INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

DEPARTMENT OF MECHANICAL ENGINEERING

ME 674 SOFT COMPUTING

ASSIGNMENT 1: ARTIFICIAL NEURAL NETWORK

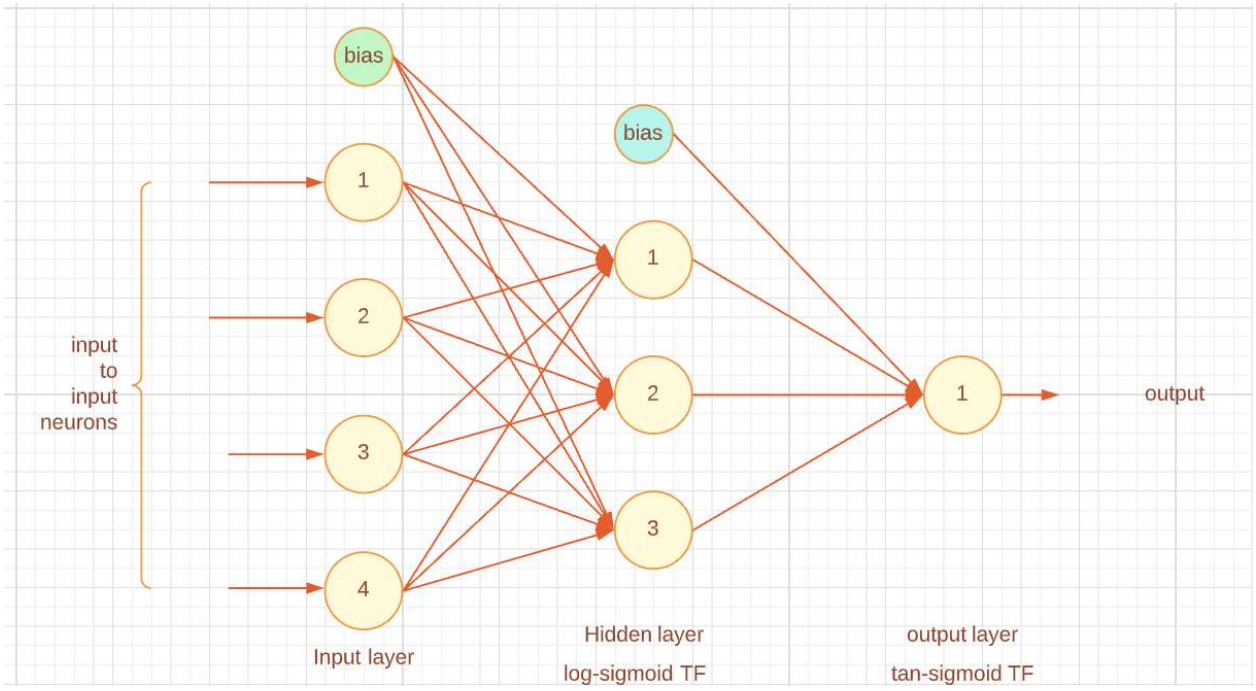
SUBMITTED BY: SONU PRIYA

ROLL NO: 214103323

SUBJECT INSTRUCTOR: Prof. SUKHOMAY PAL

CONTENTS:

FLOW CHART:



Let us consider no. of input neurons is 3, no. of hidden neurons is 3 and output neurons is 1.

TRAINING OF ARTIFICIAL NEURAL NETWORK

Let us consider 4 input neurons, 3 hidden neurons and 1 output neurons. We will assign the weight between input layer to hidden layer, V and weights between hidden layer to output layer, W. We will then propagate the signal in the forward direction which is called as forward propagation first to get a desired output. Then the desired output is checked with the target output, and we need we to minimise the error if there in the network by performing back-propagation algorithm.

Considering fully connected multi-layer feed forward neural network.

This neural network consists of 3 layers.

Let the transfer function of hidden layer is log-sigmoid and transfer function for output layer is tan-sigmoid.

1. Input to j^{th} hidden neuron.

$$I_{HJ} = \sum_{i=0}^{l} I_i V_{ij}$$

2. Output of the j[th] hidden neuron

$$O_{hj} = \frac{1}{1 + e^{-a_1 I_{hj}}}$$

3. Inputs of the k[th] output neuron

$$I_{ok} = \sum_{j=0}^{m} O_{hj} W_{jk}$$

4. Output of the k[th] output neuron

$$O_{ok} = \frac{e^{-a_2 I_{ok}} - e^{-a_2 I_{ok}}}{e^{-a_2 I_{ok}} + e^{-a_2 I_{ok}}}$$

5. Calculation of error

$$E_k = \frac{1}{2}(T_k - O_{ok})^2$$

$$here\ T_k\ is\ the\ target\ output$$

After applying all these steps we need to train the neural network using back propagation algorithm. For this we use steepest descent method and we use learning rate as 0.80 where negative gradient of the objective function to be optimised.

INPUT-TARGET OUTPUT DATA:

diameter	kinematic viscosity	Velocity	prandltl number	Nu(output)
0.1	1.787	50.02	6.901	0.113477029
0.11	1.788	50.04	6.902	0.122459306
0.12	1.789	50.06	6.903	0.131278131
0.13	1.79	50.08	6.904	0.139949665
0.14	1.791	50.1	6.905	0.148487348
0.15	1.792	50.12	6.906	0.15690252
0.16	1.793	50.14	6.907	0.165204867
0.17	1.794	50.16	6.908	0.173402748
0.18	1.795	50.18	6.909	0.181503448
0.19	1.796	50.2	6.91	0.189513365
0.2	1.797	50.22	6.911	0.19743816
0.21	1.798	50.24	6.912	0.205282874
0.22	1.799	50.26	6.913	0.213052022
0.23	1.8	50.28	6.914	0.220749672
0.24	1.801	50.3	6.915	0.228379502
0.25	1.802	50.32	6.916	0.235944856
0.26	1.803	50.34	6.917	0.243448785
0.27	1.804	50.36	6.918	0.250894084

0.28	1.805	50.38	6.919	0.258283319
0.29	1.806	50.4	6.92	0.265618858
0.3	1.807	50.42	6.921	0.272902887
0.31	1.808	50.44	6.922	0.280137437
0.32	1.809	50.46	6.923	0.287324391
0.33	1.81	50.48	6.924	0.294465506
0.34	1.811	50.5	6.925	0.301562422
0.35	1.812	50.52	6.926	0.308616673
0.36	1.813	50.54	6.927	0.315629697
0.37	1.814	50.56	6.928	0.322602845
0.38	1.815	50.58	6.929	0.329537387
0.39	1.816	50.6	6.93	0.336434523
0.4	1.817	50.62	6.931	0.34329538
0.41	1.818	50.64	6.932	0.35012103
0.42	1.819	50.66	6.933	0.356912483
0.43	1.82	50.68	6.934	0.363670698
0.44	1.821	50.7	6.935	0.370396585
0.45	1.822	50.72	6.936	0.37709101
0.46	1.823	50.74	6.937	0.383754795
0.47	1.824	50.76	6.938	0.390388723
0.48	1.825	50.78	6.939	0.396993543
0.49	1.826	50.8	6.94	0.403569967
0.5	1.827	50.82	6.941	0.410118677
0.51	1.828	50.84	6.942	0.416640325
0.52	1.829	50.86	6.943	0.423135533
0.53	1.83	50.88	6.944	0.429604901
0.54	1.831	50.9	6.945	0.436048999
0.55	1.832	50.92	6.946	0.442468379
0.56	1.833	50.94	6.947	0.448863568
0.57	1.834	50.96	6.948	0.455235074
0.58	1.835	50.98	6.949	0.461583383
0.59	1.836	51	6.95	0.467908965

C CODE:

```
#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include<time.h>

#define Max_E pow(10,-6)


int main()

{
```

```
int i,j,k,p;    //for iterations
int L,M,N,P;
float e,Mean_sq_E;
float I[100][100],IH[100][100],OH[100][100],IO[100][100],OO[100][100],TO[100][100];
float V[100][100],W[100][100],del_W[100][100],del_V[100][100];
float I_min[100],I_max[100],TO_min[100],TO_max[100];

FILE *IP, *OP_1, *OP_2;

        IP  = fopen("IP.txt","r");
        OP_1  = fopen("OP.txt","w");
        OP_2 = fopen("Result.txt","w");

fscanf(IP,"%d%d%d%d",&P,&L,&M,&N);
fprintf(OP_2,"Total No of Patterns (P)=%d\n",P);
fprintf(OP_2,"Total No of Input Neurons (L)=%d\n",L);
fprintf(OP_2,"Total No of Hidden Neurons (M)=%d\n",M);
fprintf(OP_2,"Total No of Output Neurons (N)=%d\n",N);

//Scanning & Printing Inputs for Input layer

for(p=1;p<=P;p++)
{
```

```
            for(i=1;i<=L;i++)
            {
                fscanf(IP,"%f",&I[i][p]);
            }
        }

        fprintf(OP_2,"\nI matrix of order %dX%d :\n",L,P);

        for(i=1;i<=L;i++)
        {
            for(p=1;p<=P;p++)
            {
                fprintf(OP_2,"I[%d][%d]=%f\t",i,p,I[i][p]);
            }
            fprintf(OP_2,"\n");
        }

//Scanning & Printing Target output for Output layer

        fprintf(OP_2,"\nTO matrix of order %dX%d :\n",P,N);

        for(p=1;p<=P;p++)
        {
            for(k=1;k<=N;k++)
```

```
                {
                        fscanf(IP,"%f",&TO[k][p]);
                        fprintf(OP_2,"TO[%d][%d]:%f\t",k,p,TO[k][p]);
                }
                fprintf(OP_2,"\n");
        }

//Normalization of Inputs for Input layer

        for(i=1;i<=L;i++)
        {
                I_max[i]=-1000;I_min[i]=1000;

                for(p=1;p<=P;p++)
                {
                        if(I[i][p]>I_max[i])
                        I_max[i]=I[i][p];
                        if(I[i][p]<I_min[i])
                        I_min[i]=I[i][p];
                }
        }

        for(p=1;p<=P;p++)
        {
```

```
		for(i=1;i<=L;i++)
		{
			I[i][p]=0.1+0.8*((I[i][p]-I_min[i])/(I_max[i]-I_min[i]));
		}
	}

	fprintf(OP_2,"\nNormalized I matrix of order %dX%d :\n",L,P);

	for(i=1;i<=L;i++)
	{
		for(p=1;p<=P;p++)
		{
			fprintf(OP_2,"%f\t",I[i][p]);
		}
		fprintf(OP_2,"\n");
	}


//Normalization of Target output for Output layer

for(k=1;k<N+1;k++)
	{
		TO_max[k]=-1000;TO_min[k]=1000;
		for(p=1;p<=P;p++)
```

```
        {
                if(TO[k][p]>TO_max[k])
                TO_max[k]=TO[k][p];
                if(TO[k][p]<TO_min[k])
                TO_min[k]=TO[k][p];
        }
}

for(p=1;p<=P;p++)
{
        for(k=1;k<=N;k++)
        {
                TO[k][p]=-0.1+(1.0*((TO[k][p]-TO_min[k])/(TO_max[k]-TO_min[k])));
        }
}

fprintf(OP_2,"\nNormalized TO matrix of order %dX%d :\n",P,N);

for(p=1;p<=P;p++)
{
        for(k=1;k<N+1;k++)
        {
                fprintf(OP_2,"TO[%d][%d]:%f\t",k,p,TO[k][p]);
        }
```

```
		fprintf(OP_2,"\n");
	}

srand(time(NULL));

//Define V

	fprintf(OP_2,"\nV matrix of order %dX%d :\n",L+1,M);

	for(i=0;i<L+1;i++)
	{
		for(j=1;j<=M;j++)
		{
			if(i==0)
			{
				V[i][j]=0;
			}
			else
			{
				V[i][j]=1.0*rand()/RAND_MAX;
			}
		}
	}
```

```
	for(i=0;i<=L;i++)
	{
		for(j=1;j<=M;j++)
		{
			fprintf(OP_2,"V[%d][%d]:%f\t",i,j,V[i][j]);
		}
		fprintf(OP_2,"\n");
	}
	fprintf(OP_2,"\n");

//Define W

	fprintf(OP_2,"\nW matrix of order %dX%d :\n",M+1,N);

	for(j=0;j<M+1;j++)
	{
		for(k=1;k<=N;k++)
		{
			if(j==0)
			{
				W[i][j]=0;
			}
			else
			{
```

```
                    W[j][k]=1.0*rand()/RAND_MAX;
                    }
            }
    }

    for(j=0;j<M+1;j++)
    {
            for(k=1;k<=N;k++)
            {
                    fprintf(OP_2,"W[%d][%d]:%f\t",j,k,W[j][k]);
            }
            fprintf(OP_2,"\n");
    }

    int C=1;        //C=Counter

//Do-While loop TRAINING of Patterns

    do
    {
            //Calculation for forward pass

            for(p=1;p<=P-5;p++)
            {
```

```
		IH[j][p]=0;
		for(j=1;j<M+1;j++)
		{
			for(i=1;i<L+1;i++)
			{
				IH[j][p]=IH[j][p]+(I[i][p]*V[i][j]);
			}
			IH[j][p]=IH[j][p]+(1.0);
			OH[j][p]=1/(1+exp(-IH[j][p]));
			IH[j][p]=0;
		}

	}
	fprintf(OP_1,"\n");

	//Calculation for Output of Output layer
	for(p=1;p<=P-5;p++)
	{
		IO[k][p]=0;
		for(k=1;k<N+1;k++)
		{
			for(j=1;j<M+1;j++)
			{
				IO[k][p]=IO[k][p]+OH[j][p]*W[j][k];
```

```
                        }
                        IO[k][p]=IO[k][p]+1.0;
                        OO[k][p]=(exp(IO[k][p])-exp(-1*IO[k][p]))/(exp(IO[k][p])+exp(-
1*IO[k][p]));
                        IO[k][p]=0;
                }
        }
        fprintf(OP_1,"\n");

        //Calculations del_W_jk
        for(j=1;j<=M;j++)
        {
                for(k=1;k<=N;k++)
                {
                        del_W[j][k]=0;
                        for(p=1;p<=P-5;p++)
                        {
                                del_W[j][k]=del_W[j][k]+((0.5/P)*(TO[k][p]-OO[k][p])*(1-
(OO[k][p]*OO[k][p]))*OH[j][p]);
                        }
                        fprintf(OP_1,"del_W[%d][%d]=%f\t",j,k,del_W[j][k]);
                }
                fprintf(OP_1,"\n");
        }
```

```
fprintf(OP_1,"\n");


//Calcualtions del_V_ij
for(i=1;i<=L;i++)
{
	for(j=1;j<=M;j++)
	{
		del_V[i][j]=0;
		for(p=1;p<=P-5;p++)
		{
			for(k=1;k<=N;k++)
			{
				del_V[i][j]=del_V[i][j]+((0.5/(P*N))*((TO[k][p]-
OO[k][p])*(1-(OO[k][p]*OO[k][p]))*W[j][k]*OH[j][p]*(1-OH[j][p])*I[i][p]));
			}
		}
		fprintf(OP_1,"del_V[%d][%d]=%f\t",i,j,del_V[i][j]);
	}
	fprintf(OP_1,"\n");
}


//Calcualtion for e
Mean_sq_E=0;
for(p=1;p<=P-5;p++)
```

```
{
        for(k=1;k<=N;k++)
        {
                e=pow((TO[k][p]-OO[k][p]),2)/2;
                Mean_sq_E=Mean_sq_E+e;
        }
}
Mean_sq_E=Mean_sq_E/P;
fprintf(OP_1,"\nMean_sq_E=%f\tIteration=%d\n",Mean_sq_E,C);

//Updating values of Vij
for(i=1;i<=L;i++)
{
        for(j=1;j<=M;j++)
        {
                V[i][j]=V[i][j]+del_V[i][j];
                fprintf(OP_1,"V[%d][%d]:%f\t",i,j,V[i][j]);
        }

        fprintf(OP_1,"\n");
}
fprintf(OP_1,"\n");

//Updating values of Wjk
```

```
        for(j=1;j<=M;j++)
        {
                for(k=1;k<=N;k++)
                {
                        W[j][k]=W[j][k]+del_W[j][k];
                        fprintf(OP_1,"W[%d][%d]:%f\t",j,k,W[j][k]);
                }
                fprintf(OP_1,"\n");
        }

printf("\nIteration %d completed",C);

C++;

}while(Mean_sq_E>Max_E);

for(i=1;i<=L;i++)
        {
                for(j=1;j<=M;j++)
                {
                        fprintf(OP_2,"V[%d][%d]:%f\t",i,j,V[i][j]);
                }
                fprintf(OP_2,"\n");
        }
```

```
        fprintf(OP_2,"\n");

        //Updating values of Wjk
        for(j=1;j<=M;j++)
        {
                for(k=1;k<=N;k++)
                {
                        fprintf(OP_2,"W[%d][%d]:%f\t",j,k,W[j][k]);
                }
                fprintf(OP_2,"\n");
        }

//Pattern Testing

        //Calculation for forward pass

        fprintf(OP_2,"\nForward Pass Calculation Result:");

        for(p=36;p<=40;p++)
        {
                IH[j][p]=0;
                for(j=1;j<M+1;j++)
                {
                        for(i=1;i<L+1;i++)
```

```
                {
                        IH[j][p]=IH[j][p]+(I[i][p]*V[i][j]);
                }
                IH[j][p]=IH[j][p]+1.0;
                OH[j][p]=1/(1+exp(-IH[j][p]));

fprintf(OP_2,"\nIH[%d][%d]:%f\tOH[%d][%d]:%f",j,p,IH[j][p],j,p,OH[j][p]);
                IH[j][p]=0;
        }

}

fprintf(OP_2,"\n\n");

fprintf(OP_2,"\nOutput of Output layer:");

//Calculation for Output of Output layer
for(p=36;p<=40;p++)
{
        IO[k][p]=0;
        for(k=1;k<N+1;k++)
        {
                for(j=1;j<=M+1;j++)
                {
```

```
                                        IO[k][p]=IO[k][p]+OH[j][p]*W[j][k];
                                }
                                IO[k][p]=IO[k][p]+1.0;
                                OO[k][p]=(exp(IO[k][p])-exp(-1*IO[k][p]))/(exp(IO[k][p])+exp(-
1*IO[k][p]));

        fprintf(OP_2,"\nIO[%d][%d]:%f\tOO[%d][%d]:%f\tTO[%d][%d]%f\terror=:%f",k,p,IO[k][p]
,k,p,OO[k][p],k,p,TO[k][p],fabs(OO[k][p]-TO[k][p]));
                                IO[k][p]=0;
                        }
                }

        fclose(IP);
        fclose(OP_1);
        fclose(OP_2);

        return 0;
}
```

Result

242754 iteration for convergence rate of order of 10^-4 and the output is as below:

Total No of Patterns (P)=50

Total No of Input Neurons (L)=4

Total No of Hidden Neurons (M)=3

Total No of Output Neurons (N)=1

I matrix of order 4X50 :

I[1][1]=0.100000 I[1][2]=0.110000 I[1][3]=0.120000 I[1][4]=0.130000
I[1][5]=0.140000 I[1][6]=0.150000 I[1][7]=0.160000 I[1][8]=0.170000
I[1][9]=0.180000 I[1][10]=0.190000 I[1][11]=0.200000 I[1][12]=0.210000
I[1][13]=0.220000 I[1][14]=0.230000 I[1][15]=0.240000 I[1][16]=0.250000
I[1][17]=0.260000 I[1][18]=0.270000 I[1][19]=0.280000 I[1][20]=0.290000
I[1][21]=0.300000 I[1][22]=0.310000 I[1][23]=0.320000 I[1][24]=0.330000
I[1][25]=0.340000 I[1][26]=0.350000 I[1][27]=0.360000 I[1][28]=0.370000
I[1][29]=0.380000 I[1][30]=0.390000 I[1][31]=0.400000 I[1][32]=0.410000
I[1][33]=0.420000 I[1][34]=0.430000 I[1][35]=0.440000 I[1][36]=0.450000
I[1][37]=0.460000 I[1][38]=0.470000 I[1][39]=0.480000 I[1][40]=0.490000
I[1][41]=0.500000 I[1][42]=0.510000 I[1][43]=0.520000 I[1][44]=0.530000
I[1][45]=0.540000 I[1][46]=0.550000 I[1][47]=0.560000 I[1][48]=0.570000
I[1][49]=0.580000 I[1][50]=0.590000

I[2][1]=1.787000 I[2][2]=1.788000 I[2][3]=1.789000 I[2][4]=1.790000
I[2][5]=1.791000 I[2][6]=1.792000 I[2][7]=1.793000 I[2][8]=1.794000
I[2][9]=1.795000 I[2][10]=1.796000 I[2][11]=1.797000 I[2][12]=1.798000
I[2][13]=1.799000 I[2][14]=1.800000 I[2][15]=1.801000 I[2][16]=1.802000
I[2][17]=1.803000 I[2][18]=1.804000 I[2][19]=1.805000 I[2][20]=1.806000
I[2][21]=1.807000 I[2][22]=1.808000 I[2][23]=1.809000 I[2][24]=1.810000
I[2][25]=1.811000 I[2][26]=1.812000 I[2][27]=1.813000 I[2][28]=1.814000
I[2][29]=1.815000 I[2][30]=1.816000 I[2][31]=1.817000 I[2][32]=1.818000
I[2][33]=1.819000 I[2][34]=1.820000 I[2][35]=1.821000 I[2][36]=1.822000
I[2][37]=1.823000 I[2][38]=1.824000 I[2][39]=1.825000 I[2][40]=1.826000
I[2][41]=1.827000 I[2][42]=1.828000 I[2][43]=1.829000 I[2][44]=1.830000
I[2][45]=1.831000 I[2][46]=1.832000 I[2][47]=1.833000 I[2][48]=1.834000
I[2][49]=1.835000 I[2][50]=1.836000

I[3][1]=50.020000 I[3][2]=50.040001 I[3][3]=50.060001 I[3][4]=50.080002
I[3][5]=50.099998 I[3][6]=50.119999 I[3][7]=50.139999 I[3][8]=50.160000
I[3][9]=50.180000 I[3][10]=50.200001 I[3][11]=50.220001 I[3][12]=50.240002
I[3][13]=50.259998 I[3][14]=50.279999 I[3][15]=50.299999 I[3][16]=50.320000
I[3][17]=50.340000 I[3][18]=50.360001 I[3][19]=50.380001 I[3][20]=50.400002
I[3][21]=50.419998 I[3][22]=50.439999 I[3][23]=50.459999 I[3][24]=50.480000
I[3][25]=50.500000 I[3][26]=50.520000 I[3][27]=50.540001 I[3][28]=50.560001
I[3][29]=50.580002 I[3][30]=50.599998 I[3][31]=50.619999 I[3][32]=50.639999

I[3][33]=50.660000 I[3][34]=50.680000 I[3][35]=50.700001 I[3][36]=50.720001
I[3][37]=50.740002 I[3][38]=50.759998 I[3][39]=50.779999 I[3][40]=50.799999
I[3][41]=50.820000 I[3][42]=50.840000 I[3][43]=50.860001 I[3][44]=50.880001
I[3][45]=50.900002 I[3][46]=50.919998 I[3][47]=50.939999 I[3][48]=50.959999
I[3][49]=50.980000 I[3][50]=51.000000

I[4][1]=6.901000 I[4][2]=6.902000 I[4][3]=6.903000 I[4][4]=6.904000
I[4][5]=6.905000 I[4][6]=6.906000 I[4][7]=6.907000 I[4][8]=6.908000
I[4][9]=6.909000 I[4][10]=6.910000 I[4][11]=6.911000 I[4][12]=6.912000
I[4][13]=6.913000 I[4][14]=6.914000 I[4][15]=6.915000 I[4][16]=6.916000
I[4][17]=6.917000 I[4][18]=6.918000 I[4][19]=6.919000 I[4][20]=6.920000
I[4][21]=6.921000 I[4][22]=6.922000 I[4][23]=6.923000 I[4][24]=6.924000
I[4][25]=6.925000 I[4][26]=6.926000 I[4][27]=6.927000 I[4][28]=6.928000
I[4][29]=6.929000 I[4][30]=6.930000 I[4][31]=6.931000 I[4][32]=6.932000
I[4][33]=6.933000 I[4][34]=6.934000 I[4][35]=6.935000 I[4][36]=6.936000
I[4][37]=6.937000 I[4][38]=6.938000 I[4][39]=6.939000 I[4][40]=6.940000
I[4][41]=6.941000 I[4][42]=6.942000 I[4][43]=6.943000 I[4][44]=6.944000
I[4][45]=6.945000 I[4][46]=6.946000 I[4][47]=6.947000 I[4][48]=6.948000
I[4][49]=6.949000 I[4][50]=6.950000

TO matrix of order 50X1 :

TO[1][1]:0.113477

TO[1][2]:0.122459

TO[1][3]:0.131278

TO[1][4]:0.139950

TO[1][5]:0.148487

TO[1][6]:0.156903

TO[1][7]:0.165205

TO[1][8]:0.173403

TO[1][9]:0.181503

TO[1][10]:0.189513

TO[1][11]:0.197438

TO[1][12]:0.205283

TO[1][13]:0.213052

TO[1][14]:0.220750

TO[1][15]:0.228380

TO[1][16]:0.235945

TO[1][17]:0.243449

TO[1][18]:0.250894

TO[1][19]:0.258283

TO[1][20]:0.265619

TO[1][21]:0.272903

TO[1][22]:0.280137

TO[1][23]:0.287324

TO[1][24]:0.294466

TO[1][25]:0.301562

TO[1][26]:0.308617

TO[1][27]:0.315630

TO[1][28]:0.322603

TO[1][29]:0.329537

TO[1][30]:0.336435

TO[1][31]:0.343295

TO[1][32]:0.350121

TO[1][33]:0.356912

TO[1][34]:0.363671

TO[1][35]:0.370397

TO[1][36]:0.377091

TO[1][37]:0.383755

TO[1][38]:0.390389

TO[1][39]:0.396994

TO[1][40]:0.403570

TO[1][41]:0.410119

TO[1][42]:0.416640

TO[1][43]:0.423136

TO[1][44]:0.429605

TO[1][45]:0.436049

TO[1][46]:0.442468

TO[1][47]:0.448864

TO[1][48]:0.455235

TO[1][49]:0.461583

TO[1][50]:0.467909

Normalized I matrix of order 4X50 :

0.100000	0.116327	0.132653	0.148980	0.165306	0.181633
0.197959	0.214286	0.230612	0.246939	0.263265	0.279592
0.295918	0.312245	0.328571	0.344898	0.361224	0.377551
0.393878	0.410204	0.426531	0.442857	0.459184	0.475510
0.491837	0.508163	0.524490	0.540816	0.557143	0.573469
0.589796	0.606122	0.622449	0.638776	0.655102	0.671429
0.687755	0.704082	0.720408	0.736735	0.753061	0.769388
0.785714	0.802041	0.818367	0.834694	0.851020	0.867347
0.883673	0.900000				

0.100000 0.116327 0.132655 0.148980 0.165307 0.181634
0.197960 0.214287 0.230612 0.246940 0.263267 0.279592
0.295920 0.312245 0.328572 0.344900 0.361225 0.377552
0.393878 0.410205 0.426532 0.442857 0.459185 0.475510
0.491837 0.508165 0.524490 0.540817 0.557145 0.573470
0.589797 0.606122 0.622450 0.638777 0.655102 0.671430
0.687755 0.704082 0.720410 0.736735 0.753062 0.769388
0.785715 0.802042 0.818367 0.834695 0.851020 0.867347
0.883675 0.900000

0.100000 0.116327 0.132654 0.148981 0.165305 0.181631
0.197958 0.214285 0.230612 0.246939 0.263266 0.279593
0.295917 0.312244 0.328571 0.344897 0.361224 0.377551
0.393878 0.410205 0.426529 0.442856 0.459183 0.475510
0.491837 0.508163 0.524490 0.540817 0.557144 0.573468
0.589795 0.606122 0.622449 0.638776 0.655103 0.671429
0.687756 0.704080 0.720407 0.736734 0.753061 0.769388
0.785715 0.802042 0.818369 0.834692 0.851019 0.867346
0.883673 0.900000

0.100000 0.116325 0.132651 0.148976 0.165309 0.181635
0.197960 0.214286 0.230611 0.246937 0.263262 0.279595
0.295921 0.312246 0.328571 0.344897 0.361222 0.377555
0.393881 0.410206 0.426532 0.442857 0.459183 0.475508
0.491841 0.508167 0.524492 0.540817 0.557143 0.573468
0.589801 0.606127 0.622452 0.638778 0.655103 0.671429
0.687754 0.704087 0.720413 0.736738 0.753063 0.769389
0.785714 0.802040 0.818373 0.834698 0.851024 0.867349
0.883675 0.900000

Normalized TO matrix of order 50X1 :

TO[1][1]:-0.100000

TO[1][2]:-0.074657

TO[1][3]:-0.049776

TO[1][4]:-0.025310

TO[1][5]:-0.001221

TO[1][6]:0.022521

TO[1][7]:0.045946

TO[1][8]:0.069075

TO[1][9]:0.091931

TO[1][10]:0.114530

TO[1][11]:0.136889

TO[1][12]:0.159022

TO[1][13]:0.180942

TO[1][14]:0.202661

TO[1][15]:0.224188

TO[1][16]:0.245533

TO[1][17]:0.266704

TO[1][18]:0.287711

TO[1][19]:0.308559

TO[1][20]:0.329255

TO[1][21]:0.349807

TO[1][22]:0.370218

TO[1][23]:0.390496

TO[1][24]:0.410644

TO[1][25]:0.430667

TO[1][26]:0.450570

TO[1][27]:0.470357

TO[1][28]:0.490031

TO[1][29]:0.509596

TO[1][30]:0.529056

TO[1][31]:0.548413

TO[1][32]:0.567671

TO[1][33]:0.586833

TO[1][34]:0.605900

TO[1][35]:0.624877

TO[1][36]:0.643765

TO[1][37]:0.662566

TO[1][38]:0.681283

TO[1][39]:0.699918

TO[1][40]:0.718473

TO[1][41]:0.736949

TO[1][42]:0.755350

TO[1][43]:0.773675

TO[1][44]:0.791928

TO[1][45]:0.810110

TO[1][46]:0.828221

TO[1][47]:0.846265

TO[1][48]:0.864242

TO[1][49]:0.882153

TO[1][50]:0.900000

V matrix of order 5X3 :

V[0][1]:0.000000 V[0][2]:0.000000 V[0][3]:0.000000

V[1][1]:0.990356 V[1][2]:0.154515 V[1][3]:0.430708

V[2][1]:0.012482 V[2][2]:0.387127 V[2][3]:0.349956

V[3][1]:0.360851 V[3][2]:0.650838 V[3][3]:0.249916

V[4][1]:0.234626 V[4][2]:0.867702 V[4][3]:0.907529

W matrix of order 4X1 :

W[0][1]:0.000000

W[1][1]:0.369152

W[2][1]:0.547166

W[3][1]:0.905759

V[1][1]:0.240079 V[1][2]:-0.347375 V[1][3]:-0.041741

V[2][1]:-0.738050 V[2][2]:-0.114859 V[2][3]:-0.122488

V[3][1]:-0.389706 V[3][2]:0.148844 V[3][3]:-0.222497

V[4][1]:-0.515534 V[4][2]:0.365768 V[4][3]:0.435017

W[1][1]:-4.525525

W[2][1]:1.191946

W[3][1]:1.649294

Forward Pass Calculation Result:

IH[1][36]:0.057843 OH[1][36]:0.514457

IH[2][36]:1.035169 OH[2][36]:0.737917

IH[3][36]:1.032423 OH[3][36]:0.737385

IH[1][37]:0.034935 OH[1][37]:0.508733

IH[2][37]:1.036024 OH[2][37]:0.738082

IH[3][37]:1.033211 OH[3][37]:0.737538

IH[1][38]:0.012022 OH[1][38]:0.503006

IH[2][38]:1.036881 OH[2][38]:0.738248

IH[3][38]:1.034003 OH[3][38]:0.737691

IH[1][39]:-0.010887 OH[1][39]:0.497278

IH[2][39]:1.037735 OH[2][39]:0.738413

IH[3][39]:1.034791 OH[3][39]:0.737844

IH[1][40]:-0.033796 OH[1][40]:0.491552

IH[2][40]:1.038590 OH[2][40]:0.738578

IH[3][40]:1.035579 OH[3][40]:0.737996

Output of Output layer:

IO[1][36]:0.767535 OO[1][36]:0.645494 TO[1][36]0.643765 error=:0.001729

IO[1][37]:0.793888 OO[1][37]:0.660606 TO[1][37]0.662566 error=:0.001960

IO[1][38]:0.820257 OO[1][38]:0.675210 TO[1][38]0.681283 error=:0.006073

IO[1][39]:0.846624 OO[1][39]:0.689302 TO[1][39]0.699918 error=:0.010616

IO[1][40]:0.872987 OO[1][40]:0.702889 TO[1][40]0.718473 error=:0.015584

CONCLUSION:

THE ERROR IS APPROXIMATELY 0.01% for convergence of 10^-4, the error will minimise if we change the convergence rate to 10^-6. Here it is taking too long time to convege for 10^-6. So we use convergence rate of order of 10^-4.