Application of Fuzzy Logic-Neural Network Based Reinforcement Learning to Proximity and Docking Operations

Yashvant Jani
Togai InfraLogic, Inc.

5/4/92

Cooperative Agreement NCC 9-16
Research Activity No. AR.06:
Reinforcement Learning Based on Fuzzy Logic and Neural Networks

NASA Johnson Space Center
Information Systems Directorate
Information Technology Division

Research Institute for Computing and Information Systems
University of Houston-Clear Lake
The RICIS Concept

The University of Houston-Clear Lake established the Research Institute for Computing and Information Systems (RICIS) in 1986 to encourage the NASA Johnson Space Center (JSC) and local industry to actively support research in the computing and information sciences. As part of this endeavor, UHCL proposed a partnership with JSC to jointly define and manage an integrated program of research in advanced data processing technology needed for JSC's main missions, including administrative, engineering and science responsibilities. JSC agreed and entered into a continuing cooperative agreement with UHCL beginning in May 1986, to jointly plan and execute such research through RICIS. Additionally, under Cooperative Agreement NCC 9-16, computing and educational facilities are shared by the two institutions to conduct the research.

The UHCL/RICIS mission is to conduct, coordinate, and disseminate research and professional level education in computing and information systems to serve the needs of the government, industry, community and academia. RICIS combines resources of UHCL and its gateway affiliates to research and develop materials, prototypes and publications on topics of mutual interest to its sponsors and researchers. Within UHCL, the mission is being implemented through interdisciplinary involvement of faculty and students from each of the four schools: Business and Public Administration, Education, Human Sciences and Humanities, and Natural and Applied Sciences. RICIS also collaborates with industry in a companion program. This program is focused on serving the research and advanced development needs of industry.

Moreover, UHCL established relationships with other universities and research organizations, having common research interests, to provide additional sources of expertise to conduct needed research. For example, UHCL has entered into a special partnership with Texas A&M University to help oversee RICIS research and education programs, while other research organizations are involved via the "gateway" concept.

A major role of RICIS then is to find the best match of sponsors, researchers and research objectives to advance knowledge in the computing and information sciences. RICIS, working jointly with its sponsors, advises on research needs, recommends principals for conducting the research, provides technical and administrative support to coordinate the research and integrates technical results into the goals of UHCL, NASA/JSC and industry.
Application of Fuzzy Logic-Neural Network Based Reinforcement Learning to Proximity and Docking Operations
RICIS Preface

This research was conducted under auspices of the Research Institute for Computing and Information Systems by Dr. Yashvant Jani of the Technology Systems Division of Togai InfraLogic, Inc. Dr. Kwok-bun Yue served as the RICIS research coordinator.

Funding was provided by the Information Systems Directorate, NASA/JSC through Cooperative Agreement NCC 9-16 between the NASA Johnson Space Center and the University of Houston-Clear Lake. The NASA research coordinator for this activity was Dr. Robert N. Lea of the Information Technology Division, NASA/JSC.

The views and conclusions contained in this report are those of the author and should not be interpreted as representative of the official policies, either express or implied, of UHCL, RICIS, NASA or the United States Government.
Research Activity AR.06

APPLICATION OF FUZZY LOGIC-NEURAL NETWORK BASED REINFORCEMENT LEARNING TO PROXIMITY AND DOCKING OPERATIONS

Interim Report submitted to
The Research Institute for Computing and Information Systems

Prepared by
Yashvant Jani
Technology Systems Division
Togai InfraLogic Inc.
Houston, Texas 77058

May 4, 1992
Table of Contents

1.0 Introduction ............................................. 1
2.0 Summary of Technical Progress ......................... 1
3.0 Future Plans ............................................. 2
4.0 Issues and Concerns .................................... 3
Appendix A. Source Listing of Fuzzy Learning Modules .... 4
Appendix B. Plots of Attitude-hold Test case ............... 23
1.0 Introduction:

As part of the RICIS activity, the reinforcement learning techniques developed at Ames Research Center are being applied to proximity and docking operations using the Shuttle and Solar Max satellite simulation. This activity is carried out in the software technology laboratory utilizing the Orbital Operations Simulator (OOS).

This interim report provides the status of the project after two months of activities and outlines the future plans. Our technical progress is summarized in section 2 and future plans are described in section 3.

2.0 Summary of Technical Progress:

Considerable technical progress, as described below, has been achieved during these two months of activities in the project.

1. The source code received from Ames Research Center contained the attitude controller however, the equations of motion still pertained to the inverted pendulum motion. In order to reflect the shuttle motion in the orbital environment, the source code was modified to interface with appropriate routines in the OOS. First, the onorbit digital auto pilot was modified to call the learn cycle in its normal calling sequence. Second, a special OOS structure was created to hold all relevant parameters for input and output. Third, an initialization routine was created to initialize the parameters of the learning algorithms. And, finally, the output of the learn cycle was interpreted and converted to rotational hand controller command. The source code is provided in Appendix A.

2. An attitude hold test case was set up with the shuttle in its normal orbit during its mission. The pitch attitude of the shuttle was set at 45 degrees in the local vertical local horizontal coordinate system. Because of the gravity gradient torques, the shuttle pitch rate starts increasing resulting in an increase in the pitch attitude. As soon as the attitude error (difference between the desired attitude and the current attitude) goes beyond the deadband limit, the fuzzy controller will initiate jet firing. Two neural networks are learning during each cycle and adjusts the parameters as required. The test case was setup to run up to 1000 seconds so there will be at least three attitude firings. For debug and analysis purposes, we initially performed the test case for only 30 seconds. Later, we extended the test case to 1000 and 100,000 seconds to understand the learning process and evaluate the performance of the fuzzy controller and neural networks. Plots for the 1000 seconds test case are included in Appendix B. This is one of the typical tests performed. So far we have performed over 10 test cases with modifications in the source code and changes in the learning parameters. Results and our findings are described separately in this section.
3. During the debug phase, we found several implementation errors. We had to change our structure, interface code and membership functions definitions. Our initial test results showed that the algorithm is holding the rate but not the attitude. We checked for all possible implementation errors, and when we were satisfied that there are no critical implementation errors, we did the test case again.

4. When we found that the results are the same (i.e. no attitude hold), we started analyzing the Tsukamoto's defuzzification method used in the algorithm. In this method, only one side of the triangle is used in computing the output action. In our attitude controller, we have used the max-dot and max-min methods to compute the output, and thus we compared the two methods for any differences. We found that the Tsukamoto's method computes the output with only half the value of the max-dot method. Thus, we changed the interpretation of the learning output for generating the rotational hand controller command. Results of our test indicated that the controller now holds attitude as well as rate. The analysis was involved and we learned a lot about the algorithm.

5. Since our test was successful in holding the rate and attitude, we started analyzing the learning behavior neural networks. Our results indicate that the parameter 'f' that relates to the firing strength varies in the same manner for all rules. The 'd' parameters also show the same variations for all rules. We have discussed the problem with Dr. Berenji at Ames and sent him the source listing as well as plots so that he can compare it with his results and tell us if we are doing something wrong.

6. We have analyzed the effect of the bias term x(2) and have concluded that the its most appropriate value is 0.5. We performed tests with bias values of 0.0 and 1.0 and results were really bad. For these values the controller does not perform its functions and neural networks do not learn.

7. We were invited by the SPIE fuzzy logic and neural network committee to present our initial results at the Orlando conference and special workshop on Fuzzy-Neuro control. We presented our preliminary results at this workshop on April 23, 1992.

3.0 Future Plans:

We plan to continue testing of the fuzzy learning algorithms utilizing the attitude hold test case in the orbital operations simulator. Emphasis during this activity is to ensure that the algorithms perform properly and learning by the neural networks in ARIC architecture is achieved in a satisfactory manner. Then we will switch to other attitude control tests such as attitude maneuver, rate hold and rate maneuver. We will perform these tests with proper perturbations typically present during the orbital operations.

As soon as we complete testing the attitude controller with learning, we will implement the translational controller in the simulation and perform proximity
operations test cases. Currently we plan to perform v-bar, r-bar, fly around and station-keeping test cases as we have performed these test cases to check out designs of our translational controller. Finally, we will set up a test case that will simulate docking operations. In this test case, the shuttle will approach the solar max satellite from 50 feet to 2 feet and hold the relative orientation for a specified time at the final distance so that the grappling task can be performed.

4. Issues and Concerns :

At this time the project is on schedule and there are no issues or concerns to report in this interim report.
Appendix A. Source Listing of Fuzzy Learning Modules
#include <stdio.h>
#include <math.h>
#include <sys/types.h>

/* EXTERNAL DATA STRUCTURE DEFINITION */
#include "../orb_fuzzy/learn_cycle.h"

int counter = 0;
#define max(x,y) ((x >= y) ? x : y)
#define min(x,y) ((x < y) ? x : y)
#define Gamma 0.9
#define Beta 0.2
#define Beta_h 0.05
#define Rho 1.0
#define Rho_h 0.2
#define Rho1 2.0
#define Rho_h1 0.4
extern double sgn();
extern double exp();
extern double rnd();

learn_cycle(L)
LEARN_CYCLE * L; /*IN : */
{
    int i, i1, j, k;
    double match(), calculate_z_array();

    double temp ;
L->x[2] = 0.50;

L->failure = 0;

/* Set up and evaluate the failure criteria */
if ((fabs(L->Phi) > 0.7) || (fabs(L->Phi_dot) > 0.07))
L->failure = -1.;

/* output: state evaluation */
for (i = 0; i < 31; i++)
{
    L->sum = 0.0;
    for(j = 0; j < 3; j++)
    {
        L->sum += L->a[i*3+j] * L->x_old[j];
    }
    L->y[i] = 1.0 / (1.0 + exp(-1.0 * L->sum));
}
L->sum = 0.0;
for (i = 0; i < 31; i++)
{
    L->sum += L->c[i] * L->y[i];
}
L->sum1 = 0.0;
for (j = 0; j < 3; j++)
{
    L->sum1 += L->b[j] * L->x_old[j];
}
L->v = L->sum + L->sum1;

/* output: action */
for (i = 0; i < 31; i++)
{
    i1 = i;
    L->w[i] = match(i1, L);
    L->z1[i] = calculate_z_array(i1, L);
}
L->sum1 = 0.0;
L->denom = 0.0;
for (i = 0; i < 31; i++)
{  
    L->sum1 += L->w[i] * L->z1[i] * L->f[i];  
    L->denom += L->w[i] * L->f[i];  
}
L->push = L->sum1 / L->denom;

/* output: action computations completed */
for(i = 0; i < 31; i++)
{
    L->sum = 0.0;

    for (j = 0; j < 3; j++)
        L->sum += L->d[i*3+j] * L->x_old[j];

    L->z[i] = 1.0 / (1.0 + exp(-1.0 * L->sum));
}
L->sum2 = 0.0;
L->sum3 = 0.0;

for(i = 0; i < 3; i++)
    L->sum2 += L->e[i] * L->x_old[i];

for (i = 0; i < 31; i++)
    L->sum3 += L->f[i] * L->z[i];

C */
/* 4 May 1992 - Normalize sum3 */
/* L->sum4 = L->sum3 + L->sum2; */
C */
L->sum4 = (L->sum3 / 31.0) + L->sum2;

L->p = 1.0 / (1.0 + exp(-1.0 * L->sum4));

C */
/* 15 April 1992 - Use temp variable - not push */
C */
/* L->push = (rnd() <= L->p) ? L->push : -L->push; */
/* L->unusualness = (L->push > 0) ? 1.0 - L->p : -L->p; */
temp = (rnd() <= L->p) ? L->push : -L->push;
L->unusualness = (temp > 0) ? 1.0 - L->p : -L->p;
/ * using new input values and unmodified weights. */
/* Use y_new and v_new so not to destroy y and v. */
for(i = 0; i < 31; i++)
{
    L->sum = 0.0;
    for(j = 0; j < 3; j++)
    {
        L->sum += L->a[i*3+j] * L->x[j];
    }
    L->y_new[i] = 1.0 / (1.0 + exp(-1.0 * L->sum));
}
L->sum = 0.0;
L->sum1 = 0.0;
L->sum2 = 0.0;
for(j = 0; j < 3; j++)
    L->sum1 += L->b[j] * L->x[j];
for(i = 0; i < 31; i++)
    L->sum2 += L->c[i] * L->y_new[i];
L->sum = L->sum1 + L->sum2;
L->v_new = L->sum;

/* action evaluation */
if (L->failure)
    L->r_hat = L->failure - L->v;
else
    L->r_hat = L->failure + Gamma * L->v_new - L->v;

/* modification and update to parameters */
for(i = 0; i < 31; i++)
{
    L->factor1 = Beta_h * L->r_hat * L->y[i] * (1.0 - L->y[i]) * sgn(L->c[i]);
    L->c[i] += Beta * L->r_hat * L->y[i];
    for(j = 0; j < 3; j++)
    { 
        L->a[i*3+j] += L->factor1 * L->x_old[j];
    }
}
for(i = 0; i < 3; i++)
L->b[i] += Beta * L->r_hat * L->x_old[i];

for(i = 0; i < 31; i++)
{
    L->factor2 = Rho_h * L->r_hat * L->z[i] * (1.0 - L->z[i]) * sgn(L->f[i]) * L->unusualness;
    for(j = 0; j < 3; j++)
        L->d[i*3+j] += L->factor2 * L->x_old[j];
}

for(i = 0; i < 31; i++)
{
    L->f[i] += Rho * L->r_hat * L->unusualness * L->z[i];
}

for(i = 0; i < 3; i++)
{
    L->e[i] += Rho * L->r_hat * L->unusualness * L->x_old[i];
}
L->x_old[0] = L->x[0];
L->x_old[1] = L->x[1];
L->x_old[2] = L->x[2];

/* zero_one function returns 0 for negative numbers
   1 for values > 1
   x for values between 0 and 1 */
double zero_one(x)
    double x;
{
    if (x < 0.0)
        return (-1.0);
    else if (x > 0.0)
        return (1.0);
    else
        return (0.0);
}
double x;
{
    if (x < 0) return (0.0);
    else if (x > 1) return (1.0);
    else return (x);
}

/******* Membership Function for Phi *******************/

double nb1(x)
double x;
{
    return(min( max((-2-x)/3,0.0 ), 1.0 ));
}

double nm1(x)
double x;
{
    if (x <= -2.5) return (min( max(( x+4)/1.5, 0.0 ), 1.0 ));
    else return (min( max(( -x-1)/1.5, 0.0 ), 1.0 ));
}

double ns1(x)
double x;
{
    if (x <= -1.0) return (min( max( x+2.0, 0.0 ), 1.0 ));
    else return (min( max(-x, 0.0 ), 1.0 ));
}

double zo1(x)
double x;
{
    if (x <= 0) return (min( max( x+1 , 0.0 ), 1.0 ));
    else return (min( max(-x+1 , 0.0 ), 1.0 ));
}

double ps1(x)
double x;
{
    if (x <= 1) return (min( max( x, 0.0 ), 1.0 ));
    else return (min( max(-x+2 , 0.0 ), 1.0 ));
}

double pm1(x)
double x;
{
    if (x <= 2.5) return (min( max(( x-1)/1.5, 0.0 ), 1.0 ));
    else return (min( max(( -x+4)/1.5, 0.0 ), 1.0 ));
}
double pbl(x)  
{     return(min( max(( x-2)/3 , 0.0 ), 1.0 ));
}

******************* Phi_dot Membership Functions *******************

double nb2(x)  
{     return(min( max((-.2-x)/.3,0.0 ), 1.0 ));
}

double nm2(x)  
{     if (x <-.25) return (min( max(( x+.4)/.15,0.0 ), 1.0 ));
              else return (min( max((-x-.1)/.15,0.0 ), 1.0 ));
}

double ns2(x)  
{     if (x <= -.1) return (min( max(( x+.2)/.1, 0.0 ), 1.0 ));
              else return (min( max((-x/.1), 0.0 ), 1.0 ));
}

double zo2(x)  
{     if (x <<0) return (min( max(( x+.1)/.1, 0.0 ), 1.0 ));
              else return (min( max((-x+.1)/.1, 0.0 ), 1.0 ));
}

double ps2(x)  
{     if (x <<.1) return (min( max(( x/.1), 0.0 ), 1.0 ));
              else return (min( max((-x+.2)/.1, 0.0 ), 1.0 ));
}

double pm2(x)  
{     if (x <= .25) return (min( max(( x-.1)/.15,0.0 ), 1.0 ));
              else return (min( max(( -x+.4)/.15,0.0 ), 1.0 ));
}

double pb2(x)
double x;
{
    return(min( max(( x-.2)/.3 , 0.0 ), 1.0 ));
}

/*****************
Defuzzification Process with Accel Membership Functions **************/

double nm3(x)
double x;
{
    return(-2-x);
}

double ns3(x)
double x;
{
    return(-2*x);
}

double zo3(x)
double x;
{
    return(0.0);
}

double ps3(x)
double x;
{
    return(2*x);
}

double pm3(x)
double x;
{
    return(2+x);
}

double match(i,L)
int i;
LEARN_CYCLE *L; /*IN : */
{
    double temp;
    switch (i) {
        case 0:
temp = min( zero_one(nb1(L->Phi)*L->d[i*3+0]), zero_one(zo2(L->Phi_dot)*L->d[i*3+1]));
/* REMOVE "BIAS" REFERENCE */
temp = min(temp, zero_one(L->x[2]*L->d[i*3+2]));
/* */
return(temp);
case 1:
    temp = min( zero_one(nb1(L->Phi)*L->d[i*3+0]), zero_one(ns2(L->Phi_dot)*L->d[i*3+1]));
/* REMOVE "BIAS" REFERENCE */
temp = min(temp, zero_one(L->x[2]*L->d[i*3+2]));
/* */
return(temp);
case 2:
    temp = min( zero_one(nb1(L->Phi)*L->d[i*3+0]), zero_one(nm2(L->Phi_dot)*L->d[i*3+1]));
/* REMOVE "BIAS" REFERENCE */
temp = min(temp, zero_one(L->x[2]*L->d[i*3+2]));
/* */
return(temp);
case 3:
    temp = min( zero_one(nb1(L->Phi)*L->d[i*3+0]), zero_one(nb2(L->Phi_dot)*L->d[i*3+1]));
/* REMOVE "BIAS" REFERENCE */
temp = min(temp, zero_one(L->x[2]*L->d[i*3+2]));
/* */
return(temp);
case 4:
    temp = min( zero_one(nm1(L->Phi)*L->d[i*3+0]), zero_one(zo2(L->Phi_dot)*L->d[i*3+1]));
/* REMOVE "BIAS" REFERENCE */
temp = min(temp, zero_one(L->x[2]*L->d[i*3+2]));
/* */
return(temp);
case 5:
    temp = min( zero_one(nm1(L->Phi)*L->d[i*3+0]), zero_one(ns2(L->Phi_dot)*L->d[i*3+1]));
/* REMOVE "BIAS" REFERENCE */
temp = min(temp, zero_one(L->x[2]*L->d[i*3+2]));
/* */
return(temp);
case 6:
    temp = min( zero_one(nm1(L->Phi)*L->d[i*3+0]), zero_one(nm2(L->Phi_dot)*L->d[i*3+1]));
/* REMOVE "BIAS" REFERENCE */
temp = min(temp, zero_one(L->x[2]*L->d[i*3+2]));
return(temp);
case 7:
    temp=min( zero_one(nm1(L->Phi)*L->d[i*3+0]), zero_one(nb2(L->Phi_dot)*L->d[i*3+1]));
/* REMOVE "BIAS" REFERENCE
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
    return(temp);

case 8:
    temp=min( zero_one(ns1(L->Phi)*L->d[i*3+0]), zero_one(zo2(L->Phi_dot)*L->d[i*3+1]));
/* REMOVE "BIAS" REFERENCE
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
    return(temp);

case 9:
    temp=min( zero_one(ns1(L->Phi)*L->d[i*3+0]), zero_one(ns2(L->Phi_dot)*L->d[i*3+1]));
/* REMOVE "BIAS" REFERENCE
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
    return(temp);

case 10:
    temp=min( zero_one(ns1(L->Phi)*L->d[i*3+0]), zero_one(nm2(L->Phi_dot)*L->d[i*3+1]));
/* REMOVE "BIAS" REFERENCE
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
    return(temp);

case 11:
    temp=min( zero_one(ns1(L->Phi)*L->d[i*3+0]), zero_one(nb2(L->Phi_dot)*L->d[i*3+1]));
/* REMOVE "BIAS" REFERENCE
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
    return(temp);

case 12:
    temp=min( zero_one(zo1(L->Phi)*L->d[i*3+0]), zero_one(pb2(L->Phi_dot)*L->d[i*3+1]));
/* REMOVE "BIAS" REFERENCE
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
    return(temp);

case 13:
    temp=min( zero_one(zo1(L->Phi)*L->d[i*3+0]), zero_one(pm2(L->Phi_dot)*L->d[i*3+1]));
1]));
/* REMOVE "BIAS" REFERENCE
  temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
  return(temp);
  case 14:
    temp=min( zero_one(zo1(L->Phi)*L->d[i*3+0]), zero_one(ps2(L->Phi_dot)*L->d[i*3+1])));
/* REMOVE "BIAS" REFERENCE
  temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
  return(temp);
  case 15:
    temp=min( zero_one(zo1(L->Phi)*L->d[i*3+0]), zero_one(zo2(L->Phi_dot)*L->d[i*3+1])));
/* REMOVE "BIAS" REFERENCE
  temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
  return(temp);
  case 16:
    temp=min( zero_one(zo1(L->Phi)*L->d[i*3+0]), zero_one(ns2(L->Phi_dot)*L->d[i*3+1])));
/* REMOVE "BIAS" REFERENCE
  temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
  return(temp);
  case 17:
    temp=min( zero_one(zo1(L->Phi)*L->d[i*3+0]), zero_one(nm2(L->Phi_dot)*L->d[i*3+1])));
/* REMOVE "BIAS" REFERENCE
  temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
  return(temp);
  case 18:
    temp=min( zero_one(zo1(L->Phi)*L->d[i*3+0]), zero_one(nb2(L->Phi_dot)*L->d[i*3+1])));
/* REMOVE "BIAS" REFERENCE
  temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
  return(temp);
  case 19:
    temp=min( zero_one(ps1(L->Phi)*L->d[i*3+0]), zero_one(pb2(L->Phi_dot)*L->d[i*3+1])));
/* REMOVE "BIAS" REFERENCE
  temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
  return(temp);
return(temp);
case 20:
    temp=min( zero_one(ps1(L->Phi)*L->d[i*3+0]), zero_one(pm2(L->Phi_dot)*L->d[i*3+1]));
    /* REMOVE "BIAS" REFERENCE */
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
    */
    return(temp);

case 21:
    temp=min( zero_one(ps1(L->Phi)*L->d[i*3+0]), zero_one(ps2(L->Phi_dot)*L->d[i*3+1]));
    /* REMOVE "BIAS" REFERENCE */
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
    */
    return(temp);

case 22:
    temp=min( zero_one(pm1(L->Phi)*L->d[i*3+0]), zero_one(zo2(L->Phi_dot)*L->d[i*3+1]));
    /* REMOVE "BIAS" REFERENCE */
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
    */
    return(temp);

case 23:
    temp=min( zero_one(pm1(L->Phi)*L->d[i*3+0]), zero_one(pb2(L->Phi_dot)*L->d[i*3+1]));
    /* REMOVE "BIAS" REFERENCE */
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
    */
    return(temp);

case 24:
    temp=min( zero_one(pm1(L->Phi)*L->d[i*3+0]), zero_one(pm2(L->Phi_dot)*L->d[i*3+1]));
    /* REMOVE "BIAS" REFERENCE */
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
    */
    return(temp);

case 25:
    temp=min( zero_one(pm1(L->Phi)*L->d[i*3+0]), zero_one(ps2(L->Phi_dot)*L->d[i*3+1]));
    /* REMOVE "BIAS" REFERENCE */
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
    */
    return(temp);

case 26:
    temp=min( zero_one(pm1(L->Phi)*L->d[i*3+0]), zero_one(zo2(L->Phi_dot)*L->d[i*3+1]));
    /* REMOVE "BIAS" REFERENCE */
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
    */
/* REMOVE "BIAS" REFERENCE */
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
    return(temp);
    case 27:
        temp=min( zero_one(pb1(L->Phi)*L->d[i*3+0]), zero_one(pb2(L->Phi_dot)*L->d[i*3+1]));
    /* REMOVE "BIAS" REFERENCE */
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
    return(temp);
    case 28:
        temp=min( zero_one(pb1(L->Phi)*L->d[i*3+0]), zero_one(pm2(L->Phi_dot)*L->d[i*3+1]));
    /* REMOVE "BIAS" REFERENCE */
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
    return(temp);
    case 29:
        temp=min( zero_one(pbl(L->Phi)*L->d[i*3+0]), zero_one(ps2(L->Phi_dot)*L->d[i*3+1]));
    /* REMOVE "BIAS" REFERENCE */
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
    return(temp);
    case 30:
        temp=min( zero_one(pbl(L->Phi)*L->d[i*3+0]), zero_one(zo2(L->Phi_dot)*L->d[i*3+1]));
    /* REMOVE "BIAS" REFERENCE */
    temp=min(temp, zero_one(L->x[2]*L->d[i*3+2]));
*/
    return(temp);
}

double calculate_z_array(i, L)
int i;
LEARN_CYCLE * L; /*IN :*/
{
    switch (i) {
    case 0:
        return(ps3(L->w[0]));
    case 1:
        return(ps3(L->w[1]));
    case 2:
        return(pm3(L->w[2]));
case 3:
    return(pm3(L->w[3]));
case 4:
    return(ps3(L->w[4]));
case 5:
    return(ps3(L->w[5]));
case 6:
    return(pm3(L->w[6]));
case 7:
    return(pm3(L->w[7]));
case 8:
    return(ps3(L->w[8]));
case 9:
    return(ps3(L->w[9]));
case 10:
    return(ps3(L->w[10]));
case 11:
    return(ps3(L->w[11]));
case 12:
    return(nm3(L->w[12]));
case 13:
    return(nm3(L->w[13]));
case 14:
    return(ns3(L->w[14]));
case 15:
    return(zo3(L->w[15]));
case 16:
    return(ps3(L->w[16]));
case 17:
    return(pm3(L->w[17]));
case 18:
    return(pm3(L->w[18]));
case 19:
    return(ns3(L->w[19]));
case 20:
    return(ns3(L->w[20]));
case 21:
    return(ns3(L->w[21]));
case 22:
    return(ns3(L->w[22]));
case 23:
    return(nm3(L->w[23]));
case 24:
    return(nm3(L->w[24]));
case 25:
    return(ns3(L->w[25]));
case 26:
    return(ns3(L->w[26]));
case 27:
    return(nm3(L->w[27]));
case 28:
    return(nm3(L->w[28]));
case 29:
    return(ns3(L->w[29]));
case 30:
    return(ns3(L->w[30]));
}
typedef struct {
    int start_state; /* < > */
    int learn_flag; /* < > */
    int n1; /* < > counter */
    int n2; /* < > counter */
    double Phi; /*deg < > angle error */
    double Phi_dot; /*deg < > rate error */
    double failure; /* < > */
    double x_old[3]; /* < > */
    double y_new[31]; /* < > */
    double z1[31]; /* < > */
    double f1[31]; /* < > */
    double e1[31]; /* < > */
    double w[31]; /* < > */
    double z[31]; /* < > */
    double f[31]; /* < > */
    double x[3]; /* < > */
    double a[93]; /* < > */
    double d[93]; /* < > */
    double b[3]; /* < > */
    double c[31]; /* < > */
    double e[31]; /* < > */
    double y[31]; /* < > */
    double v_new; /* < > */
    double r_new; /* < > FORCE APPLIED */
    double push; /* < > */
    double unusualness; /* < > */
    double sum; /* < > */
    double sum1; /* < > */
    double sum2; /* < > */
    double sum3; /* < > */
}
double sum4 ; /* < > */
double sumd ; /* < > */
double factor1 ; /* < > */
double factor2 ; /* < > */
double factor3 ; /* < > */
double denom ; /* < > */
double v; /* failure related parameter */
double p; /* failure related parameter */

} LEARN_CYCLE ;
#include <stdio.h>
#include <math.h>
#include <sys/types.h>

Suspicious: z_

/* EXTERNAL DATA STRUCTURE DEFINITION */

#include "../orb_fuzzy/learn_cycle.h"

double md0
    /* Returns a floating-point between 0 and 1, including 0. */
    /* rand is a number between 0 and 2^31 - 1 */

    return ((double) rand() / (double)(RAND_MAX));

set_rnd_weights (L)
LEARN_CYCLE * L;
{
    int i,j;

    for (i = 0; i < 3; i++)
    L->b[i] = rnd0 * 0.2 - 0.1;
    for (i = 0; i < 31; i++)
    {
        L->c[i] = rnd0 * 0.2 - 0.1;
        L->e[i] = rnd0 * 0.2 - 0.1;
        L->f[i] = rnd0 * 0.5 + 0.5;
        L->w[i] = 1.0;
    }
    L->x_old[0] = L->x_old[1] = L->x_old[2] = 0.0;

    for (j = 0; j < 93; j++)
    {
        L->a[j] = rnd0 * 0.2 - 0.1;
        L->d[j] = 1.0;
    }
}
Appendix B. Plots of Attitude-hold Test case
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

LVLH EULER PYR ROLL vs TIME
RUN: Att Hold Vernier (Tight DB)

VEHICLE: ORB_FUZZ_BATCH
DATA SAMPLING FREQUENCY: 0.500 Hz
LVLH EULER PYR PITCH vs TIME
RUN: Att Hold Vernier (Tight DB)

VEHICLE: ORB_FUZ_BATCH.ma
DATA SAMPLING FREQUENCY: 0.500 Hz
LVLH EULER PYR YAW vs TIME
RUN: Att Hold Vernier (Tight DB)

VEHICLE: ORB_FUZZ_BATCH
DATA SAMPLING FREQUENCY: 0.500 Hz
LVLH BIASED BODY ROLL RATE vs TIME

RUN: Att Hold Vernier (Tight DB)

VEHICLE: ORB_FU2Z BATCH
DATA SAMPLING FREQUENCY: 0.500 Hz
LVLH BIASED BODY PITCH RATE vs TIME

RUN: Att Hold Vernier (Tight DB)

VEHICLE: ORB_FUZZ_BATCH.state
DATA SAMPLING FREQUENCY: 0.500 Hz
LVLH BIASED BODY YAW RATE vs TIME
RUN: Att Hold Vernier (Tight DB)

VEHICLE: ORB_FUZZ_BATC_Lmtte
DATA SAMPLING FREQUENCY: 0.500 Hz
att[0] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.fsw
DATA SAMPLING FREQUENCY: 0.500 Hz
att[1] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.fsw
DATA SAMPLING FREQUENCY: 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

atter[2] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.fsw
DATA SAMPLING FREQUENCY: 0.500 Hz
rter[0] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.fsw
DATA SAMPLING FREQUENCY: 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

rter[1] vs TIME
RUN: All Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.fsw
DATA SAMPLING FREQUENCY: 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

rter[2] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.fzw
DATA SAMPLING FREQUENCY: 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

sfuel vs TIME
RUN: Att Hold Vernier (Tight DB)

TIME (sec)

MODULE: ORB_FUZZ_BATCH.vernier
DATA SAMPLING FREQUENCY: 0.500 Hz
w[8] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.Jean2
DATA SAMPLING FREQUENCY: 0.500 Hz
w[9] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH

DATA SAMPLING FREQUENCY: 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

w[14] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH1em2
DATA SAMPLING FREQUENCY: 0.500 Hz

ORBITAL OPERATIONS SIMULATOR
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

\[ w[15] \text{ vs TIME} \]
RUN: Att Hold Vernier (Tight DB)

\[ w[15] \]

\begin{align*}
&0.4 \quad 0.45 \quad 0.5 \quad 0.55 \quad 0.6 \quad 0.65 \quad 0.7 \quad 0.75 \quad 0.8 \quad 0.85 \quad 0.9 \quad 0.95 \quad 1 \\
&0 \quad 100 \quad 200 \quad 300 \quad 400 \quad 500 \quad 600 \quad 700 \quad 800 \quad 900 \quad 1000
\end{align*}

TIME (sec)

MODULE: ORB_FUZZ_BATCH.lemm2
DATA SAMPLING FREQUENCY: 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

w[16] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH_learn2
DATA SAMPLING FREQUENCY: 0.500 Hz
w[21] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.lean2
DATA SAMPLING FREQUENCY: 0.500 Hz
push vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.lem2
DATA SAMPLING FREQUENCY: 0.500 Hz
unusualness vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH
DATA SAMPLING FREQUENCY: 0.500 Hz

ORBITAL OPERATIONS SIMULATOR
failure vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.learn2
DATA SAMPLING FREQUENCY: 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

\[ x[0] \text{ vs TIME} \]
RUN: Att Hold Vernier (Tight DB)

\[ x[0] \]
-2.77556e-17

\[ 0.6 \]
\[ 0.5 \]
\[ 0.4 \]
\[ 0.3 \]
\[ 0.2 \]
\[ 0.1 \]
\[ 0.0 \]
\[ -0.1 \]
\[ -0.2 \]
\[ -0.3 \]
\[ -0.4 \]
\[ -0.5 \]

\[ 0 \]
\[ 100 \]
\[ 200 \]
\[ 300 \]
\[ 400 \]
\[ 500 \]
\[ 600 \]
\[ 700 \]
\[ 800 \]
\[ 900 \]
\[ 1000 \]

TIME (sec)

MODULE: ORB_FUZZ_BATCH.lem2
DATA SAMPLING FREQUENCY: 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

\[ x[1] \text{ vs TIME} \]
RUN: Att Hold Vernier (Tight DB)

\[ \begin{array}{c|c|c|c|c|c|c|c|c|c|c|c} \hline \text{TIME (sec)} & 0 & 100 & 200 & 300 & 400 & 500 & 600 & 700 & 800 & 900 & 1000 \\ \hline \hline x[1] & 1.38778e-17 & 0.05 & 0.1 & 0.15 & 0.15 & 0.1 & 0.05 & 1.38778e-17 & 0.15 & 0.1 & 0.05 \\ \hline \end{array} \]

MODULE: ORB_FUZZ_BATCH.iam2
DATA SAMPLING FREQUENCY: 0.500 Hz

ORBITAL OPERATIONS SIMULATOR 19 Wed Apr 29 1992 04:35:55 PM
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

\[ x[2] \text{ vs TIME} \]
RUN: Att Hold Vernier (Tight DB)

\[
\begin{align*}
&\text{TIME (sec)} \\
&0 \ 100 \ 200 \ 300 \ 400 \ 500 \ 600 \ 700 \ 800 \ 900 \ 1000 \\
&0.499 \ x[2] \ 0.5 \ 0.501
\end{align*}
\]

MODULE: ORB_FUZZ_BATCH.lesm2
DATA SAMPLING FREQUENCY: 0.500 Hz
1.00415

\[ d[24] \text{ vs TIME} \]

RUN: Att Hold Vernier (Tight DB)

1.00205

\[ d[24] \]

0.99995

TIME (sec)

0 100 200 300 400 500 600 700 800 900 1000

MODULE: ORB_FUZZ_BATCH.learn2
DATA SAMPLING FREQUENCY: 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

\[ d[25] \text{ vs TIME} \]

**RUN:** At Hold Vernier (Tight DB)

**TIME (sec)**

**MODULE:** ORB_FUZZ_BATCH,team2

**DATA SAMPLING FREQUENCY:** 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

\[ d[26] \text{ vs TIME} \]

RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.coin2
DATA SAMPLING FREQUENCY: 0.500 Hz
d[27] vs TIME
RUN: Alt Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.lem2
DATA SAMPLING FREQUENCY: 0.500 Hz
d[28] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.lem2
DATA SAMPLING FREQUENCY: 0.500 Hz
d[29] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_BAT_1eam2
DATA SAMPLING FREQUENCY: 0.500 Hz

TIME (sec)
**D[42] vs TIME**

**RUN:** Att Hold Vernier (Tight DB)

---

**MODULE:** ORB_FUZZ_BATCHlearn2

**DATA SAMPLING FREQUENCY:** 0.500 Hz
d[43] vs TIME
RUN: All Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.lasm2
DATA SAMPLING FREQUENCY: 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

d[44] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH,Iam2
DATA SAMPLING FREQUENCY: 0.500 Hz
d(45) vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.team2
DATA SAMPLING FREQUENCY: 0.500 Hz
d[46] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.learn2
DATA SAMPLING FREQUENCY: 0.500 Hz
**SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)**

**d[47] vs TIME**  
**RUN: Att Hold Vernier (Tight DB)**

<table>
<thead>
<tr>
<th>TIME (sec)</th>
<th>0</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
<th>600</th>
<th>700</th>
<th>800</th>
<th>900</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.999001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**MODULE:** ORB_FUZZ_BATCH_lean2  
**DATA SAMPLING FREQUENCY:** 0.500 Hz

**ORBITAL OPERATIONS SIMULATOR**

32  

Wed Apr 29 1992 04:35:55 PM
d[48] vs TIME
RUN: Att Hold Vernier (Tight DB)

1.00415

1.00205

d[48]

0.99995

TIME (sec)

MODULE: ORB_FUZZ_BATCH leases2
DATA SAMPLING FREQUENCY: 0.500 Hz
d[49] vs TIME
RUN: At Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCHElearn2
DATA SAMPLING FREQUENCY: 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

d[50] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.lem2
DATA SAMPLING FREQUENCY: 0.500 Hz

ORBITAL OPERATIONS SIMULATOR
d[63] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODUUJE: ORB FUZZ_BATCH.learn2
DATA SAMPLING FREQUENCY: 0.500 Hz
d[64] vs TIME
RUN: Au Hold Vernier (Tight DB)

1.00225

d[64]

TIME (sec)

MODULE: ORB_FUZZ_BATCH.learn2
DATA SAMPLING FREQUENCY: 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

d[65] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCHLearn2
DATA SAMPLING FREQUENCY: 0.500 Hz
d[66] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.learn2
DATA SAMPLING FREQUENCY: 0.500 Hz
d[67] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH

DATA SAMPLING FREQUENCY: 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

**d[68] vs TIME**

**RUN:** Att Hold Vernier (Tight DB)

- **TIME (sec):** 0 100 200 300 400 500 600 700 800 900 1000
- **d[68]:** 0.999001 – 1.001

**MODULE:** ORB_FUZZ_BATCH.learn2

**DATA SAMPLING FREQUENCY:** 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

sfuel vs TIME
RUN: Az Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH, primary
DATA SAMPLING FREQUENCY: 0.500 Hz

ORBITAL OPERATIONS SIMULATOR
w[22] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.learn2
DATA SAMPLING FREQUENCY: 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

\( f(8) \) vs TIME
RUN: Alt Hold Vernier (Tight DB)

 MODULE: ORB_FUZZ_BATCH_1em2
DATA SAMPLING FREQUENCY: 0.500 Hz

ORBITAL OPERATIONS SIMULATOR

1
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

f[9] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.tmem2
DATA SAMPLING FREQUENCY: 0.500 Hz

ORBITAL OPERATIONS SIMULATOR

ORBITAL OPERATIONS SIMULATOR 2

Wed Apr 29 1992 04:35:55 PM
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

\[ f[14] \text{ vs TIME} \]
RUN: Att Hold Vernier (Tight DB)

TIME (sec)

MODULE: ORB_FUZZ_BATCH_load2
DATA SAMPLING FREQUENCY: 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

\[ f[15] \text{ vs TIME} \]

**RUN:** Att Hold Vernier (Tight DB)

---

**MODULE:** ORB_FUZZ_BATCH

**DATA SAMPLING FREQUENCY:** 0.500 Hz

---

**ORBITAL OPERATIONS SIMULATOR**

Wed Apr 29 1992 04:35:55 PM
**MODULE: ORB_FUZZ_BATCH.iem2**

**DATA SAMPLING FREQUENCY: 0.500 Hz**
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

f[21] vs TIME
RUN: Att Hold Vernier (Tight DB)

MODULE: ORB_FU22_BATCH1am2
DATA SAMPLING FREQUENCY: 0.500 Hz
SIMULATION APPLICATION: FUZZY ORBITER BATCH APPLICATION (9/14/90)

f[22] vs TIME
RUN: Art Hold Vernier (Tight DB)

MODULE: ORB_FUZZ_BATCH.learn2
DATA SAMPLING FREQUENCY: 0.500 Hz