = 1.0D0
          ELSE
            X1(2) = 0.0D0
          END IF
C
C     Call ARP element 1;
C
```

```
            CALL ARP(X1, Y1, YINT1, SUM1, REWARD, W1, RHO, TEMP, LAMBDA,
     &           PN, 0)
C
C     Calculate CASH REWARD;
C
            IF ((Y1 .EQ. 1) .AND. (X1(2) .EQ. 0.0D0)) CASH = 0.1D0
            IF ((Y1 .EQ. 0) .AND. (X1(2) .EQ. 0.0D0)) CASH = 0.9D0
            IF ((Y1 .EQ. 1) .AND. (X1(2) .EQ. 1.0D0)) CASH = 0.9D0
            IF ((Y1 .EQ. 0) .AND. (X1(2) .EQ. 1.0D0)) CASH = 0.1D0
C
C     Now encode the REWARD Line (PN between 0 & 1);
C
            PN = G05CAF(PN)
            IF (PN .LT. CASH) THEN
              REWARD = 1
            ELSE
              REWARD = -1
            END IF
C
C     Call ARP element 1;
C
            CALL ARP(X1, Y1, YINT1, SUM1, REWARD, W1, RHO, TEMP, LAMBDA,
     &           PN, 1)
C
C     Calculate performance;
C     Define vectors as follows;
C
C     1 = 1,0
C     2 = 1,1
C
C     Calculate Pk11 - Probability that Y1 is '1' due to X(1,0) ;
C
            PK1(1) = PSI(W1(1),0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.
     &      0D0,0.0D0,0.0D0,TEMP)
C
C     Calculate Pk12 - Probability that Y1 is '1' due to X(1,1) ;
C
            PK1(2) = PSI(W1(1),W1(2),0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.
     &      0D0,0.0D0,0.0D0,TEMP)
C
C     Calculate Mk ;
C
            MK = 0.50D0 * (0.1D0*PK1(1) + (0.9D0*(1.0D0-PK1(1)))) + 0.
     &      50D0 * (0.9D0*PK1(2) + (0.1D0*(1.0D0-PK1(2))))
C
C     Debug Mk check option;
C
C     WRITE(1,*)MK
C
C     Put Average in SAVE array;
C
            SAVE(TRIAL,LCOUNT) = MK + SAVE(TRIAL,LCOUNT)
C
C     Put weights in WGT arrays;
C
            WGT1(TRIAL,LCOUNT) = W1(1) + WGT1(TRIAL,LCOUNT)
            WGT2(TRIAL,LCOUNT) = W1(2) + WGT2(TRIAL,LCOUNT)
C
C     End Trial loop;
C
```

```
    5     CONTINUE
C
C     End MAXCNT loop;
C
    6    CONTINUE
C
C     End Lambda loop;
C
    7 CONTINUE
C
C     Write out Mk for all cases;
C
      DO 8 JCOUNT = 1, MAXTIM
        WRITE (2,*) JCOUNT, SAVE(JCOUNT,1) / DFLOAT(MAXCNT),
     &         SAVE(JCOUNT,2) / DFLOAT(MAXCNT), SAVE(JCOUNT,3) / DFLOAT(
     &         MAXCNT)
    8 CONTINUE
C
C     Write out Weight 1 averages for all cases;
C
      DO 9 JCOUNT = 1, MAXTIM
        WRITE (3,*) JCOUNT, WGT1(JCOUNT,1) / DFLOAT(MAXCNT),
     &         WGT1(JCOUNT,2) / DFLOAT(MAXCNT), WGT1(JCOUNT,3) / DFLOAT(
     &         MAXCNT)
    9 CONTINUE
C
C     Write out Weight 2 averages for all cases;
C
      DO 10 JCOUNT = 1, MAXTIM
        WRITE (4,*) JCOUNT, WGT2(JCOUNT,1) / DFLOAT(MAXCNT),
     &         WGT2(JCOUNT,2) / DFLOAT(MAXCNT), WGT2(JCOUNT,3) / DFLOAT(
     &         MAXCNT)
   10 CONTINUE
C
      STOP
      END
```

FORTRAN77 subroutine ARP.SUB

– A_{R-P} element

```
C     ***********************************************
C     * Associative Reward-Punish Element (ARP)     *
C     * Simulation Subroutine.                      *
C     *                                             *
C     * - This subroutine simulates one ARP element *
C     *   with 10 possible X inputs;                *
C     * - The inputs are -                          *
C     *   X(1,10) and REWARD ;                      *
C     * - LAMBDA (Real*8) - Learning Rate ;         *
C     * - ACTION - This defines what the subroutine *
C     *   does on entry. ;                          *
C     * - If ACTION=0 then the element calculates   *
C     *   the summation, and encodes this using the *
C     *   Psi distribution;                         *
C     * - If ACTION=1 then the element calculates   *
C     *   weights according to the reward input;    *
C     * - The output is -                           *
C     *   Y (integer +1/0)  - This is expressed     *
C     *   internally for the element as (+1/-1)     *
C     *   in order to allow weight updates;         *
C     * - The weights for each element, W1 etc      *
C     *   are passed through to the main program    *
C     *   in order to preserve their values;        *
C     * - The NAG routine G05CAF is used to provide *
C     *   uniform random numbers between 0 and 1.   *
C     * - The NAG routine G05CCF must be called in  *
C     *   the Main program to ensure non-repeatable *
C     *   sequences.                                *
C     * - A non uniform distribution is generated   *
C     *   by using the Inversion method.            *
C     * - Program uses DOUBLE PRECISION variables;  *
C     * - Subroutines called;                       *
C     *   *NAG                                      *
C     *                                             *
C     * Author: Richard Leaver                      *
C     * Created: 17th January 1987                  *
C     * Update : 19th March   1987                  *
C     * Frozen : 23rd March   1987                  *
C     ***********************************************
      SUBROUTINE ARP(X, Y, YINT, SUM, REWARD, W, RHO, TEMP, LAMBDA, PN,
     &           ACTION)
C
C     Define all variables as DOUBLE PRECISION;
C
      REAL*8 W(10), INC(10), X(10)
      REAL*8 PN, TEMP, RHO, G05CAF
      REAL*8 THRESH, SUM, E, LAMBDA
C
C     Define Integer variables;
C
      INTEGER Y, YINT, REWARD, ACTION
C
C     If ACTION is 0 then do summation and encoding;
C
      IF (ACTION .EQ. 0) THEN
C
C     Calculate Summation;
C
       SUM = 0.0D0
C
       DO 1 ICOUNT = 1, 10
         SUM = SUM + X(ICOUNT) * W(ICOUNT)
```

```
   1    CONTINUE
C
C      Encode this summation with a special distribution;
C      Psi=(1+exp(-s/t))^-1
C
C      This is implemented using a equiprobable random number u;
C      using the Inversion Method;
C      s=-t.ln[1/u -1]
C
        PN = G05CAF(PN)
        THRESH = -1.0D0 * TEMP * DLOG(1.0D0/PN - 1.0D0)
        IF (SUM .GT. (-1.0D0*THRESH)) THEN
          Y = 1
          YINT = 1
        ELSE
          Y = 0
          YINT = -1
        END IF
      ELSE
C
C      If ACTION is 1 then calculate the weight update;
C
C
C      Calculate Distribution Function E;
C
        E = (DEXP(SUM/TEMP) - 1.0D0) / (DEXP(SUM/TEMP) + 1.0D0)
C
C      Amend weights - change in weight = INC1,INC2 ;
C
        DO 2 ICOUNT = 1, 10
          IF (REWARD .EQ. 1) THEN
            INC(ICOUNT) = RHO * (REWARD*DFLOAT(YINT) - E) * X(ICOUNT)
          ELSE
            INC(ICOUNT) = LAMBDA * RHO * (REWARD*DFLOAT(YINT) - E) * X(
     &      ICOUNT)
          END IF
          W(ICOUNT) = W(ICOUNT) + INC(ICOUNT)
   2    CONTINUE
C
C      End ACTION IF ;
C
      END IF
      RETURN
      END
```

FORTRAN77 function ARP.FUNC

– A_{R-P} I/O characteristic

```
C     ***********************************************
C     * Associative Reward-Punish Element (ARP)     *
C     * Simulation Function Routine;                *
C     *                                             *
C     * - This function computes the Psi probability *
C     *   that an element will fire with 10 possible *
C     *   Weight inputs and one Temperature input;   *
C     * - The inputs are -                          *
C     *   WT(1,10) and TEMP (all Real*8)            *
C     * - Unused WT inputs must be set to 0.0D0 ;    *
C     * - Program uses DOUBLE PRECISION variables;  *
C     *                                             *
C     * Author: Richard Leaver                      *
C     * Created:  6th March 1987                    *
C     * Update : 19th March 1987                    *
C     * Frozen : 23rd March 1987                    *
C     ***********************************************
      DOUBLE PRECISION FUNCTION PSI(WT1,WT2,WT3,WT4,WT5,WT6,WT7,WT8,WT9,
     &       WT10,TEMP)
C
C     Define all Real*8 variables;
C
      REAL*8 WT1, WT2, WT3, WT4, WT5, WT6, WT7, WT8, WT9, WT10, TEMP

C
C     Compute function;
C
      IF ((WT1 + WT2 + WT3 + WT4 + WT5 + WT6 + WT7 + WT8 + WT9 + WT10)/
     &       TEMP .GT. 10.0D0) THEN
        PSI = 1.0D0
      ELSE
        IF ((WT1 + WT2 + WT3 + WT4 + WT5 + WT6 + WT7 + WT8 + WT9 + WT10)
     &         /TEMP .LT. - 10.0D0) THEN
          PSI = 0.0D0
        ELSE
          PSI = 1.0D0 / (1.0D0+DEXP(-(WT1 + WT2 + WT3 + WT4 + WT5 + WT6
     &    + WT7 + WT8 + WT9 + WT10)/TEMP))
        END IF
      END IF
C
C     Return to calling program;
C
      RETURN
      END
```

FORTRAN77 program ARP.2EL

– two A_{R-P} element simulation

```
C     ************************************************
C     * Associative Reward-Punish Element (ARP)      *
C     * Simulation Linking Program.                  *
C     *                                              *
C     *                                              *
C     * - This program simulates two ARP elements    *
C     *   connected as in the second example of the  *
C     *   Learning by statistical cooperation paper  *
C     *   and outputs the performance average        *
C     *   Mk over n sequences of up to 10000         *
C     *   trials;                                    *
C     * - Both element weights are also output;      *
C     * - Three cases for Lambda are taken -         *
C     *   0.01,0.05,0.25                             *
C     * - The NAG routine G05CCF initialises the     *
C     *   random number generator to non             *
C     *   repeatable sequences.                      *
C     * - Subroutines called;                        *
C     *   ARP in ARP.SUB                             *
C     *   PSI in ARP.FUNC                            *
C     *   *NAG                                       *
C     *                                              *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created: 17th January 1987                   *
C     * Update : 17th June 1987                      *
C     * Frozen :  6th June 1987                      *
C     ************************************************
      PROGRAM TWO
C
C     Define averaging array for Reward line;
C     (for three cases of Lambda)
C
      IMPLICIT REAL*8(A - H,O - Z)
      REAL*8 SAVE(10000,3), WGT1(10000,3), WGT2(10000,3)
      REAL*8 WGT3(10000,3), WGT4(10000,3)
C
C     Define Lambda array;
C
      REAL*8 LAM(3)
C
C     Define all variables as DOUBLE PRECISION;
C
C     Define Real variables element 1;
C
      REAL*8 PN, CASH, RHO, TEMP, MK, LAMBDA, PSI
      REAL*8 W1(10), X1(10), G05CAF
      REAL*8 PK1(2)
C
C     Define Real variables element 2;
C
      REAL*8 W2(10), X2(10)
      REAL*8 PK2(2)
C
C     Define Integer variables element 1;
C
      INTEGER Y1, TRIAL, REWARD, YINT1
C
C     Define Integer variables element 2;
C
      INTEGER Y2, YINT2
C
C     Initialise Random Number Generator (non-repeatable sequences);
C
```

```
      CALL G05CCF
      PN = 1.0D0
C
C      Set max time;
C
      MAXTIM = 10000
C
C      Initialise SAVE & Weight arrays;
C
      DO 2 I = 1, MAXTIM
C
        DO 1 J = 1, 3
          SAVE(I,J) = 0.0D0
          WGT1(I,J) = 0.0D0
          WGT2(I,J) = 0.0D0
          WGT3(I,J) = 0.0D0
          WGT4(I,J) = 0.0D0
    1   CONTINUE
C
    2 CONTINUE
C
C      Put values in Lambda array;
C
      LAM(1) = 0.01D0
      LAM(2) = 0.05D0
      LAM(3) = 0.25D0
C
C      Set initial parameters  Element 1 & 2;
C
C      X11 and X21 are set to have a constant input of 1;
C      X12 is set randomly;
C      X22 is Y1;
C
      X1(1) = 1.0D0
      X2(1) = 1.0D0
C
      DO 3 K = 3, 10
        X1(K) = 0.0D0
        X2(K) = 0.0D0
    3 CONTINUE
C
C      Read variables;
C
      WRITE (6,*)'Please enter values for Rho,Temp and Maxcnt'
      READ (5,*) RHO, TEMP, MAXCNT
C
C      Do Lambda loop
C
      DO 7 LCOUNT = 1, 3
        LAMBDA = LAM(LCOUNT)
C
C      Do Averaging loop;
C
        DO 6 ICOUNT = 1, MAXCNT
C
C      Set initial weights Element 1 & 2 to zero;
C
          DO 4 K = 1, 10
            W1(K) = 0.0D0
            W2(K) = 0.0D0
```

```
   4     CONTINUE
C
C     Begin loop;
C
         DO 5 TRIAL = 1, MAXTIM
C
C     Generate Random X12 (either 0 or 1 with probability 0.5);
C
            PN = G05CAF(PN)
            IF (PN .LT. (0.5D0)) THEN
              X1(2) = 1.0D0
            ELSE
              X1(2) = 0.0D0
            END IF
C
C     Call ARP element 1;
C
            CALL ARP(X1, Y1, YINT1, SUM1, REWARD, W1, RHO, TEMP, LAMBDA,
     &           PN, 0)
C
C     Connect elements;
C
            X2(2) = DFLOAT(Y1)
C
C     Call ARP element 2;
C
            CALL ARP(X2, Y2, YINT2, SUM2, REWARD, W2, RHO, TEMP, LAMBDA,
     &           PN, 0)
C
C     Calculate CASH REWARD;
C
            IF ((Y2 .EQ. 1) .AND. (X1(2) .EQ. 1.0D0)) CASH = 0.9D0
            IF ((Y2 .EQ. 0) .AND. (X1(2) .EQ. 1.0D0)) CASH = 0.1D0
            IF ((Y2 .EQ. 1) .AND. (X1(2) .EQ. 0.0D0)) CASH = 0.1D0
            IF ((Y2 .EQ. 0) .AND. (X1(2) .EQ. 0.0D0)) CASH = 0.9D0
C
C     Now encode the REWARD Line (PN between 0 & 1);
C
            PN = G05CAF(PN)
            IF (PN .LT. CASH) THEN
              REWARD = 1
            ELSE
              REWARD = -1
            END IF
C
C     Call ARP element 1;
C
            CALL ARP(X1, Y1, YINT1, SUM1, REWARD, W1, RHO, TEMP, LAMBDA,
     &           PN, 1)
C
C     Call ARP element 2;
C
            CALL ARP(X2, Y2, YINT2, SUM2, REWARD, W2, RHO, TEMP, LAMBDA,
     &           PN, 1)
C
C     Calculate performance;
C     Define vectors as follows;
C
C     1 = 1,0
C     2 = 1,1
C
C     Calculate Pk11 - Probability that Y1 is '1' due to X(1,0) ;
C
```

```
            PK1(1) = PSI(W1(1),0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.
     &      0D0,0.0D0,0.0D0,TEMP)
C
C     Calculate Pk12 - Probability that Y1 is '1' due to X(1,1) ;
C
            PK1(2) = PSI(W1(1),W1(2),0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.
     &      0D0,0.0D0,0.0D0,TEMP)
C
C     Calculate Pk21 - Probability that Y2 is '1' due to X(1,0) ;
C
            PK2(1) = PK1(1) * PSI(W2(1),W2(2),0.0D0,0.0D0,0.0D0,0.0D0,0.
     &      0D0,0.0D0,0.0D0,0.0D0,TEMP) + (1.0D0-PK1(1)) * PSI(W2(1),0.
     &      0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,TEMP)
C
C     Calculate Pk22 - Probability that Y2 is '1' due to X(1,1) ;
C
            PK2(2) = PK1(2) * PSI(W2(1),W2(2),0.0D0,0.0D0,0.0D0,0.0D0,0.
     &      0D0,0.0D0,0.0D0,0.0D0,TEMP) + (1.0D0-PK1(2)) * PSI(W2(1),0.
     &      0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,TEMP)
C
C     Calculate Mk ;
C
            MK = 0.50D0 * (0.1D0*PK2(1) + (0.9*(1.0D0-PK2(1)))) + 0.
     &      50D0 * (0.9D0*PK2(2) + (0.1*(1.0D0-PK2(2))))
C
C     Debug Mk check option;
C
C     WRITE(1,*)MK
C
C     Put Average in SAVE array;
C
            SAVE(TRIAL,LCOUNT) = MK + SAVE(TRIAL,LCOUNT)
C
C     Put weights in WGT arrays;
C
            WGT1(TRIAL,LCOUNT) = W1(1) + WGT1(TRIAL,LCOUNT)
            WGT2(TRIAL,LCOUNT) = W1(2) + WGT2(TRIAL,LCOUNT)
            WGT3(TRIAL,LCOUNT) = W2(1) + WGT3(TRIAL,LCOUNT)
            WGT4(TRIAL,LCOUNT) = W2(2) + WGT4(TRIAL,LCOUNT)
C
C     End Trial loop;
C
   5     CONTINUE
C
C     End MAXCNT loop;
C
   6   CONTINUE
C
C     End Lambda loop;
C
   7 CONTINUE
C
C     Write out Mk for all cases;
C
      DO 8 JCOUNT = 1, MAXTIM
        WRITE (2,*) JCOUNT, SAVE(JCOUNT,1) / DFLOAT(MAXCNT),
     &        SAVE(JCOUNT,2) / DFLOAT(MAXCNT), SAVE(JCOUNT,3) / DFLOAT(
     &        MAXCNT)
   8 CONTINUE
C
C     Write out Weight 1 averages for all cases;
C
```

```
      DO 9 JCOUNT = 1, MAXTIM
        WRITE (3,*) JCOUNT, WGT1(JCOUNT,1) / DFLOAT(MAXCNT),
     &         WGT1(JCOUNT,2) / DFLOAT(MAXCNT), WGT1(JCOUNT,3) / DFLOAT(
     &         MAXCNT)
    9 CONTINUE
C
C     Write out Weight 2 averages for all cases;
C
      DO 10 JCOUNT = 1, MAXTIM
        WRITE (4,*) JCOUNT, WGT2(JCOUNT,1) / DFLOAT(MAXCNT),
     &         WGT2(JCOUNT,2) / DFLOAT(MAXCNT), WGT2(JCOUNT,3) / DFLOAT(
     &         MAXCNT)
   10 CONTINUE
C
C     Write out Weight 3 averages for all cases;
C
      DO 11 JCOUNT = 1, MAXTIM
        WRITE (13,*) JCOUNT, WGT3(JCOUNT,1) / DFLOAT(MAXCNT),
     &         WGT3(JCOUNT,2) / DFLOAT(MAXCNT), WGT3(JCOUNT,3) / DFLOAT(
     &         MAXCNT)
   11 CONTINUE
C
C     Write out Weight 4 averages for all cases;
C
      DO 12 JCOUNT = 1, MAXTIM
        WRITE (14,*) JCOUNT, WGT4(JCOUNT,1) / DFLOAT(MAXCNT),
     &         WGT4(JCOUNT,2) / DFLOAT(MAXCNT), WGT4(JCOUNT,3) / DFLOAT(
     &         MAXCNT)
   12 CONTINUE
C
      STOP
      END
```

FORTRAN77 program ARP.EXOR

– Exclusive OR simulation

```
C     ************************************************
C     * Associative Reward-Punish Element (ARP)      *
C     * Simulation Linking Program.                  *
C     *                                              *
C     *                                              *
C     * - This program simulates two ARP elements    *
C     *   connected in parallel and outputs the      *
C     *   performance average for the EXCLUSIVE OR,  *
C     *   Mk over n sequences of up to 10000         *
C     *   trials;                                    *
C     * - Three cases for Lambda can be selected;    *
C     *   0.08,0.10,0.25                             *
C     * - The NAG routine G05CCF initialises the     *
C     *   random number generator to non             *
C     *   repeatable sequences.                      *
C     * - This case is for the third example in the  *
C     *   Learning by statistical cooperation paper. *
C     * - Subroutines called;                        *
C     *   ARP in ARP.SUB                             *
C     *   PSI in ARP.FUNC                            *
C     *   *NAG                                       *
C     *                                              *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created: 17th January 1987                   *
C     * Update :  6th May 1987                       *
C     * Frozen :  7th April 1988                     *
C     ************************************************
      PROGRAM EXOR
C
C     Define averaging arrays for Mk and Weights;
C     (for three cases of Lambda)
C
      IMPLICIT REAL*8(A - H,O - Z)
      REAL*8 SAVE(10000,3)
      REAL*8 WGT11(10000,3), WGT12(10000,3), WGT13(10000,3)
      REAL*8 WGT21(10000,3), WGT22(10000,3), WGT23(10000,3)
      REAL*8 WGT24(10000,3)
C
C     Define Lambda array;
C
      REAL*8 LAM(3)
C
C     Define all variables as DOUBLE PRECISION;
C
C     Define Real variables element 1;
C
      REAL*8 PN, CASH, RHO, TEMP, MK, LAMBDA, PSI
      REAL*8 W1(10), X1(10), G05CAF
      REAL*8 PK1(4)
C
C     Define Real variables element 2;
C
      REAL*8 W2(10), X2(10)
      REAL*8 PK2(4)
C
C     Define Integer variables element 1;
C
      INTEGER Y1, TRIAL, REWARD, YINT1
C
C     Define Integer variables element 2;
C
```

```
      INTEGER Y2, YINT2
C
C      Initialise Random Number Generator (non-repeatable sequences);
C
      CALL G05CCF
      PN = 1.0D0
C
C      Set max time;
C
      MAXTIM = 10000
C
C      Initialise SAVE and Weight arrays;
C
      DO 2 I = 1, MAXTIM
C
        DO 1 J = 1, 3
          SAVE(I,J) = 0.0D0
          WGT11(I,J) = 0.0D0
          WGT12(I,J) = 0.0D0
          WGT13(I,J) = 0.0D0
          WGT21(I,J) = 0.0D0
          WGT22(I,J) = 0.0D0
          WGT23(I,J) = 0.0D0
          WGT24(I,J) = 0.0D0
    1   CONTINUE
C
    2 CONTINUE
C
C      Put values in Lambda array;
C
      LAM(1) = 0.08D0
      LAM(2) = 0.10D0
      LAM(3) = 0.25D0
C
C      Set initial parameters  Element 1;
C
C      X11 and X12 are set randomly;
C      X13 is set to be a constant input of 1;
C
      X1(3) = 1.0D0
C
      DO 3 K = 4, 10
        X1(K) = 0.0D0
    3 CONTINUE
C
C      Set initial parameters  Element 2;
C
C      X21 is the same as X11;
C      X22 is set from the element 1 output Y1;
C      X23 is the same as X12;
C      X24 is set to be a constant input of 1;
C
      X2(4) = 1.0D0
C
      DO 4 K = 5, 10
        X2(K) = 0.0D0
    4 CONTINUE
C
C      Read variables;
C
```

```
      WRITE (6,*)'Please enter values for Rho,Temp and Maxcnt'
      READ (5,*) RHO, TEMP, MAXCNT
C
C     Do Lambda loop  (select Lambda)
C
      DO 9 LCOUNT = 1, 1
        LAMBDA = LAM(LCOUNT)
C
C     Do Averaging loop;
C
        DO 8 ICOUNT = 1, MAXCNT
C
C     Set initial weights Element 1 to zero;
C
          DO 5 K = 1, 10
            W1(K) = 0.0D0
    5     CONTINUE
C
C     Set initial weights Element 2 to zero;
C
          DO 6 K = 1, 10
            W2(K) = 0.0D0
    6     CONTINUE
C
C     Begin loop;
C
          DO 7 TRIAL = 1, MAXTIM
C
C     Generate Random X11 (either 0 or 1 with probability 0.5);
C
            PN = G05CAF(PN)
            IF (PN .LT. (0.5D0)) THEN
              X1(1) = 1.0D0
            ELSE
              X1(1) = 0.0D0
            END IF
C
C     Generate Random X12 (either 0 or 1 with probability 0.5);
C
            PN = G05CAF(PN)
            IF (PN .LT. (0.5D0)) THEN
              X1(2) = 1.0D0
            ELSE
              X1(2) = 0.0D0
            END IF
C
C     Call ARP element 1;
C
            CALL ARP(X1, Y1, YINT1, SUM1, REWARD, W1, RHO, TEMP, LAMBDA,
     &           PN, 0)
C
C     Connect the two elements;
C
            X2(1) = X1(1)
            X2(2) = DFLOAT(Y1)
            X2(3) = X1(2)
C
C     Call ARP element 2;
C
            CALL ARP(X2, Y2, YINT2, SUM2, REWARD, W2, RHO, TEMP, LAMBDA,
```

```
     &           PN, 0)
C
C      Calculate CASH REWARD;
C
            IF (X1(1) .EQ. 1.0D0) THEN
              IF ((Y2 .EQ. 1) .AND. (X1(2) .EQ. 1.0D0)) CASH = 0.1D0
              IF ((Y2 .EQ. 0) .AND. (X1(2) .EQ. 1.0D0)) CASH = 0.9D0
              IF ((Y2 .EQ. 1) .AND. (X1(2) .EQ. 0.0D0)) CASH = 0.9D0
              IF ((Y2 .EQ. 0) .AND. (X1(2) .EQ. 0.0D0)) CASH = 0.1D0
            ELSE
              IF ((Y2 .EQ. 1) .AND. (X1(2) .EQ. 1.0D0)) CASH = 0.9D0
              IF ((Y2 .EQ. 0) .AND. (X1(2) .EQ. 1.0D0)) CASH = 0.1D0
              IF ((Y2 .EQ. 1) .AND. (X1(2) .EQ. 0.0D0)) CASH = 0.1D0
              IF ((Y2 .EQ. 0) .AND. (X1(2) .EQ. 0.0D0)) CASH = 0.9D0
            END IF
C
C      Now encode the REWARD Line (PN between 0 & 1);
C
            PN = G05CAF(PN)
            IF (PN .LT. CASH) THEN
              REWARD = 1
            ELSE
              REWARD = -1
            END IF
C
C      Call ARP element 1;
C
            CALL ARP(X1, Y1, YINT1, SUM1, REWARD, W1, RHO, TEMP, LAMBDA,
     &           PN, 1)
C
C      Call ARP element 2;
C
            CALL ARP(X2, Y2, YINT2, SUM2, REWARD, W2, RHO, TEMP, LAMBDA,
     &           PN, 1)
C
C      Calculate performance;
C      Define vectors as follows;
C
C      1 = 0,0
C      2 = 0,1
C      3 = 1,0
C      4 = 1,1
C
C      Calculate Pk11 - Probability that X22 is '1' due to X(0,0) ;
C
            PK1(1) = PSI(W1(3),0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.
     &      0D0,0.0D0,0.0D0,TEMP)
C
C      Calculate Pk12 - Probability that X22 is '1' due to X(0,1) ;
C
            PK1(2) = PSI(W1(2),W1(3),0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.
     &      0D0,0.0D0,0.0D0,TEMP)
C
C      Calculate Pk13 - Probability that X22 is '1' due to X(1,0) ;
C
            PK1(3) = PSI(W1(1),W1(3),0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.
     &      0D0,0.0D0,0.0D0,TEMP)
C
C      Calculate Pk14 - Probability that X22 is '1' due to X(1,1) ;
C
```

```
            PK1(4) = PSI(W1(1),W1(2),W1(3),0.0D0,0.0D0,0.0D0,0.0D0,0.
     &      0D0,0.0D0,0.0D0,TEMP)
C
C     Calculate Pk21 - Probability that Y2 is '1' due to X(0,0) ;
C
            PK2(1) = PK1(1) * PSI(W2(2),W2(4),0.0D0,0.0D0,0.0D0,0.0D0,0.
     &      0D0,0.0D0,0.0D0,0.0D0,TEMP) + (1.0D0-PK1(1)) * PSI(W2(4),0.
     &      0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,TEMP)
C
C     Calculate Pk22 - Probability that Y2 is '1' due to X(0,1) ;
C
            PK2(2) = PK1(2) * PSI(W2(2),W2(3),W2(4),0.0D0,0.0D0,0.0D0,0.
     &      0D0,0.0D0,0.0D0,0.0D0,TEMP) + (1.0D0-PK1(2)) * PSI(W2(3),W2(
     &      4),0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,TEMP)
C
C     Calculate Pk23 - Probability that Y2 is '1' due to X(1,0) ;
C
            PK2(3) = PK1(3) * PSI(W2(1),W2(2),W2(4),0.0D0,0.0D0,0.0D0,0.
     &      0D0,0.0D0,0.0D0,0.0D0,TEMP) + (1.0D0-PK1(3)) * PSI(W2(1),W2(
     &      4),0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,TEMP)
C
C     Calculate Pk24 - Probability that Y2 is '1' due to X(1,1) ;
C
            PK2(4) = PK1(4) * PSI(W2(1),W2(2),W2(3),W2(4),0.0D0,0.0D0,0.
     &      0D0,0.0D0,0.0D0,0.0D0,TEMP) + (1.0D0-PK1(4)) * PSI(W2(1),W2(
     &      3),W2(4),0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,TEMP)
C
C     Calculate Mk ;
C
            MK = 0.25D0 * (0.1D0*PK2(1) + (0.9*(1.0D0-PK2(1)))) + 0.
     &      25D0 * (0.9D0*PK2(2) + (0.1*(1.0D0-PK2(2)))) + 0.25D0 * (0.
     &      9D0*PK2(3) + (0.1*(1.0D0-PK2(3)))) + 0.25D0 * (0.1D0*PK2(4)
     &      + (0.9*(1.0D0-PK2(4))))
C
C     Put Mk in SAVE array;
C
            SAVE(TRIAL,LCOUNT) = MK + SAVE(TRIAL,LCOUNT)
C
C     Put Weights in WGT arrays;
C
            WGT11(TRIAL,LCOUNT) = W1(1) + WGT11(TRIAL,LCOUNT)
            WGT12(TRIAL,LCOUNT) = W1(2) + WGT12(TRIAL,LCOUNT)
            WGT13(TRIAL,LCOUNT) = W1(3) + WGT13(TRIAL,LCOUNT)
            WGT21(TRIAL,LCOUNT) = W2(1) + WGT21(TRIAL,LCOUNT)
            WGT22(TRIAL,LCOUNT) = W2(2) + WGT22(TRIAL,LCOUNT)
            WGT23(TRIAL,LCOUNT) = W2(3) + WGT23(TRIAL,LCOUNT)
            WGT24(TRIAL,LCOUNT) = W2(4) + WGT24(TRIAL,LCOUNT)
C
C     End Trial loop;
C
   7      CONTINUE
C
C     End MAXCNT loop;
C
   8    CONTINUE
C
C     End Lambda loop;
C
   9 CONTINUE
C
C     Write out Mk for all cases;
C
```

```
      DO 10 JCOUNT = 1, MAXTIM
        WRITE (2,*) JCOUNT, SAVE(JCOUNT,1) / DFLOAT(MAXCNT),
     &         SAVE(JCOUNT,2) / DFLOAT(MAXCNT), SAVE(JCOUNT,3) / DFLOAT(
     &         MAXCNT)
   10 CONTINUE
C
C     Write out Weight 11 averages for all cases;
C
      DO 11 JCOUNT = 1, MAXTIM
        WRITE (3,*) JCOUNT, WGT11(JCOUNT,1) / DFLOAT(MAXCNT), -100.,
     &         -100.
   11 CONTINUE
C
C     Write out Weight 12 averages for all cases;
C
      DO 12 JCOUNT = 1, MAXTIM
        WRITE (4,*) JCOUNT, WGT12(JCOUNT,1) / DFLOAT(MAXCNT), -100.,
     &         -100.
   12 CONTINUE
C
C     Write out Weight 13 averages for all cases;
C
      DO 13 JCOUNT = 1, MAXTIM
        WRITE (7,*) JCOUNT, WGT13(JCOUNT,1) / DFLOAT(MAXCNT), -100.,
     &         -100.
   13 CONTINUE
C
C     Write out Weight 21 averages for all cases;
C
      DO 14 JCOUNT = 1, MAXTIM
        WRITE (8,*) JCOUNT, WGT21(JCOUNT,1) / DFLOAT(MAXCNT), -100.,
     &         -100.
   14 CONTINUE
C
C     Write out Weight 22 averages for all cases;
C
      DO 15 JCOUNT = 1, MAXTIM
        WRITE (10,*) JCOUNT, WGT22(JCOUNT,1) / DFLOAT(MAXCNT), -100.,
     &         -100.
   15 CONTINUE
C
C     Write out Weight 23 averages for all cases;
C
      DO 16 JCOUNT = 1, MAXTIM
        WRITE (11,*) JCOUNT, WGT23(JCOUNT,1) / DFLOAT(MAXCNT), -100.,
     &         -100.
   16 CONTINUE
C
C     Write out Weight 24 averages for all cases;
C
      DO 17 JCOUNT = 1, MAXTIM
        WRITE (12,*) JCOUNT, WGT24(JCOUNT,1) / DFLOAT(MAXCNT), -100.,
     &         -100.
   17 CONTINUE
C
C     Write out a dummy channel for plot convenience;
C
      DO 18 JCOUNT = 1, MAXTIM
        WRITE (90,*) JCOUNT, -100.0D0, -100.0D0, -100.0D0
   18 CONTINUE
C
```

```
STOP
END
```

Simulation programs for Appendix 6

– Asymptotic value prediction for the A_{R-P} simulations

```
C     ************************************************
C     * Associative Reward-Punish Element (ARP)      *
C     * Asymptote Calculation program;               *
C     *                                              *
C     * - This program calculates the theoretical    *
C     *   ARP Mk and Weight asymptotes.              *
C     * - The program requests the appropriate       *
C     *   input data.                                *
C     * - Note that (1,1) is the first pattern       *
C     *   and that (1,0) is the second pattern       *
C     * - Program uses DOUBLE PRECISION variables.   *
C     * - Program uses file of form ASYM.DATA        *
C     *                                              *
C     * Author: Richard Leaver                       *
C     * Created: 20th March 1987                     *
C     * Update : 20th March 1987                     *
C     * Frozen : 23rd March 1987                     *
C     ************************************************
      PROGRAM ASYMP
      IMPLICIT REAL*8(A - H,O - Z)
C
C     Define Real variables (Double Precision) ;
C
      REAL*8 REW1, REW2, LAMBDA, TEMP
      REAL*8 PK1, PK2
      REAL*8 PUN1, PUN2
      REAL*8 PK1A, PK2A
      REAL*8 PK1B, PK2B
      REAL*8 WGT1, WGT2
      REAL*8 MK
      REAL*8 THETA1, THETA2
C
C     Quadratic variables;
C
      REAL*8 A1, B1, C1
      REAL*8 A2, B2, C2
C
C     Read in variables;
C
      READ (1,*) REW1, PUN1, REW2, PUN2, LAMBDA, TEMP
C
C     Compute A1;
C
      A1 = (REW1 - PUN1) * (1.0D0-LAMBDA)
C
C     Compute B1;
C
      B1 = 2.0D0 * LAMBDA * (1.0D0-PUN1) + (PUN1 - REW1)
C
C     Compute C1;
C
      C1 = LAMBDA * (PUN1 - 1.0D0)
C
C     Solve quadratic in PK1;
C
      PK1A = (-B1 + DSQRT(B1**2 - (4.0D0*A1*C1))) / (2.0D0*A1)
      PK1B = (-B1 - DSQRT(B1**2 - (4.0D0*A1*C1))) / (2.0D0*A1)
      WRITE (6,*) PK1A, PK1B
C
C     If either of these is outside limits, print it & take the
C     one inside the limits (0<P<1) as PK1;
      IF ((PK1A .LT. 0.0D0) .OR. (PK1A .GT. 1.0D0)) THEN
```

FORTRAN77 program ARP.ASYM

– for automatic computation of asymptotes

```
        CONTINUE
      ELSE
        PK1 = PK1A
      END IF
      IF ((PK1B .LT. 0.0D0) .OR. (PK1B .GT. 1.0D0)) THEN
        CONTINUE
      ELSE
        PK1 = PK1B
      END IF
C
C
C     Compute A2;
C
      A2 = (REW2 - PUN2) * (1.0D0-LAMBDA)
C
C     Compute B2;
C
      B2 = 2.0D0 * LAMBDA * (1.0D0-PUN2) + (PUN2 - REW2)
C
C     Compute C2;
C
      C2 = LAMBDA * (PUN2 - 1.0D0)
C
C     Solve quadratic in PK2;
C
      PK2A = (-B2 + DSQRT(B2**2 - (4.0D0*A2*C2))) / (2.0D0*A2)
      PK2B = (-B2 - DSQRT(B2**2 - (4.0D0*A2*C2))) / (2.0D0*A2)
      WRITE (6,*) PK2A, PK2B
C
C     If either of these is outside limits, print it & take the
C     one inside the limits (0<P<1) as PK2;
      IF ((PK2A .LT. 0.0D0) .OR. (PK2A .GT. 1.0D0)) THEN
        CONTINUE
      ELSE
        PK2 = PK2A
      END IF
      IF ((PK2B .LT. 0.0D0) .OR. (PK2B .GT. 1.0D0)) THEN
        CONTINUE
      ELSE
        PK2 = PK2B
      END IF
C
C     Compute Mk;
C
      MK = 0.5D0 * PK1 * REW1 + 0.5D0 * (1.0D0-PK1) * PUN1 + 0.5D0 *
     &PK2 * REW2 + 0.5D0 * (1.0D0-PK2) * PUN2
      WRITE (6,*)'MK=', MK
C
C     Compute Theta1;
C
      THETA1 = TEMP * DLOG(1.0D0/(-1.0D0*(PK1 - 1.0D0)) - 1.0D0)
C
C     Compute Theta2;
C
      THETA2 = TEMP * DLOG(1.0D0/(-1.0D0*(PK2 - 1.0D0)) - 1.0D0)
C
C     Compute Weight1 - (function of PK2 only since X2 =  (1,0) );
C
      WGT1 = THETA2
      WRITE (6,*)'WGT1=', WGT1
C
C     Compute Weight2 - (function of PK1 & PK2 since X1 =  (1,1) );
C
```

```
WGT2 = THETA1 - WGT1
WRITE (6,*)'WGT2=', WGT2
STOP
END
```

ASYM.DATA

– Input data file for asymptote computation program

.9,

.1,

.1,

.9,

.25,

.5