Matlab Implementation of the Enhanced Interval Approach for Encoding Words into Interval Type-2 Fuzzy Sets

Dongrui Wu, Jerry M. Mendel, and Simon Coupland

Matlab implementation of the EIA, proposed in the paper:

> Dongrui Wu, Simon Coupland, and Jerry M. Mendel, "Enhanced Interval Approach for Encoding Words into Interval Type-2 Fuzzy Sets and Its Convergence Analysis," submitted to *IEEE Trans. on Fuzzy Systems*, 2011.

is given here so that it can be freely used by other researchers. The input and output parameters are:

- *a*: left end-points of the data intervals from survey
- *b*: right end-points of the data intervals from survey
- *MFs*: MF of the word model defined by 9 parameters. For left-shoulder, it is $[0, 0, \overline{a}_{\rm MF}, \overline{b}_{\rm MF}, 0, 0, \underline{a}_{\rm MF}, \underline{b}_{\rm MF}, 1]$. For interior FOU, it is $[\underline{a}_{\rm MF}, \underline{c}_{\rm MF}, \overline{c}_{\rm MF}, \overline{b}_{\rm MF}, \overline{b}_{\rm MF}, \overline{a}_{\rm MF}, p, p, \underline{b}_{\rm MF}, \mu_p]$. For right-shoulder, it is $[\underline{a}_{\rm MF}, \underline{b}_{\rm MF}, M, M, \overline{a}_{\rm MF}, \overline{b}_{\rm MF}, M, M, 1]$
- *nums*: number of remaining intervals after each precessing step
- *shape*: left-shoulder (1), interior (2), or right-shoulder (3)
- *FSL*: left endpoints of the remaining embedded T1 FSs
- *FSR*: right endpoints of the remaining embedded T1 FSs

```
function [MF, nums, shape, FSL, FSR] = EIA(a,b)

MF=zeros(1,9); nums=zeros(1,8); shape=0; FSL=0; FSR=0; n=length(a);

%% Remove incomplete data intervals
index=find(isnan(a)+isnan(b)); a(index)=[]; b(index)=[];

%% Bad data processing
for i=length(a):-1:1
    if a(i)<0 || a(i)>10 || b(i)<0 || b(i)>10 ||  b(i)<=a(i) || b(i)-a(i)>=10
        a(i)=[]; b(i)=[];
    end
end
n1=length(a); % n'
if n1==0; return; end

%% Outlier processing
l=sort(a); r=sort(b); n3=floor(n1*0.25); n4=floor(n1*0.75);

% Compute Q(0.25), Q(0.75) and IQR for a
Qa25=l(n3)*(1-rem(0.25*n1,1))+l(n3+1)*rem(0.25*n1,1);
```

Dongrui Wu is with the Industrial Artificial Intelligence Laboratory, GE Global Research, Niskayuna, NY 12309 USA (email: wud@ge.com).

Jerry M. Mendel is with the Signal and Image Processing Institute, Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089-2564 USA (e-mail: mendel@sipi.usc.edu).

Simon Coupland is with the Center for Computational Intelligence, De Montfort University, Leicester, LE1 9BH UK (e-mail: simonc@dmu.ac.uk).

```
Qa75=l(n4)*(1-rem(0.75*n1,1))+l(n4+1)*rem(0.75*n1,1);
IQRa=Qa75 - Qa25;

% Compute Q(0.25), Q(0.75) and IQR for b
Qb25=r(n3)*(1-rem(0.25*n1,1))+r(n3+1)*rem(0.25*n1,1);
Qb75=r(n4)*(1-rem(0.75*n1,1))+r(n4+1)*rem(0.75*n1,1);
IQRb=Qb75-Qb25;

% Outlier processing for a and b
for i=n1:-1:1
    if a(i)<Qa25-1.5*IQRa || a(i)>Qa75+1.5*IQRa...
            || b(i)<Qb25-1.5*IQRb || b(i)>Qb75+1.5*IQRb
        a(i)=[];   b(i)=[];
    end
end
n2=length(a); % n''
if n2==0; return; end

% Compute Q(0.25), Q(0.75) and IQR for interval length
L=b-a; leng=sort(L); n3=floor(n2*0.25); n4=floor(n2*0.75);
QL25=leng(n3)*(1-rem(0.25*n2,1))+leng(n3+1)*rem(0.25*n2,1);
QL75=leng(n4)*(1-rem(0.75*n2,1))+leng(n4+1)*rem(0.75*n2,1);
IQRL=QL75-QL25;

% outlier processing for interval length
for i=n2:-1:1
    if L(i)<QL25-1.5*IQRL || L(i)>QL75+1.5*IQRL
        a(i)=[];  b(i)=[];   L(i)=[];
    end
end
m1=length(a); % m'
if m1==0; return; end

%% Tolerance limit processing
ma=mean(a); stda=std(a); mb=mean(b); stdb=std(b);
K=[5 4.275 5.079 6.634 6.612 7.855 10.3
    6 3.712 4.414 5.775 5.337 6.345 8.301
    7 3.369 4.007 5.248 4.613 5.488 7.187
    8 3.136 3.732 4.891 4.125 4.936 6.468
    9 2.967 3.532 4.631 3.822 4.550 5.966
    10 2.839 3.379 4.433 3.582 4.265 5.594
    11 2.737 3.259 4.277 3.397 4.045 5.308
    12 2.655 3.162 4.150 3.250 3.870 5.079
    13 2.587 3.081 4.044 3.130 3.727 4.893
    14 2.529 3.012 3.955 3.029 3.608 4.737
    15 2.480 2.954 3.878 2.945 3.507 4.605
    16 2.437 2.903 3.812 2.872 3.421 4.492
    17 2.400 2.858 3.754 9.808 3.345 4.393
    18 2.366 2.819 3.702 2.753 3.279 4.307
    19 2.337 2.784 3.656 2.703 3.221 4.230
    20 2.310 2.752 3.615 2.659 3.168 4.161
    25 2.208 2.631 3.457 2.494 2.972 3.904
```

```
    30 2.140 2.549 3.350 2.385 2.841 3.733
    35 2.090 2.490 3.272 2.306 2.748 3.611
    40 2.052 2.445 3.213 2.247 2.677 3.518
    45 2.021 2.408 3.165 2.200 2.621 3.444
    50 1.996 2.379 3.126 2.162 2.576 3.385
    60 1.958 2.333 3.066 2.103 2.506 3.293
    70 1.929 2.299 3.021 2.060 2.454 3.225
    80 1.907 2.272 2.986 2.026 2.414 3.173
    90 1.889 2.251 2.958 1.999 2.382 3.130
    100 1.874 2.233 2.934 1.977 2.355 3.096
    150 1.825 2.175 2.859 1.905 2.270 2.983
    200 1.798 2.143 2.816 1.865 2.222 2.921
    250 1.780 2.121 2.788 1.839 2.191 2.880
    300 1.767 2.106 2.767 1.820 2.169 2.850];
[temp,index]=min(abs(K(:,1)-m1));
k=K(index,2); % gamma=0.05, alpha=0.1

% Tolerance limit processing for a and b
for i=m1:-1:1
    if a(i)<ma-k*stda || a(i)>ma+k*stda || b(i)<mb-k*stdb || b(i)>mb+k*stdb
        a(i)=[];   b(i)=[];   L(i)=[];
    end
end
mplus=length(a); % m*
if mplus==0; return; end

% Tolerance limit processing for interval length
mL=mean(L); stdL=std(L);
[temp,index]=min(abs(K(:,1)-mplus));
k=min([K(index,2),mL/stdL,(10-mL)/stdL]);
for i=mplus:-1:1
    if  L(i)<mL-k*stdL || L(i)>mL+k*stdL
        a(i)=[];   b(i)=[];  L(i)=[];
    end
end
m2=length(a); %m''
if m2==0; return; end

%% Reasonable interval processing
ma=mean(a); stda=std(a); mb=mean(b); stdb=std(b);

% Determine xi*
if stda==stdb
    xi=(ma+mb)/2;
elseif stda==0
    xi=ma+0.0001;
elseif stdb==0
    xi=mb-0.0001;
else
    xi1=(mb*stda^2-ma*stdb^2+stda*stdb*sqrt((ma-mb)^2 ...
    +2*(stda^2-stdb^2)*log(stda/stdb)))/(stda^2-stdb^2);
    xi2=(mb*stda^2-ma*stdb^2-stda*stdb*sqrt((ma-mb)^2 ...
```

```
        +2*(stda^2-stdb^2)*log(stda/stdb)))/(stda^2-stdb^2);
    if  xi1>=ma && xi1<=mb
        xi=xi1;
    else
        xi=xi2;
    end
end

% Reasonable interval processing
for i=m2:-1:1
    if a(i)>=xi || b(i)<=xi || a(i)<2*ma-xi || b(i)>2*mb-xi
        a(i)=[];    b(i)=[];    L(i)=[];
    end
end
m=length(a);
if m==0; return; end

%%  Admissible region determination
tTable=[5 1.476 2.015 3.365
    6 1.440 1.943 3.143
    7 1.415 1.895 2.998
    8 1.397 1.860 2.896
    9 1.383 1.833 2.821
    10 1.372 1.812 2.764
    11 1.363 1.796 2.718
    12 1.356 1.782 2.681
    13 1.350 1.771 2.650
    14 1.345 1.761 2.624
    15 1.341 1.753 2.602
    16 1.337 1.746 2.583
    17 1.333 1.740 2.567
    18 1.330 1.734 2.552
    19 1.328 1.729 2.539
    20 1.325 1.725 2.528
    21 1.323 1.721 2.518
    22 1.321 1.717 2.508
    23 1.319 1.714 2.500
    24 1.318 1.711 2.492
    25 1.316 1.708 2.485
    26 1.315 1.706 2.479
    27 1.314 1.703 2.473
    28 1.313 1.701 2.467
    29 1.311 1.699 2.462
    30 1.310 1.697 2.457
    40 1.303 1.684 2.423
    50 1.299 1.676 2.403
    60 1.296 1.671 2.390
    80 1.292 1.664 2.374
    100 1.290 1.660 2.364
    1000 1.282 1.646 2.330];
[temp,index]=min(abs(tTable(:,1)-m+1));
t=tTable(index,3); ml=mean(a); mr=mean(b) ;
```

```
c=b-5.831*a; d=b-0.171*a-8.29;
tc=t*std(c)/sqrt(m); td=t*std(d)/sqrt(m);
FSL=zeros(1,m); FSR=zeros(1,m);

%% Establish nature of FOU
if mr>5.831*ml-tc && mr<.171*ml+8.29-td
    for i=m:-1:1  % lef-shoulder
        FSL(i)=0.5*(a(i)+b(i))-(b(i)-a(i))/sqrt(6);
        FSR(i)=0.5*(a(i)+b(i))+sqrt(6)*(b(i)-a(i))/3;
        % Delete inadmissible T1 FSs
        if FSL(i)<0 || FSR(i)>10
            FSL(i)=[];   FSR(i)=[];
        end
    end
    shape=1; m0=length(FSL);
    if m0==0; return; end
    UMF=[0,  0, max(FSL), max(FSR)];
    LMF=[0, 0, min(FSL), min(FSR), 1];
elseif  mr<5.831*ml-tc && mr>.171*ml+8.29-td
    for i=m:-1:1 % right-shoulder
        FSL(i)=0.5*(a(i)+b(i))-sqrt(6)*(b(i)-a(i))/3;
        FSR(i)=0.5*(a(i)+b(i))+(b(i)-a(i))/sqrt(6);
        % Delete inadmissible T1 FSs
        if FSL(i)<0 || FSR(i)>10
            FSL(i)=[];  FSR(i)=[];
        end
    end
    shape=3; m0=length(FSL);
    if m0==0; return; end
    UMF=[min(FSL), min(FSR), 10, 10];
    LMF=[max(FSL), max(FSR), 10, 10, 1];
else
    for i=m:-1:1 % Interior FOU
        FSL(i)=0.5*(a(i)+b(i))-sqrt(2)*(b(i)-a(i))/2;
        FSR(i)=0.5*(a(i)+b(i))+sqrt(2)*(b(i)-a(i))/2;
        % Delete inadmissible T1 FSs
        if FSL(i)< 0 || FSR(i)>10
            FSL(i)=[];   FSR(i)=[];
        end
    end
    FSC=(FSL+FSR)/2;
    L1=min(FSL);  L2=max(FSL); R1=min(FSR);  R2=max(FSR);
    C1=min(FSC);  C2=max(FSC); shape=2; m0=length(FSL);
    if m0==0; return; end
    hs=zeros(1,m0*(m0-1));
    for i=1:m0
        hs((i-1)*m0+(1:m0))=(FSR(i)-FSL)./(FSR(i)-FSL+FSC-FSC(i));
    end
    [h,index]=min(hs);
    i=ceil(index/m0);   j=index-(i-1)*m0;
    p=FSL(j)+h*(FSC(j)-FSL(j));
    UMF=[L1, C1, C2, R2];   LMF=[L2, p, p, R1, h];
```

```
end

MF=[UMF LMF];
nums=[n n1 n2 m1 mplus m2 m m0];
```