Trajectory determination of war artillery projectile using neural networks & fuzzy logic

KUMBHARE PRATHMESH RAJENDRA
Dept. of Electrical Engineering
Missouri University of Science & Technology
Rolla, USA
pkxvd@mst.edu

***Abstract*—— a novel way to eliminate the need of firing table is introduced. Trajectory for a given range of the target and wind velocity is modelled. For demonstration, case study of 155mm HE M101 howitzer projectile is presented. A fuzzy logic system consisting of IF-THEN rules is constructed which gives angle of firing for particular wind velocity & range with less computation. More comprehensive data from trajectory tracking is obtained to form input & target data. With the use of this information, neural network is designed & trained by backpropagation algorithm. This network is then used for providing angle of firing for any given wind velocity and range. Finally, the report concludes with the comparison between fuzzy logic based system & neural network based system & traditional firing table.**

***Index Terms*—Neural networks, point mass system, fuzzy logic, forward observer (FO), firing table.**

I. INTRODUCTION

Determination of elevation for a known range of target is a very complicated procedure. The firing techniques employed in the past were time consuming and a bit tedious (Fig. 1). With the introduction of modern technologies this process has been considerably improved with increased accuracy. Another novel approach towards the firing technique of Howitzer cannons is introduced in this report. Increased simplicity in designing of firing algorithm and also improvement in the time required for firing is demonstrated by this approach. Fuzzy logic and neural network is utilized by this algorithm. Projectile motion is an active research area and its accuracy is constantly improving but for the sake of simplicity point mass model equations (1), (2), (3) are used for study with some assumptions

In the early stages the distance at which the target is located was given by one person called as forward observer (FO). This person who was situated in the nearby area would compute the distance at which the target is located with the help of binoculars. Once the distance was determined, required elevation was calculated from firing table. Depending upon the elevation the cannon would be adjusted and then fired. The movement of the cannon was done manually by the soldiers in the beginning but today the motors and gears drives the cannon. This method makes use of something called as firing table during firing. The firing table is nothing but a database created by taking tests of the cannon. This table accounts for wind velocity, air temperature and atmospheric pressure correction factors in the trajectory determination. So each time the forward observer gives distance of the target, using firing table data the soldiers set the cannon accordingly. Making of firing table is long procedure with possibility of human error in calculation [2]. Various type of chart are constructed indicating relationship between different factors. Determining trajectory requires extraction of data from firing table [3]. This process requires manual calculation & it takes some time.

Two different versions for elevation computation are presented, with one constructed using neural network while the other utilizes if-then rules of fuzzy logic. As above mentioned, determination of elevation is a lengthy process which requires manual calculation. The purpose of the firing algorithm is to minimize the involvement of human from computation so that it eliminates human error and eventually increase rate of firing. The fuzzy logic algorithm uses condition which determine elevation if range and wind velocity satisfy constructed rule. Second approach requires input and target data for training the neural network. This trained network using backpropagation algorithm can provide elevation value for any input parameters. Elevation is essentially a non-linear function of range, wind & various other factors. So, its computation using fuzzy logic and neural network is convenient.

The aim of this report is to present a method which provides certain advantage over traditional firing method. The algorithm do not provide increased accuracy in the trajectory computation as the data used is limited and the trajectory computation equations are basic, neglecting many factors which are encountered in practical application. The report is organized as follows. The logic used for algorithm construction is covered in Section II. Simulation results obtained from the case study of 155mm he 101 projectile are presented and analyzed in Section III. Conclusions based on comparison between fuzzy logic system, neural network system and traditional firing table are included in Section IV. Matlab codes used for simulation are included in the appendix.

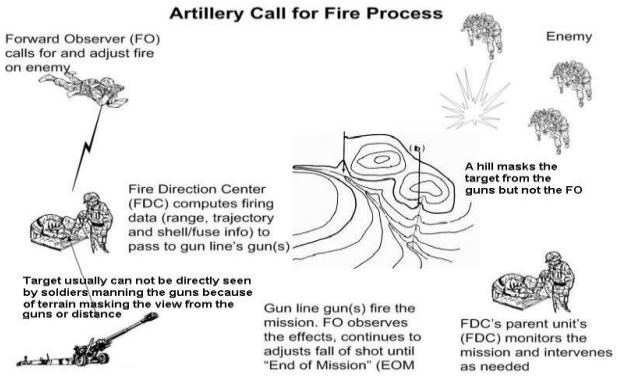

Fig. 1. Traditional Firing Process

II. METHODOLOGY

A. Projectile Motion Equations

Projectile motion is the most basic form of motion in which an object is thrown near the earth's surface. It follows a curved path under the action of gravity only. In this scenario gravity acts in downward direction to cause acceleration. There are no other forces acting on the projectile & it is assumed that motion takes place in vacuum. In practical situations, many aerodynamic forces acting on the projectile are encountered which affect the trajectory significantly. Various models are used for trajectory computation according to required accuracy [4]. Complexity of the model increases as accuracy increases, with vacuum model giving lowest accuracy & six degree of freedom model giving highest accuracy.

In this report, point mass model is used for trajectory computation [1]. This model takes into account air density of the atmosphere, drag coefficient, ballistic coefficient, wind & gravitational acceleration due to earth. The equations are mathematically represented as:

$$\ddot{x} = (-\Omega V K_d/C)(\dot{x}-W_x)+a_x \quad (1)$$

$$\ddot{y} = (-\Omega V K_d\, \dot{y}/C)(-g)+a_y \quad (2)$$

$$\ddot{z} = (-\Omega V K_d/C)(\dot{z}-W_z)+a_z \quad (3)$$

Where:

K_d is the drag coefficient,
C is the ballistic coefficient,
Ω is the air density,
V is the projectile velocity,
W_x & W_z are range wind & cross wind respectively, and
a_x, a_y, a_z are accelerations due to rotation of earth.

These are the assumptions made for the point mass model:

- Projectile is fired at muzzle velocity of 376 m/s only.
- Air density is taken as 1.225 Kg per cubic meter.
- Air Temperature is 15 degree Celsius.
- Accelerations due to rotation of earth are neglected.
- Mach number is equal to 1.1 for these conditions. The drag coefficient and ballistic coefficient of the projectile are non-linear functions of Mach number. During the projectile travel, velocity constantly changes and so these coefficient change [5]. For this study, the coefficients are taken as constant value corresponding to above mentioned Mach number.
- Equations for range (x direction) & height (y direction) are solved. Equation for deviation (z direction) is not considered.
- Target and the point of firing are assumed to be located at same height.

In this study, angle of firing is determined from known range and wind velocity values. Program 1 is used calculate the angle of projectile in vacuum condition. This angle is then put in the program 2 which is used to calculate range of the projectile in point mass model conditions. Then, the angle is incremented by some factor (0.01 in this case) in program 3 until the range of projectile does not equal desired range. In similar fashion, angle of firing is determined for range values from 500 to 7000 meter & wind velocity from 1 to 20 knots. Increment factor for range is taken as 130 meters & the wind velocity is incremented by 1 knot. So, in total angle of firing for 1020 different values are computed and then stored in variable for further process by program 4. These values are used as training data for learning neural network. All the values are converted in standard unit and then used for calculation.

B. Fuzzy logic algorithm

Fuzzy logic can be conceptualized as a generalization of classical logic. Modern fuzzy logic was developed by Lotfi Zadeh in the mid-1960s to model those problems in which imprecise data must be used or in which the rules of inference are formulated in a very general way making use of diffuse categories. In fuzzy logic, which is also sometimes called diffuse logic, there are not just two alternatives but a whole continuum of truth values for logical propositions. A proposition can have the truth value 0.4 and its complement the truth value 0.5. According to the type of negation operator that is used, the two truth values must not be necessarily add up to 1. Fuzzy logic is weakly connected to probability theory. It is strongly connected to set theory & geometry. Comparisons like young & old, tall & short can be easily made using fuzzy logic rules and operators.

Operation on fuzzy sets & rules constructed from them are defined mathematically [6]. It can be seen that union of two membership functions is given by maximum value of the function & intersection of those functions is provided by minimum value of the function. AND operation is defined as minimum value while OR is defined as maximum value. Simple conditional rules like IF-THEN are constructed to form fuzzy logic system.

In this case a fuzzy logic system is made using 71 IF-THEN rules which are constructed using this logic: IF range of the target is certain value AND wind velocity is certain value THEN angle of firing is this value. Mamdani type of system is

constructed & centroid type of defuzzification is selected. The values obtained from trajectory computation in above mentioned section are taken to construct the rules for the fuzzy logic system. Range values used are from 500 to 7000 meter with increment factor of 500 meter. Wind velocities are taken as 1, 5, 10, 15 & 20 knots. Only one case of range 3500 meters & tail wind velocity of 5 knots is taken for analysis. Fuzzy logic toolbox of Matlab software is used for computation. Table 1 is constructed for this case. Program 7 is used to create fuzzy system.

Table 1: Database for fuzzy logic rules

Wind / Range	1 knot	5 knots	10 knots	15 knots	20 knots
500 m	1.044	1.044	1.044	1.044	1.044
1000 m	2.159	2.159	2.159	2.159	2.159
1500 m	3.377	3.377	3.377	3.367	3.367
2000 m	4.688	4.688	4.688	4.678	4.668
2500 m	6.114	6.114	6.114	6.104	6.094
3000 m	7.687	7.677	7.667	7.667	7.647
3500 m	9.427	9.407	9.387	9.367	9.357
4000 m	11.347	11.327	11.297	11.267	11.257
4500 m	13.507	13.477	13.447	13.407	13.367
5000 m	15.980	15.930	15.880	15.830	15.780
5500 m	18.877	18.807	18.727	18.657	18.577
6000 m	22.411	22.321	22.191	22.081	21.961
6500 m	27.184	27.004	26.774	26.564	26.354
7000 m	37.510	36.250	35.190	34.390	33.710

C. Neural network algorithm

Neural network, which is more properly called as artificial neural network (ANN) is defined by Dr. Robert Hecht-Nielsen as: a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs [7]. Neural networks are typically organized in layers. Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. The hidden layers is then linked to an 'output layer' which contains the answer. Mathematically neural network output (y) is represented as [8]:

$$y = \sigma \left(\sum_j w_j a_j + b\right) \quad (4)$$

Where:

σ is the activation function,
b is the bias, and
$\sum_j w_j a_j$ is weighted combination output from hidden neurons.

Most ANNs contain some form of 'learning rule' which is used to modify the weights of the connections according to the input patterns that it is presented with. In a sense, ANNs learn by example as do their biological counterparts; a child learns to recognize dogs from examples of dogs. Although there are many different kinds of learning rules used by neural networks, the rule we are concerned with is the delta rule [9]. The delta rule is often utilized by the most common class of ANNs called 'backpropagation neural networks' (BPNNs). Backpropagation is an abbreviation for the backwards propagation of error. With the delta rule, as with other types of backpropagation, 'learning' is a supervised process that occurs with each cycle or 'epoch' (i.e. each time the network is presented with a new input pattern) through a forward activation flow of outputs, and the backwards error propagation of weight adjustments. More simply, when a neural network is initially presented with a pattern it makes a random 'guess' as to what it might be. It then sees how far its answer was from the actual one and makes an appropriate adjustment to its connection weights. Note also, that within each hidden layer node is a sigmoidal activation function which polarizes network activity and helps it to stabilize. Gradient descent is performed by backpropagation within the solution's vector space towards a 'global minimum' along the steepest vector of the error surface. The global minimum is that theoretical solution with the lowest possible error. The error surface itself is a hyper paraboloid but is seldom 'smooth'. Indeed, in most problems, the solution space is quite irregular with numerous 'pits' and 'hills' which may cause the network to settle down in a 'local minimum' which is not the best overall solution.

Since the nature of the error space cannot be known prior, neural network analysis often requires a large number of individual runs to determine the best solution. Most learning rules have built-in mathematical terms to assist in this process which control the 'speed' (Beta-coefficient) and the 'momentum' of the learning. The speed of learning is actually the rate of convergence between the current solution and the global minimum. Momentum helps the network to overcome obstacles (local minima) in the error surface and settle down at or near the global minimum.

Once a neural network is 'trained' to a satisfactory level it may be used as an analytical tool on other data. To do this, the training runs are no longer specified by user and instead the network is allowed to work in forward propagation mode only. New inputs are presented to the input pattern where they filter into and are processed by the middle layers as though training were taking place, however, at this point the output is retained and no backpropagation occurs. The output of a forward propagation run is the predicted model for the data which can then be used for further analysis and interpretation. Gradient descent method is used by backpropagation algorithm. In this method the derivative of the squared error function with respect to the weights of the network is calculated [10]. Assuming one output neuron, the squared error function is given as:

$$E = (1/2)(t-y)^2 \quad (5)$$

Where:

E is the squared error,
t is the target output for a training sample, and
y is the actual output of the output neuron.

To update the weight Wij using gradient descent, one must choose a learning rate, α. The change in weight, which is added to the old weight, is equal to the product of the learning rate and the gradient, multiplied by -1:

$$\Delta W_{ij} = (-\alpha)(\partial E/\partial W_{ij}) \quad (6)$$

In this study, the values stored in the variable mentioned in earlier section are used as input data and target data to train the neural network. A feedforward backpropagation neural network is designed with following properties:

- The network is composed of 3 hidden layers consisting of 50, 20 & 10 neurons respectively. Input vector has range and wind velocity data. Angle of firing is provided by output layer containing 1 neuron.
- Activation function used for each layer is tangent sigmoid.
- Training algorithm used is Levenberg-Marquardt backpropagation (trainlm) [11].
- Performance function used is mean squared normalized error (mse).
- Backpropagation weight/bias learning function implemented is gradient descent with momentum weight and bias learning function (learngdm).
- Epochs used for training are 2000.
- All other training specifications are kept as given by trainlm function in Matlab software.

Program 5 is programmed to design & train the neural network (Fig. 6). Program 6 is used to create a 3D plot of angle of firing as a function of range & wind velocity (Fig. 9) & Error in the neural network output (Fig. 10). This is based on trajectory computation done earlier. The case of range of target equal to 3500 meters & wind velocity of 5 knots is analysed.

III. RESULTS & DISCUSSIONS

A. Firing table result

Firing table is essentially constructed by compiling data from thousands of tests conducted under various conditions. Final product obtained is set of tables, each containing factor along with the correction required to make. Since point mass model has been implemented in this study, table F for HE M107 projectile is used for analysis (Fig. 5 & Fig. 6). It should be noted that projectile HE M107 is very similar to HE M101, except minor differences. The table is for 5G charge, standard firing conditions & a muzzle velocity of 376 m/s. Analysis based on this projectile was made as firing table for particular projectile was not available.

1	2	3	4	5	6	7	8	9
RANGE	ELEV	FS FOR GRAZE BURST FUZE M564	DFS PER 10 M DEC HOB	DR PER 1 MIL D ELEV	FORK	TIME OF FLIGHT	AZIMUTH CORRECTIONS	
							DRIFT (CORR TO L)	CW OF 1 KNOT
M	MIL			M	MIL	SEC	MIL	MIL
3500	155.3	10.8	0.19	19	2	10.9	3.0	0.22
3600	160.6	11.1	0.18	19	2	11.2	3.2	0.22
3700	165.9	11.5	0.18	19	2	11.6	3.3	0.23
3800	171.2	11.8	0.17	19	2	11.9	3.4	0.23
3900	176.6	12.2	0.17	18	2	12.3	3.5	0.24
4000	182.1	12.5	0.16	18	2	12.6	3.6	0.24
4100	187.6	12.9	0.16	18	3	13.0	3.7	0.24
4200	193.1	13.3	0.16	18	3	13.3	3.8	0.25
4300	198.7	13.6	0.15	18	3	13.7	3.9	0.25
4400	204.3	14.0	0.15	18	3	14.1	4.1	0.26
4500	210.0	14.3	0.14	18	3	14.4	4.2	0.26
4600	215.7	14.7	0.14	17	3	14.8	4.3	0.26
4700	221.5	15.1	0.14	17	3	15.2	4.4	0.27
4800	227.3	15.5	0.13	17	3	15.5	4.6	0.27
4900	233.2	15.8	0.13	17	3	15.9	4.7	0.27
5000	239.2	16.2	0.13	17	3	16.3	4.8	0.28

Fig. 2. Firing table F for HE M107 shell

1	10	11	12	13	14	15	16	17	18	19
RANGE	RANGE CORRECTIONS FOR									
	MUZZLE VELOCITY 1 M/S		RANGE WIND 1 KNOT		AIR TEMP 1 PCT		AIR DENSITY 1 PCT		PROJ WT OF 1 SD (4 SD STD)	
	DEC	INC	HEAD	TAIL	DEC	INC	DEC	INC	DEC	INC
M	M	M	M	M	M	M	M	M	M	M
3500	11.6	-11.9	3.5	-3.9	7.0	-9.1	-4.9	4.7	-20	19
3600	11.8	-12.0	3.8	-4.1	7.5	-9.5	-5.1	4.8	-20	19
3700	12.0	-12.2	4.0	-4.3	8.0	-10.0	-5.3	5.0	-20	19
3800	12.2	-12.4	4.2	-4.4	8.5	-10.5	-5.5	5.2	-20	19
3900	12.3	-12.5	4.4	-4.6	9.0	-11.0	-5.7	5.4	-20	19
4000	12.5	-12.7	4.6	-4.8	9.6	-11.4	-5.8	5.6	-21	19
4100	12.6	-12.9	4.9	-5.0	10.1	-11.9	-6.0	5.7	-21	19
4200	12.8	-13.0	5.1	-5.2	10.7	-12.4	-6.2	5.9	-21	20
4300	13.0	-13.2	5.3	-5.4	11.2	-12.9	-6.4	6.1	-21	20
4400	13.1	-13.3	5.5	-5.6	11.7	-13.3	-6.6	6.3	-21	20
4500	13.3	-13.5	5.8	-5.8	12.3	-13.8	-6.9	6.5	-21	20
4600	13.4	-13.6	6.0	-6.0	12.8	-14.3	-7.1	6.7	-21	20
4700	13.6	-13.8	6.2	-6.2	13.4	-14.8	-7.3	7.0	-21	20
4800	13.7	-13.9	6.5	-6.4	13.9	-15.2	-7.5	7.2	-21	20
4900	13.9	-14.1	6.7	-6.6	14.5	-15.7	-7.7	7.4	-21	20
5000	14.0	-14.2	6.9	-6.7	15.0	-16.2	-7.9	7.6	-21	20

Fig. 3. Firing table F continued

From Fig. 2 & Fig. 3 it is noted that for range of 3500 meter, elevation is 155.3 mil. As tail wind velocity of 5 knots is present, correction of (-3.9 *5) is made. The effective range considered is 3480 meter (range is taken to nearest 10 meter) for this case. To achieve desired range of 3500 meter, 20 meter is added to the effective range. In order to achieve this range,

elevation of 1 mil is added so that range correction of 19 meter is made (column 5 from Fig. 2). So, elevation obtained is 155.3 plus 1 mil which comes out to 156.3*0.05628= 8.8 degree.

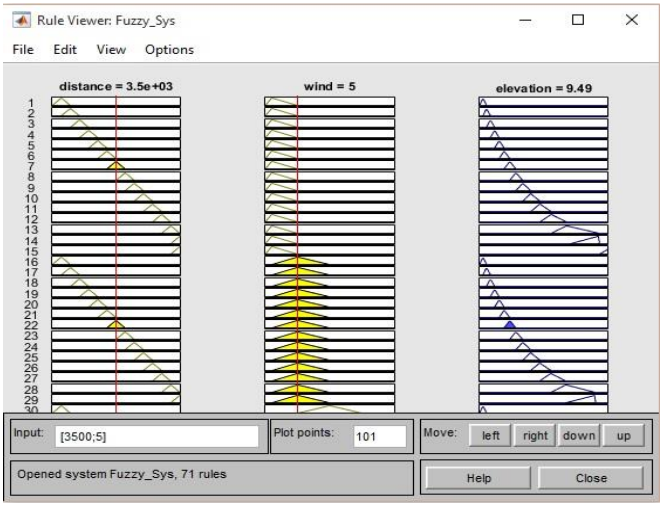

Fig. 4. Output of fuzzy system

B. Fuzzy logic algorithm result

Algorithm was constructed using fuzzy logic toolbox of Matlab software. For this case, inputs to the system are given as 3500 range & 5 knots of tail wind velocity. Angle of firing obtained from the fuzzy system is 9.49 degree (Fig. 4). Also, a 3D plot displaying elevation as a function of range & tail wind velocity is plotted (Fig. 5).

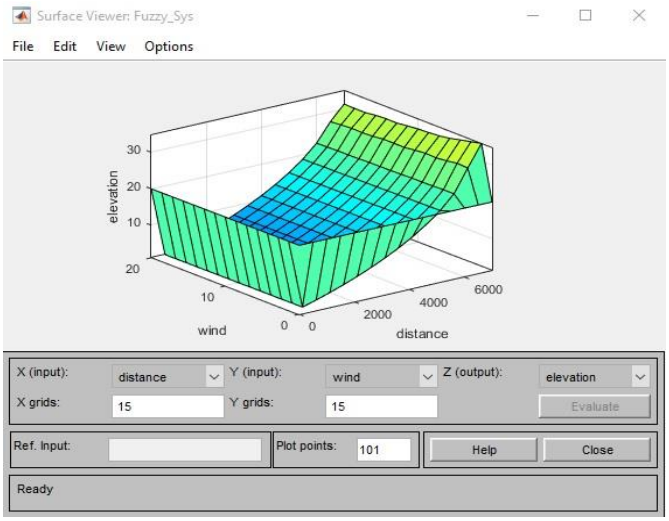

Fig. 5. Elevation vs range & wind 3D plot

C. Neural network algorithm result

A neural network mentioned in section II C is designed and trained (Fig. 6). The trained network is given input as 3500 meter range & tail wind velocity of 5 knots. Output obtained from the system is 9.4006 degree. Performance plots are also obtained (Fig. 7 & Fig, 8). Also, a 3D plot displaying elevation as function of range & wind is plotted (Fig. 9).

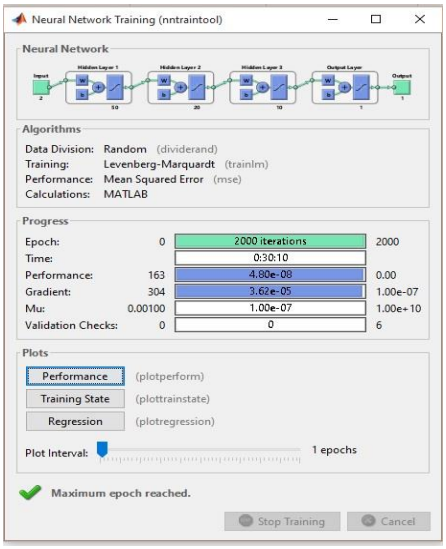

Fig. 6. Neural network

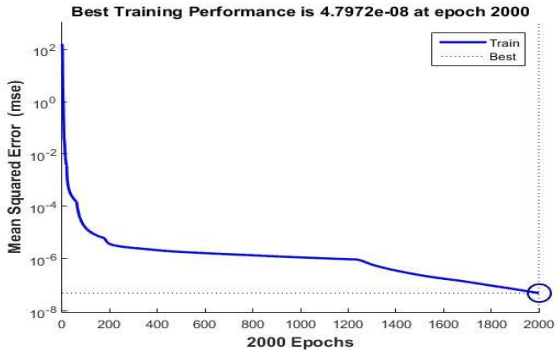

Fig. 7. Performance of network by epochs

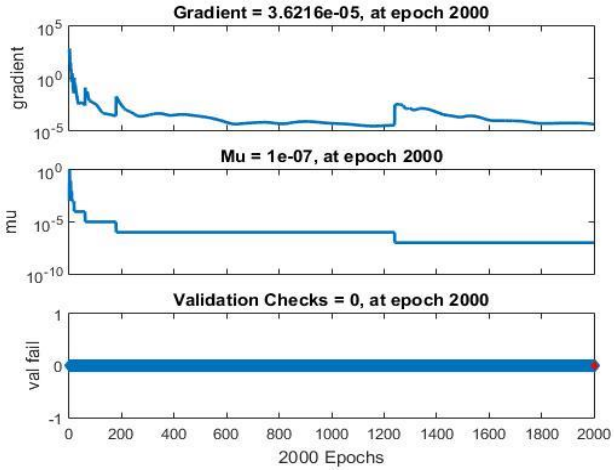

Fig. 8. Training states of network

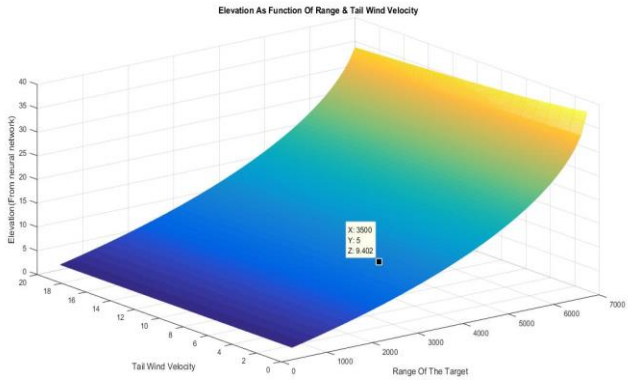

Fig. 9. Elevation vs range & wind

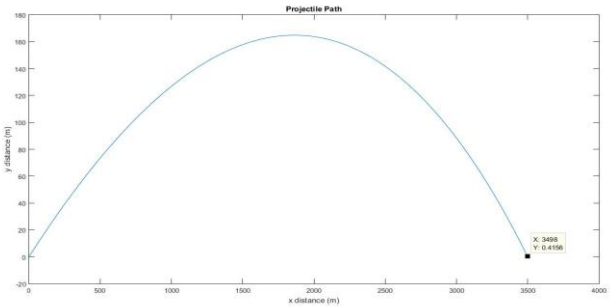

Fig. 10. Trajectory at 9.4 degrees

IV. CONCLUSIONS

From section III it is noted that the elevation extracted from firing table is 8.8 degree. Firing angles of 9.49 & 9.4 degree are given by fuzzy logic algorithm & neural network algorithm respectively. Also, angle of firing obtained from projectile motion equations is 9.4 degree (Fig. 10). Following are the possible reasons for difference between actual & computed value:

- This study implements point mass model for trajectory computation. Many significant forces like Lift force, Magnus force, Coriolis force are not included in this model. Modified point mass model or six degree of freedom model can increase the accuracy.
- Drag coefficient & ballistic coefficient for any projectile non-linear functions of Mach number. During projectile travel Mach number varies as velocity changes. This results in constant variation in coefficient values. For this study, coefficients are taken constant values for the sake of simplicity. Dynamic coefficients can improve the accuracy.

It can be concluded that both the algorithms provide satisfactory results when compared to data from projectile motion. Error between neural network output & actual target is found to be satisfactory (Fig. 11). Trajectory computation can be improved with above mentioned models. Use of fuzzy logic and neural network depend upon applications. For this study, neural network is preferred as the data from trajectory computation is huge, making the training of network straightforward. As the data increases, if-then rules required to construct increase.

For this study, a case with only range and wind velocity variations is considered. Many other factors can be included in the algorithm. With the required data, this algorithm can be used to compute angle of firing for any projectile in very less duration, saving valuable time in the war.

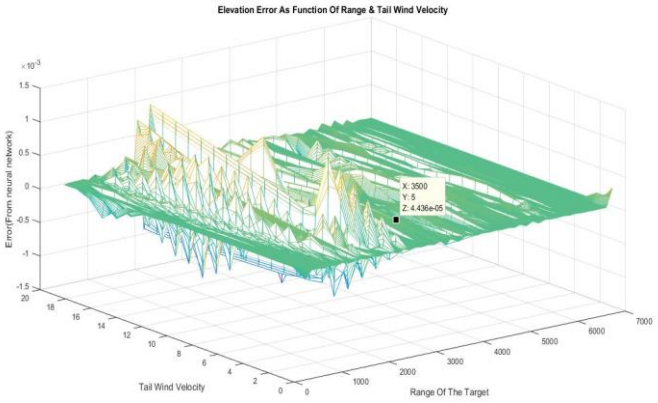

Fig. 11. Error in neural network output

REFERENCES

[1] Elizabeth R. Dickinson. "Analysis of raw data" in The Production Of Firing Tables For Cannon Artillery, report no. 1371, November 1967, pp. 21-23.

[2] Elizabeth R. Dickinson. "Computations for the tabular firing table" in The Production Of Firing Tables For Cannon Artillery, report no. 1371, November 1967, pp. 50-72.

[3] "Chapter7:FiringTables"Internet:http://www.globalsecurity.org/military/library/policy/army/fm/6-40/Ch7.htm[Dec. 2, 2015]

[4] Col.Asst.Dr.Phaderm Nangsue. "Generation of Artillery Firing Tables for The L119 Howitzer with Base-Bleed Projectiles". Journal of the Faculty Senate, CRMA, Vol. 8, pp. 127-135, 2010

[5] Charles T. Odom. A drag coefficient, Kd based on the 155MM shell HE M101, memorandum report no. 1167, September 1958.

[6] S.N. Sivanandam, S.Sumathi & S.N. Deepa "Fuzzy Rule-Based System" in Introduction to Fuzzy Logic using MatLab, Springer-Verlag Berlin Heidelberg, 2007, pp. 113-149

[7] "A Basic Introduction To Neural Networks" Internet: http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html[Dec. 2, 2015]

[8] "Chapter 4: A visual proof that neural nets can compute any function"Internet:http://neuralnetworksanddeeplearning,com/chap4.html[Dec. 2, 2015]

[9] "Delta rule" Internet: https://en.wikipedia.org/wiki/Delta_rule [Dec. 2, 2015]

[10] "Backpropagation"Internet:https://en.wikipedia.org/wiki/Backpropagation[Dec. 2, 2015]

[11] "LevenbergMarquardtalgorithm"Internet:https://en.wikipedia.org/wiki/Levenberg%E2%80%93Marquardt_algorithm[Dec.2, 2015]

APPENDIX

Matlab Codes:

Programs 1,2,3,4:

```
disp('## Final Program For Discrete Neural Network Project ##');

for n=1:1:20;  % Wind variations will be from 1 to 20 knots with 1 knot
increment factor

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%$$$$$

disp('## Program 1: Range & Vacuum angle Finder ##');

range=(500:130:7000);                          % range variations will be
from 500 to 7000 meters with 130 meters increment factor
theta=(0.5)*asind((range)*(9.81)/(376^2));
range1=[range];
theta1=[theta];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%$$$$$

disp('## Program 2: For Range in Air & Wind Conditions With Vacuum
Conditions Angle ##');

%Constants for 155mm shell

c=.152;           %Drag Coefficient of a 155mm shell
rho= 1.288;       %kg/m^3 (density of air)
g=9.81;           %m/s^2 (acceleration due to gravity)

%Initial Conditions



theta2=theta1;
theta3=theta2;
q=length(range1);

wind(1:q)=n;
wind1{n}=wind;
desired_range{n}=range1;

for k = 1:1:q;

delta_t= .01;  %s
x(1)=0  ;
y(1)=0;
```

```
V=376;         % m/s

u=V*cosd(theta2(1,k));
v=V*sind(theta2(1,k));
x=0;
y=0;



t(1)=0;        %Start Loop
i=1;
while min(y)> -.001;
ax=-(rho*V*c)/1736.42*(u-(0.5*wind(1,k)));     %  ax=-(rho*V*c)/1362.17*(u-
(0.5*(wind velocity in knots)));
ay=(-(rho*V*c)/1736.42)*v-g;
u=u+ax*delta_t;
v=v+ay*delta_t;
V=sqrt(u^2+v^2);
x(i+1)=x(i)+u*delta_t+.5*ax*delta_t^2;
y(i+1)=y(i)+v*delta_t+.5*ay*delta_t^2;
t(i+1)=t(i)+delta_t;
i=i+1;
end

realrange=x(1,i);
realrange1(1,k)=realrange;
realrange2=realrange1;

clear realrange;
clear u v x y t ax ay i V;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

disp('## Program 3: For Angle In Air & Wind Conditions ##');


for L=1:1:q;




    m=1;
    while le(realrange2(1,L),range1(1,L));

    theta3(1,L)=theta3(1,L)+0.01;                    % Angle increment
factor which decides the accuracy required and computation time.

    delta_t= .01;  %s
    x(1)=0  ;
    y(1)=0;
    V=376;         % m/s

    u=V*cosd(theta3(1,L));
    v=V*sind(theta3(1,L));
    x=0;
```

```
    y=0;

    t(1)=0;         %Start Loop

      i=1;
      while min(y)> -.001;

      ax=-(rho*V*c)/1736.42*(u-(0.5*wind(1,L)));      %  ax=-
(rho*V*c)/1362.17*(u-(0.5*(wind velocity in knots)));
      ay=(-(rho*V*c)/1736.42)*v-g;
      u=u+ax*delta_t;
      v=v+ay*delta_t;
      V=sqrt(u^2+v^2);
      x(i+1)=x(i)+u*delta_t+.5*ax*delta_t^2;
      y(i+1)=y(i)+v*delta_t+.5*ay*delta_t^2;
      t(i+1)=t(i)+delta_t;
      i=i+1;

      end

    realrange3=x(1,i);
    realrange4(1,m)=realrange3;
    realrange2(1,L)=realrange3;
    theta4(1,m)=theta3(1,L);

    clear realrange3 x y u v t ax ay i V;
    m=m+1;

    end

clear m;
realrange5(1,L)=max(realrange4);
theta5(1,L)=max(theta4);


clear realrange4 theta4;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

realrange6{n}=realrange5;
theta6{n}=theta5;
clear wind realrange5 theta5;

end

wind2=cell2mat(wind1);
realrange7=cell2mat(realrange6);
theta7=cell2mat(theta6);
desired_range1=cell2mat(desired_range);

save('input1_range','realrange7');
save('input2_wind','wind2');
save('output_angle','theta7');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

disp('## Program 4: Creating Data Vectors For Training the neural network
##');

b=length(theta7);   % Total number of trajectories computed is stored in
the variable b

for j=1:1:b;

  X(1,j)={[realrange7(1,j);wind2(1,j)]};    % For arranging input data in
specific format
  T(1,j)={theta7(1,j)};                     % For arranging output data in
specific format

end

save('input_vector','X');
save('output_vector','T');
clear all;
load('input_vector.mat');
load('output_vector.mat');

disp(' X is the input vector containing range and wind data while T is the
output vector containing angle of firing ');
```

Program 5:

```
disp(' ## Program 5: For Learning Network ## ');

% Initiating Network For Learning.
% This is a feed forward backpropagation network.
% The network has 3 layers with 50,20 & 1 neurons respectively, each
% consisting of tansig as activation function.
% Backprop network training function is trainlm.
% Backprop weight/bias learning function is learngdm.
% Performance function is mse.

load('input_vector');
load('output_vector');

net = newff(X,T,[50 20 10],{'tansig' 'tansig' 'tansig'
'tansig'},'trainlm','learngdm','mse');
net=configure(net,X,T);
net.trainParam.epochs=2000;
net=train(net,X,T);
```

Program 6:

```
disp('## Program 6: For creating 3D plot of the data ##');

% Range,Wind Velocity & Angle Of Firing is ploted

load('input1_range');
load('input2_wind');
load('network3');
load('input_vector');
load('output_vector');

network_angle=sim(net,X);
network_angle=cell2mat(network_angle);

[e,f] = meshgrid(500:5:7000,1:1:20);
z = griddata(realrange7(1,:),wind2(1,:),network_angle,e,f);
mesh(e,f,z);
hold on;
plot3(e,f,z);
xlabel('Range Of The Target');
ylabel('Tail Wind Velocity');
zlabel('Elevation(From neural network)');
title('Elevation As Function Of Range & Tail Wind Velocity');
figure;

T=cell2mat(T);
error=T-network_angle;

y = griddata(realrange7(1,:),wind2(1,:),error,e,f);
mesh(e,f,y);
hold on;
plot3(e,f,y);
xlabel('Range Of The Target');
ylabel('Tail Wind Velocity');
zlabel('Error(From neural network)');
title('Elevation Error As Function Of Range & Tail Wind Velocity');
```

```
disp('## Program 7: Fuzzy Logic System');

fsys = newfis('Projectile');

% For 1st Input(Target Range):

fsys.input(1).name = 'distance';
fsys.input(1).range = [0 7000];
fsys.input(1).mf(1).name = '500';fsys.input(1).mf(1).type =
'trimf';fsys.input(1).mf(1).params = [0 500 1000];
fsys.input(1).mf(2).name = '1000';fsys.input(1).mf(2).type =
'trimf';fsys.input(1).mf(2).params = [500  1000 1500];
fsys.input(1).mf(3).name = '1500';fsys.input(1).mf(3).type =
'trimf';fsys.input(1).mf(3).params = [1000 1500 2000];
fsys.input(1).mf(4).name = '2000';fsys.input(1).mf(4).type =
'trimf';fsys.input(1).mf(4).params = [1500 2000 2500];
fsys.input(1).mf(5).name = '2500';fsys.input(1).mf(5).type =
'trimf';fsys.input(1).mf(5).params = [2000 2500 3000];
fsys.input(1).mf(6).name = '3000';fsys.input(1).mf(6).type =
'trimf';fsys.input(1).mf(6).params = [2500 3000 3500];
fsys.input(1).mf(7).name = '3500';fsys.input(1).mf(7).type =
'trimf';fsys.input(1).mf(7).params = [3000 3500 4000];
fsys.input(1).mf(8).name = '4000';fsys.input(1).mf(8).type =
'trimf';fsys.input(1).mf(8).params = [3500 4000 4500];
fsys.input(1).mf(9).name = '4500';fsys.input(1).mf(9).type =
'trimf';fsys.input(1).mf(9).params = [4000 4500 5000];
fsys.input(1).mf(10).name = '5000';fsys.input(1).mf(10).type =
'trimf';fsys.input(1).mf(10).params = [4500 5000 5500];
fsys.input(1).mf(11).name = '5500';fsys.input(1).mf(11).type =
'trimf';fsys.input(1).mf(11).params = [5000 5500 6000];
fsys.input(1).mf(12).name = '6000';fsys.input(1).mf(12).type =
'trimf';fsys.input(1).mf(12).params = [5500 6000 6500];
fsys.input(1).mf(13).name = '6500';fsys.input(1).mf(13).type =
'trimf';fsys.input(1).mf(13).params = [6000 6500 7000];
fsys.input(1).mf(14).name = '7000';fsys.input(1).mf(14).type =
'trimf';fsys.input(1).mf(14).params = [6500 7000 7000];

% For 2nd Input(Wind Velocity):

fsys.input(2).name = 'wind';
fsys.input(2).range = [0 20];
fsys.input(2).mf(1).name = '1knot';fsys.input(2).mf(1).type =
'trimf';fsys.input(2).mf(1).params = [0 1 5];
fsys.input(2).mf(2).name = '5knots';fsys.input(2).mf(2).type =
'trimf';fsys.input(2).mf(2).params = [1 5 10];
fsys.input(2).mf(3).name = '10knots';fsys.input(2).mf(3).type =
'trimf';fsys.input(2).mf(3).params = [5 10 15];
fsys.input(2).mf(4).name = '15knots';fsys.input(2).mf(4).type =
'trimf';fsys.input(2).mf(4).params = [10 15 20];
fsys.input(2).mf(5).name = '20knots';fsys.input(2).mf(5).type =
'trimf';fsys.input(2).mf(5).params = [15 20 20];

% For Output(Angle Of Firing):

fsys.output(1).name = 'elevation';
fsys.output(1).range = [0 40];
fsys.output(1).mf(1).name = '1.044';fsys.output(1).mf(1).type =
'trimf';fsys.output(1).mf(1).params = [0 1.044 2.159];
fsys.output(1).mf(2).name = '2.159';fsys.output(1).mf(2).type =
'trimf';fsys.output(1).mf(2).params = [1.044 2.159 3.377];
```

```
fsys.output(1).mf(3).name = '3.377';fsys.output(1).mf(3).type =
'trimf';fsys.output(1).mf(3).params = [2.159 3.377 4.688];
fsys.output(1).mf(4).name = '4.688';fsys.output(1).mf(4).type =
'trimf';fsys.output(1).mf(4).params = [3.377 4.688 6.114];
fsys.output(1).mf(5).name = '6.114';fsys.output(1).mf(5).type =
'trimf';fsys.output(1).mf(5).params = [4.688 6.114 7.687];
fsys.output(1).mf(6).name = '7.687';fsys.output(1).mf(6).type =
'trimf';fsys.output(1).mf(6).params = [6.114 7.687 9.427];
fsys.output(1).mf(7).name = '9.427';fsys.output(1).mf(7).type =
'trimf';fsys.output(1).mf(7).params = [7.687 9.427 11.347];
fsys.output(1).mf(8).name = '11.347';fsys.output(1).mf(8).type =
'trimf';fsys.output(1).mf(8).params = [9.427 11.347 13.507];
fsys.output(1).mf(9).name = '13.507';fsys.output(1).mf(9).type =
'trimf';fsys.output(1).mf(9).params = [11.347 13.507 15.980];
fsys.output(1).mf(10).name = '15.980';fsys.output(1).mf(10).type =
'trimf';fsys.output(1).mf(10).params = [13.507 15.980 18.877];
fsys.output(1).mf(11).name = '18.877';fsys.output(1).mf(11).type =
'trimf';fsys.output(1).mf(11).params = [15.980 18.877 22.411];
fsys.output(1).mf(12).name = '22.411';fsys.output(1).mf(12).type =
'trimf';fsys.output(1).mf(12).params = [18.877 22.411 27.184];
fsys.output(1).mf(13).name = '27.184';fsys.output(1).mf(13).type =
'trimf';fsys.output(1).mf(13).params = [22.411 27.184 37.510];
fsys.output(1).mf(14).name = '37.510';fsys.output(1).mf(14).type =
'trimf';fsys.output(1).mf(14).params = [28.344 37.220 37.220];
fsys.output(1).mf(15).name = 'mf15';fsys.output(1).mf(15).type =
'trimf';fsys.output(1).mf(15).params = [37.603 41.623 47.123];
fsys.output(1).mf(16).name = '5knots_5500';fsys.output(1).mf(16).type =
'trimf';fsys.output(1).mf(16).params = [15.930 18.807 22.321];
fsys.output(1).mf(17).name = '5knots_6000';fsys.output(1).mf(17).type =
'trimf';fsys.output(1).mf(17).params = [18.807 22.321 27.004];
fsys.output(1).mf(18).name = '5knots_6500';fsys.output(1).mf(18).type =
'trimf';fsys.output(1).mf(18).params = [22.321 27.004 36.250];
fsys.output(1).mf(19).name = '5knots_7000';fsys.output(1).mf(19).type =
'trimf';fsys.output(1).mf(19).params = [27.004 36.250 36.250];
fsys.output(1).mf(20).name = '10knots_2500';fsys.output(1).mf(20).type =
'trimf';fsys.output(1).mf(20).params = [4.688 6.114 7.667];
fsys.output(1).mf(21).name = '10knots_3000';fsys.output(1).mf(21).type =
'trimf';fsys.output(1).mf(21).params = [6.114 7.667 9.387];
fsys.output(1).mf(22).name = '10knots_3500';fsys.output(1).mf(22).type =
'trimf';fsys.output(1).mf(22).params = [7.667 9.387 11.297];
fsys.output(1).mf(23).name = '10knots_4000';fsys.output(1).mf(23).type =
'trimf';fsys.output(1).mf(23).params = [9.387 11.297 13.447];
fsys.output(1).mf(24).name = '10knots_4500';fsys.output(1).mf(24).type =
'trimf';fsys.output(1).mf(24).params = [11.297 13.447 15.880];
fsys.output(1).mf(25).name = '10knots_5000';fsys.output(1).mf(25).type =
'trimf';fsys.output(1).mf(25).params = [13.447 15.880 18.727];
fsys.output(1).mf(26).name = '10knots_5500';fsys.output(1).mf(26).type =
'trimf';fsys.output(1).mf(26).params = [15.880 18.727 22.191];
fsys.output(1).mf(27).name = '10knots_6000';fsys.output(1).mf(27).type =
'trimf';fsys.output(1).mf(27).params = [18.727 22.191 26.774];
fsys.output(1).mf(28).name = '10knots_6000';fsys.output(1).mf(28).type =
'trimf';fsys.output(1).mf(28).params = [18.727 22.191 26.774];
fsys.output(1).mf(29).name = '10knots_7000';fsys.output(1).mf(29).type =
'trimf';fsys.output(1).mf(29).params = [26.774 35.190 35.190];
fsys.output(1).mf(30).name = '10knots_6500';fsys.output(1).mf(30).type =
'trimf';fsys.output(1).mf(30).params = [22.191 26.774 35.190];
fsys.output(1).mf(31).name = '15knots_2000';fsys.output(1).mf(31).type =
'trimf';fsys.output(1).mf(31).params = [3.367 4.678 6.104];
fsys.output(1).mf(32).name = '15knots_2500';fsys.output(1).mf(32).type =
'trimf';fsys.output(1).mf(32).params = [4.678 6.104 7.667];
```

```
fsys.output(1).mf(33).name = '15knots_3000';fsys.output(1).mf(33).type =
'trimf';fsys.output(1).mf(33).params = [6.104 7.667 9.367];
fsys.output(1).mf(34).name = '15knots_3500';fsys.output(1).mf(34).type =
'trimf';fsys.output(1).mf(34).params = [7.667 9.367 11.267];
fsys.output(1).mf(35).name = '15knots_4000';fsys.output(1).mf(35).type =
'trimf';fsys.output(1).mf(35).params = [9.367 11.267 13.407];
fsys.output(1).mf(36).name = '15knots_4500';fsys.output(1).mf(36).type =
'trimf';fsys.output(1).mf(36).params = [11.267 13.407 15.830];
fsys.output(1).mf(37).name = '15knots_5000';fsys.output(1).mf(37).type =
'trimf';fsys.output(1).mf(37).params = [13.407 15.830 18.657];
fsys.output(1).mf(38).name = '15knots_5500';fsys.output(1).mf(38).type =
'trimf';fsys.output(1).mf(38).params = [15.830 18.657 22.081];
fsys.output(1).mf(39).name = '15knots_6000';fsys.output(1).mf(39).type =
'trimf';fsys.output(1).mf(39).params = [18.657 22.081 26.564];
fsys.output(1).mf(40).name = '15knots_6500';fsys.output(1).mf(40).type =
'trimf';fsys.output(1).mf(40).params = [22.081 26.564 34.390];
fsys.output(1).mf(41).name = '15knots_7000';fsys.output(1).mf(41).type =
'trimf';fsys.output(1).mf(41).params = [26.564 34.390 34.390];
fsys.output(1).mf(42).name = '20knots_1000';fsys.output(1).mf(42).type =
'trimf';fsys.output(1).mf(42).params = [1.044 2.159 3.367];
fsys.output(1).mf(43).name = '20knots_1500';fsys.output(1).mf(43).type =
'trimf';fsys.output(1).mf(43).params = [2.159 3.367 4.668];
fsys.output(1).mf(44).name = '10knots_2000';fsys.output(1).mf(44).type =
'trimf';fsys.output(1).mf(44).params = [3.377 4.688 6.114];
fsys.output(1).mf(45).name = '10knots_2500';fsys.output(1).mf(45).type =
'trimf';fsys.output(1).mf(45).params = [4.688 6.114 7.667];
fsys.output(1).mf(46).name = '20knots_3000';fsys.output(1).mf(46).type =
'trimf';fsys.output(1).mf(46).params = [6.094 7.647 9.357];
fsys.output(1).mf(47).name = '20knots_3500';fsys.output(1).mf(47).type =
'trimf';fsys.output(1).mf(47).params = [7.647 9.357 11.257];
fsys.output(1).mf(48).name = '20knots_4000';fsys.output(1).mf(48).type =
'trimf';fsys.output(1).mf(48).params = [9.357 11.257 13.367];
fsys.output(1).mf(49).name = '20knots_4500';fsys.output(1).mf(49).type =
'trimf';fsys.output(1).mf(49).params = [11.257 13.367 15.780];
fsys.output(1).mf(50).name = '20knots_5000';fsys.output(1).mf(50).type =
'trimf';fsys.output(1).mf(50).params = [13.367 15.780 18.577];
fsys.output(1).mf(51).name = '20knots_5500';fsys.output(1).mf(51).type =
'trimf';fsys.output(1).mf(51).params = [15.780 18.577 21.961];
fsys.output(1).mf(52).name = '20knots_6000';fsys.output(1).mf(52).type =
'trimf';fsys.output(1).mf(52).params = [18.577 21.961 26.354];
fsys.output(1).mf(53).name = '20knots_6500';fsys.output(1).mf(53).type =
'trimf';fsys.output(1).mf(53).params = [21.961 26.354 33.710];
fsys.output(1).mf(54).name = '20knots_7000';fsys.output(1).mf(54).type =
'trimf';fsys.output(1).mf(54).params = [26.354 33.710 33.710];

% Rules For System:

fsys.rule(1).antecedent = [1 1];fsys.rule(1).consequent =
[1];fsys.rule(1).weight = 1;fsys.rule(1).connection = 1;
fsys.rule(2).antecedent = [2 1];fsys.rule(2).consequent =
[2];fsys.rule(2).weight = 1;fsys.rule(2).connection = 1;
fsys.rule(3).antecedent = [3 1];fsys.rule(3).consequent =
[3];fsys.rule(3).weight = 1;fsys.rule(3).connection = 1;
fsys.rule(4).antecedent = [4 1];fsys.rule(4).consequent =
[4];fsys.rule(4).weight = 1;fsys.rule(4).connection = 1;
fsys.rule(5).antecedent = [5 1];fsys.rule(5).consequent =
[5];fsys.rule(5).weight = 1;fsys.rule(5).connection = 1;
fsys.rule(6).antecedent = [6 1];fsys.rule(6).consequent =
[6];fsys.rule(6).weight = 1;fsys.rule(6).connection = 1;
fsys.rule(7).antecedent = [7 1];fsys.rule(7).consequent =
[7];fsys.rule(7).weight = 1;fsys.rule(7).connection = 1;
```

```
fsys.rule(8).antecedent = [8 1];fsys.rule(8).consequent =
[8];fsys.rule(8).weight = 1;fsys.rule(8).connection = 1;
fsys.rule(9).antecedent = [9 1];fsys.rule(9).consequent =
[9];fsys.rule(9).weight = 1;fsys.rule(9).connection = 1;
fsys.rule(10).antecedent = [10 1];fsys.rule(10).consequent =
[10];fsys.rule(10).weight = 1;fsys.rule(10).connection = 1;
fsys.rule(11).antecedent = [11 1];fsys.rule(11).consequent =
[11];fsys.rule(11).weight = 1;fsys.rule(11).connection = 1;
fsys.rule(12).antecedent = [12 1];fsys.rule(12).consequent =
[12];fsys.rule(12).weight = 1;fsys.rule(12).connection = 1;
fsys.rule(13).antecedent = [13 1];fsys.rule(13).consequent =
[13];fsys.rule(13).weight = 1;fsys.rule(13).connection = 1;
fsys.rule(14).antecedent = [14 1];fsys.rule(14).consequent =
[14];fsys.rule(14).weight = 1;fsys.rule(14).connection = 1;
fsys.rule(15).antecedent = [14 1];fsys.rule(15).consequent =
[15];fsys.rule(15).weight = 1;fsys.rule(15).connection = 1;
fsys.rule(16).antecedent = [1 2];fsys.rule(16).consequent =
[1];fsys.rule(16).weight = 1;fsys.rule(16).connection = 1;
fsys.rule(17).antecedent = [2 2];fsys.rule(17).consequent =
[2];fsys.rule(17).weight = 1;fsys.rule(17).connection = 1;
fsys.rule(18).antecedent = [3 2];fsys.rule(18).consequent =
[3];fsys.rule(18).weight = 1;fsys.rule(18).connection = 1;
fsys.rule(19).antecedent = [4 2];fsys.rule(19).consequent =
[4];fsys.rule(19).weight = 1;fsys.rule(19).connection = 1;
fsys.rule(20).antecedent = [5 2];fsys.rule(20).consequent =
[5];fsys.rule(20).weight = 1;fsys.rule(20).connection = 1;
fsys.rule(21).antecedent = [6 2];fsys.rule(21).consequent =
[6];fsys.rule(21).weight = 1;fsys.rule(21).connection = 1;
fsys.rule(22).antecedent = [7 2];fsys.rule(22).consequent =
[7];fsys.rule(22).weight = 1;fsys.rule(22).connection = 1;
fsys.rule(23).antecedent = [8 2];fsys.rule(23).consequent =
[8];fsys.rule(23).weight = 1;fsys.rule(23).connection = 1;
fsys.rule(24).antecedent = [9 2];fsys.rule(24).consequent =
[9];fsys.rule(24).weight = 1;fsys.rule(24).connection = 1;
fsys.rule(25).antecedent = [10 2];fsys.rule(25).consequent =
[10];fsys.rule(25).weight = 1;fsys.rule(25).connection = 1;
fsys.rule(26).antecedent = [11 2];fsys.rule(26).consequent =
[16];fsys.rule(26).weight = 1;fsys.rule(26).connection = 1;
fsys.rule(27).antecedent = [12 2];fsys.rule(27).consequent =
[17];fsys.rule(27).weight = 1;fsys.rule(27).connection = 1;
fsys.rule(28).antecedent = [13 2];fsys.rule(28).consequent =
[18];fsys.rule(28).weight = 1;fsys.rule(28).connection = 1;
fsys.rule(29).antecedent = [14 2];fsys.rule(29).consequent =
[19];fsys.rule(29).weight = 1;fsys.rule(29).connection = 1;
fsys.rule(30).antecedent = [1 3];fsys.rule(30).consequent =
[1];fsys.rule(30).weight = 1;fsys.rule(30).connection = 1;
fsys.rule(31).antecedent = [2 3];fsys.rule(31).consequent =
[2];fsys.rule(31).weight = 1;fsys.rule(31).connection = 1;
fsys.rule(32).antecedent = [3 3];fsys.rule(32).consequent =
[3];fsys.rule(32).weight = 1;fsys.rule(32).connection = 1;
fsys.rule(33).antecedent = [4 3];fsys.rule(33).consequent =
[4];fsys.rule(33).weight = 1;fsys.rule(33).connection = 1;
fsys.rule(34).antecedent = [5 3];fsys.rule(34).consequent =
[20];fsys.rule(34).weight = 1;fsys.rule(34).connection = 1;
fsys.rule(35).antecedent = [6 3];fsys.rule(35).consequent =
[21];fsys.rule(35).weight = 1;fsys.rule(35).connection = 1;
fsys.rule(36).antecedent = [7 3];fsys.rule(36).consequent =
[22];fsys.rule(36).weight = 1;fsys.rule(36).connection = 1;
fsys.rule(37).antecedent = [8 3];fsys.rule(37).consequent =
[23];fsys.rule(37).weight = 1;fsys.rule(37).connection = 1;
```

```
fsys.rule(38).antecedent = [9 3];fsys.rule(38).consequent =
[24];fsys.rule(38).weight = 1;fsys.rule(38).connection = 1;
fsys.rule(39).antecedent = [10 3];fsys.rule(39).consequent =
[25];fsys.rule(39).weight = 1;fsys.rule(39).connection = 1;
fsys.rule(40).antecedent = [11 3];fsys.rule(40).consequent =
[26];fsys.rule(40).weight = 1;fsys.rule(40).connection = 1;
fsys.rule(41).antecedent = [12 3];fsys.rule(41).consequent =
[27];fsys.rule(41).weight = 1;fsys.rule(41).connection = 1;
fsys.rule(42).antecedent = [13 3];fsys.rule(42).consequent =
[30];fsys.rule(42).weight = 1;fsys.rule(42).connection = 1;
fsys.rule(43).antecedent = [14 3];fsys.rule(43).consequent =
[29];fsys.rule(43).weight = 1;fsys.rule(43).connection = 1;
fsys.rule(44).antecedent = [1 4];fsys.rule(44).consequent =
[1];fsys.rule(44).weight = 1;fsys.rule(44).connection = 1;
fsys.rule(45).antecedent = [2 4];fsys.rule(45).consequent =
[2];fsys.rule(45).weight = 1;fsys.rule(45).connection = 1;
fsys.rule(46).antecedent = [3 4];fsys.rule(46).consequent =
[3];fsys.rule(46).weight = 1;fsys.rule(46).connection = 1;
fsys.rule(47).antecedent = [4 4];fsys.rule(47).consequent =
[31];fsys.rule(47).weight = 1;fsys.rule(47).connection = 1;
fsys.rule(48).antecedent = [5 4];fsys.rule(48).consequent =
[32];fsys.rule(48).weight = 1;fsys.rule(48).connection = 1;
fsys.rule(49).antecedent = [6 4];fsys.rule(49).consequent =
[33];fsys.rule(49).weight = 1;fsys.rule(49).connection = 1;
fsys.rule(50).antecedent = [7 4];fsys.rule(50).consequent =
[34];fsys.rule(50).weight = 1;fsys.rule(50).connection = 1;
fsys.rule(51).antecedent = [8 4];fsys.rule(51).consequent =
[35];fsys.rule(51).weight = 1;fsys.rule(51).connection = 1;
fsys.rule(52).antecedent = [9 4];fsys.rule(52).consequent =
[36];fsys.rule(52).weight = 1;fsys.rule(52).connection = 1;
fsys.rule(53).antecedent = [10 4];fsys.rule(53).consequent =
[37];fsys.rule(53).weight = 1;fsys.rule(53).connection = 1;
fsys.rule(54).antecedent = [11 4];fsys.rule(54).consequent =
[38];fsys.rule(54).weight = 1;fsys.rule(54).connection = 1;
fsys.rule(55).antecedent = [12 4];fsys.rule(55).consequent =
[39];fsys.rule(55).weight = 1;fsys.rule(55).connection = 1;
fsys.rule(56).antecedent = [13 4];fsys.rule(56).consequent =
[40];fsys.rule(56).weight = 1;fsys.rule(56).connection = 1;
fsys.rule(57).antecedent = [14 4];fsys.rule(57).consequent =
[41];fsys.rule(57).weight = 1;fsys.rule(57).connection = 1;
fsys.rule(58).antecedent = [1 5];fsys.rule(58).consequent =
[1];fsys.rule(58).weight = 1;fsys.rule(58).connection = 1;
fsys.rule(59).antecedent = [2 5];fsys.rule(59).consequent =
[42];fsys.rule(59).weight = 1;fsys.rule(59).connection = 1;
fsys.rule(60).antecedent = [3 5];fsys.rule(60).consequent =
[43];fsys.rule(60).weight = 1;fsys.rule(60).connection = 1;
fsys.rule(61).antecedent = [4 5];fsys.rule(61).consequent =
[44];fsys.rule(61).weight = 1;fsys.rule(61).connection = 1;
fsys.rule(62).antecedent = [5 5];fsys.rule(62).consequent =
[45];fsys.rule(62).weight = 1;fsys.rule(62).connection = 1;
fsys.rule(63).antecedent = [6 5];fsys.rule(63).consequent =
[46];fsys.rule(63).weight = 1;fsys.rule(63).connection = 1;
fsys.rule(64).antecedent = [7 5];fsys.rule(64).consequent =
[47];fsys.rule(64).weight = 1;fsys.rule(64).connection = 1;
fsys.rule(65).antecedent = [8 5];fsys.rule(65).consequent =
[48];fsys.rule(65).weight = 1;fsys.rule(65).connection = 1;
fsys.rule(66).antecedent = [9 5];fsys.rule(66).consequent =
[49];fsys.rule(66).weight = 1;fsys.rule(66).connection = 1;
fsys.rule(67).antecedent = [10 5];fsys.rule(67).consequent =
[50];fsys.rule(67).weight = 1;fsys.rule(67).connection = 1;
```

```
fsys.rule(68).antecedent = [11 5];fsys.rule(68).consequent =
[51];fsys.rule(68).weight = 1;fsys.rule(68).connection = 1;
fsys.rule(69).antecedent = [12 5];fsys.rule(69).consequent =
[52];fsys.rule(69).weight = 1;fsys.rule(69).connection = 1;
fsys.rule(70).antecedent = [13 5];fsys.rule(70).consequent =
[53];fsys.rule(70).weight = 1;fsys.rule(70).connection = 1;
fsys.rule(71).antecedent = [14 5];fsys.rule(71).consequent =
[54];fsys.rule(71).weight = 1;fsys.rule(71).connection = 1;

writefis(fsys,'Fuzzy_Sys');  % Creates FIS file.
```