ANEXO A

1. Os programas em MATLAB, o sinal ECG sintético e sinal do sensor, utilizados neste trabalho, estão neste anexo.

I. As principais funções que realizam a difusão anisotrópica-mediana neste trabalho são: “filtrow0M.m”, “filtrow3m.m”, “filtrow5m.m”, “filtrow7m.m”, “filtrow9m.m”, “filtrow3mit.m”, “filtrow3p.m”, “filtrosensorx.m”, “filtrosensorxn.m”.

Referências: Biblioteca para Processamento de Imagens e Visão Computacional IMG (KIM, 2006) de autoria do Prof. Dr. Hae Yong Kim. Referências também em (Melo, 2008), (Melo, 2004), (Kim, 2003), (www.physionet.org) e (Kovesi, 2003).

II. A função que calcula a função MAD é: “mad.m”

III. A função que gera o sinal artificial de ECG é: “ECGwaveGen.m”

IV. A função que gera ruído alfa estável é: “randalph.m”

http://www.ee.udel.edu/~gonzalez/ (fornecida por Dr. Pander)

V. A função que gera o intervalo de tempo do sinal do banco de dados do MIT é “PhysioNet_ECG_DAT_MAT.mat”

Referências (www.physionet.org), (Melo, 2008)

VI. A função que converte os sinais em de ECG “.dat” para “.MAT” é: “PhysioNet_ECG_Exporter”

Physionet DataBank Available: (www.physionet.org)

2. A função filtrow0M.m realiza somente a difusão anisotrópica.

3. A funçao filtrow3m.m realiza a difusão-mediana com janela 3.

4. A função filtrow5m.m realiza a difusão-mediana com janela 5.

5. A funçao filtrow7m.m realiza a difusão-mediana com janela 7.

6. A função filtrow9m.m realiza a difusão-mediana com janela 9.

7. A função filtrow3mit.m realiza a difusão-mediana com janela 3 para dados de ECG do MIT.

8. A função filtrow3p.m realiza a difusão-mediana com janela 3 para dados de ECG usados em (Pander, 2004).

9. A função filtrosensorx.m realiza a difusão-mediana com janela 3 para dados do sensor piezoelétrico, fornecidos por (DELIJAICOV, 2004). Nesta função as amostras das extremidades do sinal a ser filtrado são mantidas constantes ao longo do processo de fiiltragem.

10. A função filtrosensorxn.m realiza somente a difusão anisotrópica para dados do sensor piezoelétrico, fornecidos por (DELIJAICOV, 2004). Nesta função as amostras das extremidades do sinal a ser filtrado são mantidas constantes ao longo do processo de fiiltragem.

11. Sinal ECG sintético.

12. Sinal sensor.

As funções acima apresentam os parâmetros de entrada (A,B,C,D,E,F,G,H,I,J,K), exceto a função filtrosensorxn.m que apresenta os parâmetros (A,B,C,D,E,F,G,H)

A = o vetor sinal a ser filtrado
B = número de iterações
C = valor da escala usada
D = peso dado às contribuições dos 2 vizinhos na difusão. No caso de sinais o valor normalmente é igual a 0,5.
E = tipo da função parada-na-aresta a ser utilizada, podendo ser:

- 1 tipo Malik-Perona1,
- 2 tipo Malik-Perona2,
- 3 tipo Degrau,
- 4 tipo Tukey,
- 5 tipo Huber,
- 6 tipo Norma L1,
- 7 tipo Geman and Mc Clure,
- 8 tipo Forward and Backward C1,
- 9 tipo SIDE,
- 10 tipo Forward and Backward C2

F = Ganho proporcional do gradiente, usar = 1

G = Ganho integral do gradiente, usar = 0

H = Ganho derivativo do gradiente, usar = 0

I = Números de pontos do sinal de entrada

J = Sinal sem ruído usado para cálculo do erro

K = Ganho de escala da curva de estimação, usar = 0 (não usado)

EXEMPLO DE USO:

a) *Defina o filtro não linear a ser usado escolhendo uma função do item 1 descrito acima, defina o valor de iterações, escala, peso dado às contribuições dos 2 vizinhos na difusão (0,5 para sinais) e o tipo da função parada-na-aresta. Por exemplo:*

50 iterações, escala igual a 0.198, contribuição dos 2 vizinhos igual a .5 e tipo da função parada-na-aresta (1=Malik-Perona1).

b) *defina um vetor linha com os dados a serem filtrados.*

>> amp= [1 2 3 4 5 6 7 8 9 10]

c) *Processe a função atribuindo o resultado produzido pela função a um vetor linha de saída.*

>> xout=filtroXX(amp,50,0.198,.5,1,1,0,0,10,J,0);

d) Use o comando Plot para visualizar os dados filtrados.

>> plot (xout)

```
function [diff,vetor,erro,NMSE] = filtroW0M(im,niterations, kappa, lambda, option,p,i,↙
d,pontos,io,ganho)

Kp=p;
Ki=i;
Kd=d;




im = double(im);
[rows,cols] = size(im);
diff = im;

integral_erroN= [diff*0];
integral_erroS= [diff*0];
integral_erroE= [diff*0];
integral_erroW= [diff*0];
erroN_anterior= [diff*0];
erroS_anterior= [diff*0];
erroE_anterior= [diff*0];
erroW_anterior= [diff*0];
std_diff=std(diff);
std_diff_anterior=std(diff);
for i = 1:niterations




  diffl = zeros(rows+2, cols+2);
  diffl(2:rows+1, 2:cols+1) = diff;


  deltaNa = diffl(1:rows,2:cols+1) - diff;
  deltaSa = diffl(3:rows+2,2:cols+1) - diff;
  deltaEa = diffl(2:rows+1,3:cols+2) - diff;
  deltaWa = diffl(2:rows+1,1:cols) - diff;

  derivada_erroNa = deltaNa - erroN_anterior;
  derivada_erroSa = deltaSa - erroS_anterior;
  derivada_erroEa = deltaEa - erroE_anterior;
  derivada_erroWa = deltaWa - erroW_anterior;

  erroN_anterior = deltaNa; %GUARDA ERRO ANTERIOR
  erroS_anterior = deltaSa;
  erroE_anterior = deltaEa;
  erroW_anterior = deltaWa;

  integral_erroN = deltaNa + integral_erroN;
  integral_erroS = deltaSa + integral_erroS;
  integral_erroE = deltaEa + integral_erroE;
  integral_erroW = deltaWa + integral_erroW;

  deltaN = ((Kp*deltaNa)+(Ki*integral_erroN)+(Kd*derivada_erroNa));
```

```
deltaS = ((Kp*deltaSa)+(Ki*integral_erroS)+(Kd*derivada_erroSa));
deltaE = ((Kp*deltaEa)+(Ki*integral_erroE)+(Kd*derivada_erroEa));
deltaW = ((Kp*deltaWa)+(Ki*integral_erroW)+(Kd*derivada_erroWa));


if option == 1 %perona 1

  cN = 1./(1+(((deltaN.^2))/((kappa.^2))));
  cS = 1./(1+(((deltaS.^2))/((kappa.^2))));
  cE = 1./(1+(((deltaE.^2))/((kappa.^2))));
  cW = 1./(1+(((deltaW.^2))/((kappa.^2))));

elseif option == 2 % perona 2

  cN = exp(-(((deltaN.^2)/(2*(kappa.^2)))));
  cS = exp(-(((deltaS.^2)/(2*(kappa.^2)))));
  cE = exp(-(((deltaE.^2)/(2*(kappa.^2)))));
  cW = exp(-(((deltaW.^2)/(2*(kappa.^2)))));

elseif option == 3 % degrau

  cN = deltaN./deltaN;
  cS = deltaS./deltaS;
  cE = deltaE./deltaE;
  cW = deltaW./deltaW;

  cN((deltaN.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cS((deltaS.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cE((deltaE.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cW((deltaW.^2./(sqrt(2)*kappa).^2)>=0.5)=0;

  cN((deltaN.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cS((deltaS.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cE((deltaE.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cW((deltaW.^2./(sqrt(2)*kappa).^2)<0.5)=1;

elseif option == 4  % tukey

  cN = (1-((deltaN.^2)/(5*(kappa.^2)))).^2;
  cS = (1-((deltaS.^2)/(5*(kappa.^2)))).^2;
  cE = (1-((deltaE.^2)/(5*(kappa.^2)))).^2;
  cW = (1-((deltaW.^2)/(5*(kappa.^2)))).^2;

  cN(abs((deltaN.^2)/5)>((kappa.^2)))=0;
  cS(abs((deltaS.^2)/5)>((kappa.^2)))=0;
  cE(abs((deltaE.^2)/5)>((kappa.^2)))=0;
  cW(abs((deltaW.^2)/5)>((kappa.^2)))=0;

elseif option == 5 % huber

  cN=abs(kappa./(deltaN.*3.7));
  cS=abs(kappa./(deltaS.*3.7));
  cE=abs(kappa./(deltaE.*3.7));
  cW=abs(kappa./(deltaW.*3.7));

  cN(abs(deltaN.*3.7)<=kappa)=1;
```

```
    cS(abs(deltaS.*3.7)<=kappa)=1;
    cE(abs(deltaE.*3.7)<=kappa)=1;
    cW(abs(deltaW.*3.7)<=kappa)=1;

  elseif option == 6 % L1 norm

    cN = deltaN./deltaN;
    cS = deltaS./deltaS;
    cE = deltaE./deltaE;
    cW = deltaW./deltaW;

    cN((deltaN)<0)=-1;
    cS((deltaS)<0)=-1;
    cE((deltaE)<0)=-1;
    cW((deltaW)<0)=-1;

    cN((deltaN)>0)=1;
    cS((deltaS)>0)=1;
    cE((deltaE)>0)=1;
    cW((deltaW)>0)=1;



  elseif option == 7  % geman and mcclure

    cN = 2./((1+(.014.*deltaN.^2)).^2);
    cS = 2./((1+(.014.*deltaS.^2)).^2);
    cE = 2./((1+(.014.*deltaE.^2)).^2);
    cW = 2./((1+(.014.*deltaW.^2)).^2);

  elseif option == 8   % forwardbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=2;
    alfa=kf./(2.*(kb+w));

    cN =(1./(1+((deltaN./kf).^n))) - (alfa./(1+(((deltaN-kb)./w).^(2*m))));
    cS =(1./(1+((deltaS./kf).^n))) - (alfa./(1+(((deltaS-kb)./w).^(2*m))));
    cE =(1./(1+((deltaE./kf).^n))) - (alfa./(1+(((deltaE-kb)./w).^(2*m))));
    cW =(1./(1+((deltaW./kf).^n))) - (alfa./(1+(((deltaW-kb)./w).^(2*m))));

  elseif option == 9  % side

    cN = (exp(-((((abs(deltaN)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaN)))));
    cS = (exp(-((((abs(deltaS)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaS)))));
    cE = (exp(-((((abs(deltaE)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaE)))));
    cW = (exp(-((((abs(deltaW)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaW)))));

    cN((deltaN)==0)=0;
```

```
    cS((deltaS)==0)=0;
    cE((deltaE)==0)=0;
    cW((deltaW)==0)=0;


   elseif option == 10  % c2forwardandbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=1;
    alfa=kf./(2.*kb);

    cN = zeros(size(deltaN));
    cS = zeros(size(deltaS));
    cE = zeros(size(deltaE));
    cW = zeros(size(deltaW));


    [i,j]=size(deltaN);
    [i,j]=size(deltaS);
    [i,j]=size(deltaE);
    [i,j]=size(deltaW);


for i = 1:i, ;
    for j =1:j, ;
         if deltaN(i,j) <= kf;
             if deltaN(i,j)  >= 0;
             cN(i,j) = 1-((deltaN(i,j)./kf).^n);
              end
               elseif deltaN(i,j) <= (kb+w);
                   if deltaN(i,j) >= (kb-w);
                    cN(i,j)=alfa.*((((deltaN(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
 end

for i = 1:i, ;
    for j =1:j, ;
         if deltaS(i,j) <= kf;
             if deltaS(i,j)  >= 0;
             cS(i,j) = 1-((deltaS(i,j)./kf).^n);
              end
               elseif deltaS(i,j) <= (kb+w);
                   if deltaS(i,j) >= (kb-w);
                    cS(i,j)=alfa.*((((deltaS(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
 end
```

```
for i = 1:i, ;
    for j =1:j, ;
         if deltaE(i,j) <= kf;
             if deltaE(i,j)  >= 0;
             cE(i,j) = 1-((deltaE(i,j)./kf).^n);
              end
                elseif deltaE(i,j) <= (kb+w);
                    if deltaE(i,j) >= (kb-w);
                     cE(i,j)=alfa.*((((deltaE(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end


for i = 1:i, ;
    for j =1:j, ;
         if deltaW(i,j) <= kf;
             if deltaW(i,j)  >= 0;
             cW(i,j) = 1-((deltaW(i,j)./kf).^n);
              end
                elseif deltaW(i,j) <= (kb+w);
                    if deltaW(i,j) >= (kb-w);
                     cW(i,j)=alfa.*((((deltaW(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end


72

 end
```

```
%diff = diff + lambda*(cE.*deltaE + cW.*deltaW + cN.*deltaN + cS.*deltaS);
diff = diff + lambda*(cE.*deltaE + cW.*deltaW );
%deltaN
%deltaS

s=i;% backup i
diff1=diff;
h=length(diff1);
for i = 2:h-1;

%    if  abs((diff1(i)-diff1(i-1))) < (kappa);

%         if  abs((diff1(i)-diff1(i+1))) < (kappa);
```

```
            %diff(i)=median([diff1(i-1) diff1(i)  diff1(i+1)]);
%          end
%    end
end
gout=diff;
ee=(fft(gout))/length(gout);
dd=ee(:,(200):(pontos/2));

ddd=ee(:,(2):(200));
vetor(1,s)=(sum(abs(ddd)) - sum(abs(dd)));
erro(1,s)=sqrt((sum((im-io).^2))/(sum((diff-io).^2)));
NMSE(1,s)=(sum((diff-io).^2))/ (sum(io.^2));
end
fprintf('\n');
```

```
function [diff,vetor,erro,NMSE] = filtroW3m(im,niterations, kappa, lambda, option,p,i,↙
d,pontos,io,ganho)

Kp=p;
Ki=i;
Kd=d;




im = double(im);
[rows,cols] = size(im);
diff = im;

integral_erroN= [diff*0];
integral_erroS= [diff*0];
integral_erroE= [diff*0];
integral_erroW= [diff*0];
erroN_anterior= [diff*0];
erroS_anterior= [diff*0];
erroE_anterior= [diff*0];
erroW_anterior= [diff*0];
std_diff=std(diff);
std_diff_anterior=std(diff);
for i = 1:niterations




  diffl = zeros(rows+2, cols+2);
  diffl(2:rows+1, 2:cols+1) = diff;


  deltaNa = diffl(1:rows,2:cols+1) - diff;
  deltaSa = diffl(3:rows+2,2:cols+1) - diff;
  deltaEa = diffl(2:rows+1,3:cols+2) - diff;
  deltaWa = diffl(2:rows+1,1:cols) - diff;

  derivada_erroNa = deltaNa - erroN_anterior;
  derivada_erroSa = deltaSa - erroS_anterior;
  derivada_erroEa = deltaEa - erroE_anterior;
  derivada_erroWa = deltaWa - erroW_anterior;

  erroN_anterior = deltaNa; %GUARDA ERRO ANTERIOR
  erroS_anterior = deltaSa;
  erroE_anterior = deltaEa;
  erroW_anterior = deltaWa;

  integral_erroN = deltaNa + integral_erroN;
  integral_erroS = deltaSa + integral_erroS;
  integral_erroE = deltaEa + integral_erroE;
  integral_erroW = deltaWa + integral_erroW;

  deltaN = ((Kp*deltaNa)+(Ki*integral_erroN)+(Kd*derivada_erroNa));
```

```
deltaS = ((Kp*deltaSa)+(Ki*integral_erroS)+(Kd*derivada_erroSa));
deltaE = ((Kp*deltaEa)+(Ki*integral_erroE)+(Kd*derivada_erroEa));
deltaW = ((Kp*deltaWa)+(Ki*integral_erroW)+(Kd*derivada_erroWa));


if option == 1 %perona 1

  cN = 1./(1+(((deltaN.^2))/((kappa.^2))));
  cS = 1./(1+(((deltaS.^2))/((kappa.^2))));
  cE = 1./(1+(((deltaE.^2))/((kappa.^2))));
  cW = 1./(1+(((deltaW.^2))/((kappa.^2))));

elseif option == 2 % perona 2

  cN = exp(-(((deltaN.^2)/(2*(kappa.^2)))));
  cS = exp(-(((deltaS.^2)/(2*(kappa.^2)))));
  cE = exp(-(((deltaE.^2)/(2*(kappa.^2)))));
  cW = exp(-(((deltaW.^2)/(2*(kappa.^2)))));

elseif option == 3 % degrau

  cN = deltaN./deltaN;
  cS = deltaS./deltaS;
  cE = deltaE./deltaE;
  cW = deltaW./deltaW;

  cN((deltaN.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cS((deltaS.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cE((deltaE.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cW((deltaW.^2./(sqrt(2)*kappa).^2)>=0.5)=0;

  cN((deltaN.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cS((deltaS.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cE((deltaE.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cW((deltaW.^2./(sqrt(2)*kappa).^2)<0.5)=1;

elseif option == 4  % tukey

  cN = (1-((deltaN.^2)/(5*(kappa.^2)))).^2;
  cS = (1-((deltaS.^2)/(5*(kappa.^2)))).^2;
  cE = (1-((deltaE.^2)/(5*(kappa.^2)))).^2;
  cW = (1-((deltaW.^2)/(5*(kappa.^2)))).^2;

  cN(abs((deltaN.^2)/5)>((kappa.^2)))=0;
  cS(abs((deltaS.^2)/5)>((kappa.^2)))=0;
  cE(abs((deltaE.^2)/5)>((kappa.^2)))=0;
  cW(abs((deltaW.^2)/5)>((kappa.^2)))=0;

elseif option == 5 % huber

  cN=abs(kappa./(deltaN.*3.7));
  cS=abs(kappa./(deltaS.*3.7));
  cE=abs(kappa./(deltaE.*3.7));
  cW=abs(kappa./(deltaW.*3.7));

  cN(abs(deltaN.*3.7)<=kappa)=1;
```

```
    cS(abs(deltaS.*3.7)<=kappa)=1;
    cE(abs(deltaE.*3.7)<=kappa)=1;
    cW(abs(deltaW.*3.7)<=kappa)=1;

  elseif option == 6 % L1 norm

    cN = deltaN./deltaN;
    cS = deltaS./deltaS;
    cE = deltaE./deltaE;
    cW = deltaW./deltaW;

    cN((deltaN)<0)=-1;
    cS((deltaS)<0)=-1;
    cE((deltaE)<0)=-1;
    cW((deltaW)<0)=-1;

    cN((deltaN)>0)=1;
    cS((deltaS)>0)=1;
    cE((deltaE)>0)=1;
    cW((deltaW)>0)=1;



  elseif option == 7  % geman and mcclure

    cN = 2./((1+(.014.*deltaN.^2)).^2);
    cS = 2./((1+(.014.*deltaS.^2)).^2);
    cE = 2./((1+(.014.*deltaE.^2)).^2);
    cW = 2./((1+(.014.*deltaW.^2)).^2);

  elseif option == 8   % forwardbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=2;
    alfa=kf./(2.*(kb+w));

    cN =(1./(1+((deltaN./kf).^n))) - (alfa./(1+(((deltaN-kb)./w).^(2*m))));
    cS =(1./(1+((deltaS./kf).^n))) - (alfa./(1+(((deltaS-kb)./w).^(2*m))));
    cE =(1./(1+((deltaE./kf).^n))) - (alfa./(1+(((deltaE-kb)./w).^(2*m))));
    cW =(1./(1+((deltaW./kf).^n))) - (alfa./(1+(((deltaW-kb)./w).^(2*m))));

  elseif option == 9  % side

    cN = (exp(-((((abs(deltaN)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaN)))));
    cS = (exp(-((((abs(deltaS)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaS)))));
    cE = (exp(-((((abs(deltaE)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaE)))));
    cW = (exp(-((((abs(deltaW)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaW)))));

    cN((deltaN)==0)=0;
```

```
    cS((deltaS)==0)=0;
    cE((deltaE)==0)=0;
    cW((deltaW)==0)=0;


   elseif option == 10  % c2forwardandbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=1;
    alfa=kf./(2.*kb);

    cN = zeros(size(deltaN));
    cS = zeros(size(deltaS));
    cE = zeros(size(deltaE));
    cW = zeros(size(deltaW));


    [i,j]=size(deltaN);
    [i,j]=size(deltaS);
    [i,j]=size(deltaE);
    [i,j]=size(deltaW);


for i = 1:i, ;
    for j =1:j, ;
         if deltaN(i,j) <= kf;
             if deltaN(i,j)  >= 0;
             cN(i,j) = 1-((deltaN(i,j)./kf).^n);
              end
               elseif deltaN(i,j) <= (kb+w);
                   if deltaN(i,j) >= (kb-w);
                    cN(i,j)=alfa.*((((deltaN(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
 end

for i = 1:i, ;
    for j =1:j, ;
         if deltaS(i,j) <= kf;
             if deltaS(i,j)  >= 0;
             cS(i,j) = 1-((deltaS(i,j)./kf).^n);
              end
               elseif deltaS(i,j) <= (kb+w);
                   if deltaS(i,j) >= (kb-w);
                    cS(i,j)=alfa.*((((deltaS(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
 end
```

```
for i = 1:i, ;
    for j =1:j, ;
         if deltaE(i,j) <= kf;
             if deltaE(i,j)  >= 0;
             cE(i,j) = 1-((deltaE(i,j)./kf).^n);
              end
                elseif deltaE(i,j) <= (kb+w);
                    if deltaE(i,j) >= (kb-w);
                     cE(i,j)=alfa.*((((deltaE(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end


for i = 1:i, ;
    for j =1:j, ;
         if deltaW(i,j) <= kf;
             if deltaW(i,j)  >= 0;
             cW(i,j) = 1-((deltaW(i,j)./kf).^n);
              end
                elseif deltaW(i,j) <= (kb+w);
                    if deltaW(i,j) >= (kb-w);
                     cW(i,j)=alfa.*((((deltaW(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end


72

 end







%diff = diff + lambda*(cE.*deltaE + cW.*deltaW + cN.*deltaN + cS.*deltaS);
diff = diff + lambda*(cE.*deltaE + cW.*deltaW );
%deltaN
%deltaS

s=i;% backup i
diff1=diff;
h=length(diff1);
for i = 2:h-1;

%    if  abs((diff1(i)-diff1(i-1))) < (kappa);

%         if  abs((diff1(i)-diff1(i+1))) < (kappa);
```

```
              diff(i)=median([diff1(i-1) diff1(i)  diff1(i+1)]);
%          end
%    end
end
gout=diff;
ee=abs(fft(gout))/length(gout);
dd=ee(:,(pontos/9):(pontos/2));

ddd=ee(:,(pontos/720):(pontos/9));
vetor(1,s)=((sum((ddd))))- ((sum((dd))));
erro(1,s)=sqrt((sum((im-io).^2))/(sum((diff-io).^2)));
NMSE(1,s)=(sum((diff-io).^2))/ (sum(io.^2));
end
fprintf('\n');
```

```
function [diff,vetor,erro,NMSE] = filtroW5m(im,niterations, kappa, lambda, option,p,i,↙
d,pontos,io,ganho)

Kp=p;
Ki=i;
Kd=d;




im = double(im);
[rows,cols] = size(im);
diff = im;

integral_erroN= [diff*0];
integral_erroS= [diff*0];
integral_erroE= [diff*0];
integral_erroW= [diff*0];
erroN_anterior= [diff*0];
erroS_anterior= [diff*0];
erroE_anterior= [diff*0];
erroW_anterior= [diff*0];
std_diff=std(diff);
std_diff_anterior=std(diff);
for i = 1:niterations




  diffl = zeros(rows+2, cols+2);
  diffl(2:rows+1, 2:cols+1) = diff;


  deltaNa = diffl(1:rows,2:cols+1) - diff;
  deltaSa = diffl(3:rows+2,2:cols+1) - diff;
  deltaEa = diffl(2:rows+1,3:cols+2) - diff;
  deltaWa = diffl(2:rows+1,1:cols) - diff;

  derivada_erroNa = deltaNa - erroN_anterior;
  derivada_erroSa = deltaSa - erroS_anterior;
  derivada_erroEa = deltaEa - erroE_anterior;
  derivada_erroWa = deltaWa - erroW_anterior;

  erroN_anterior = deltaNa; %GUARDA ERRO ANTERIOR
  erroS_anterior = deltaSa;
  erroE_anterior = deltaEa;
  erroW_anterior = deltaWa;

  integral_erroN = deltaNa + integral_erroN;
  integral_erroS = deltaSa + integral_erroS;
  integral_erroE = deltaEa + integral_erroE;
  integral_erroW = deltaWa + integral_erroW;

  deltaN = ((Kp*deltaNa)+(Ki*integral_erroN)+(Kd*derivada_erroNa));
```

```
deltaS = ((Kp*deltaSa)+(Ki*integral_erroS)+(Kd*derivada_erroSa));
deltaE = ((Kp*deltaEa)+(Ki*integral_erroE)+(Kd*derivada_erroEa));
deltaW = ((Kp*deltaWa)+(Ki*integral_erroW)+(Kd*derivada_erroWa));


if option == 1 %perona 1

  cN = 1./(1+(((deltaN.^2))/((kappa.^2))));
  cS = 1./(1+(((deltaS.^2))/((kappa.^2))));
  cE = 1./(1+(((deltaE.^2))/((kappa.^2))));
  cW = 1./(1+(((deltaW.^2))/((kappa.^2))));

elseif option == 2 % perona 2

  cN = exp(-(((deltaN.^2)/(2*(kappa.^2)))));
  cS = exp(-(((deltaS.^2)/(2*(kappa.^2)))));
  cE = exp(-(((deltaE.^2)/(2*(kappa.^2)))));
  cW = exp(-(((deltaW.^2)/(2*(kappa.^2)))));

elseif option == 3 % degrau

  cN = deltaN./deltaN;
  cS = deltaS./deltaS;
  cE = deltaE./deltaE;
  cW = deltaW./deltaW;

  cN((deltaN.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cS((deltaS.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cE((deltaE.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cW((deltaW.^2./(sqrt(2)*kappa).^2)>=0.5)=0;

  cN((deltaN.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cS((deltaS.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cE((deltaE.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cW((deltaW.^2./(sqrt(2)*kappa).^2)<0.5)=1;

elseif option == 4  % tukey

  cN = (1-((deltaN.^2)/(5*(kappa.^2)))).^2;
  cS = (1-((deltaS.^2)/(5*(kappa.^2)))).^2;
  cE = (1-((deltaE.^2)/(5*(kappa.^2)))).^2;
  cW = (1-((deltaW.^2)/(5*(kappa.^2)))).^2;

  cN(abs((deltaN.^2)/5)>((kappa.^2)))=0;
  cS(abs((deltaS.^2)/5)>((kappa.^2)))=0;
  cE(abs((deltaE.^2)/5)>((kappa.^2)))=0;
  cW(abs((deltaW.^2)/5)>((kappa.^2)))=0;

elseif option == 5 % huber

  cN=abs(kappa./(deltaN.*3.7));
  cS=abs(kappa./(deltaS.*3.7));
  cE=abs(kappa./(deltaE.*3.7));
  cW=abs(kappa./(deltaW.*3.7));

  cN(abs(deltaN.*3.7)<=kappa)=1;
```

```
    cS(abs(deltaS.*3.7)<=kappa)=1;
    cE(abs(deltaE.*3.7)<=kappa)=1;
    cW(abs(deltaW.*3.7)<=kappa)=1;

  elseif option == 6 % L1 norm

    cN = deltaN./deltaN;
    cS = deltaS./deltaS;
    cE = deltaE./deltaE;
    cW = deltaW./deltaW;

    cN((deltaN)<0)=-1;
    cS((deltaS)<0)=-1;
    cE((deltaE)<0)=-1;
    cW((deltaW)<0)=-1;

    cN((deltaN)>0)=1;
    cS((deltaS)>0)=1;
    cE((deltaE)>0)=1;
    cW((deltaW)>0)=1;



  elseif option == 7  % geman and mcclure

    cN = 2./((1+(.014.*deltaN.^2)).^2);
    cS = 2./((1+(.014.*deltaS.^2)).^2);
    cE = 2./((1+(.014.*deltaE.^2)).^2);
    cW = 2./((1+(.014.*deltaW.^2)).^2);

  elseif option == 8   % forwardbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=2;
    alfa=kf./(2.*(kb+w));

    cN =(1./(1+((deltaN./kf).^n))) - (alfa./(1+(((deltaN-kb)./w).^(2*m))));
    cS =(1./(1+((deltaS./kf).^n))) - (alfa./(1+(((deltaS-kb)./w).^(2*m))));
    cE =(1./(1+((deltaE./kf).^n))) - (alfa./(1+(((deltaE-kb)./w).^(2*m))));
    cW =(1./(1+((deltaW./kf).^n))) - (alfa./(1+(((deltaW-kb)./w).^(2*m))));

  elseif option == 9  % side

    cN = (exp(-((((abs(deltaN)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaN)))));
    cS = (exp(-((((abs(deltaS)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaS)))));
    cE = (exp(-((((abs(deltaE)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaE)))));
    cW = (exp(-((((abs(deltaW)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaW)))));

    cN((deltaN)==0)=0;
```

```
    cS((deltaS)==0)=0;
    cE((deltaE)==0)=0;
    cW((deltaW)==0)=0;


   elseif option == 10  % c2forwardandbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=1;
    alfa=kf./(2.*kb);

    cN = zeros(size(deltaN));
    cS = zeros(size(deltaS));
    cE = zeros(size(deltaE));
    cW = zeros(size(deltaW));


    [i,j]=size(deltaN);
    [i,j]=size(deltaS);
    [i,j]=size(deltaE);
    [i,j]=size(deltaW);


for i = 1:i, ;
    for j =1:j, ;
         if deltaN(i,j) <= kf;
             if deltaN(i,j)  >= 0;
             cN(i,j) = 1-((deltaN(i,j)./kf).^n);
              end
               elseif deltaN(i,j) <= (kb+w);
                   if deltaN(i,j) >= (kb-w);
                    cN(i,j)=alfa.*((((deltaN(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
 end

for i = 1:i, ;
    for j =1:j, ;
         if deltaS(i,j) <= kf;
             if deltaS(i,j)  >= 0;
             cS(i,j) = 1-((deltaS(i,j)./kf).^n);
              end
               elseif deltaS(i,j) <= (kb+w);
                   if deltaS(i,j) >= (kb-w);
                    cS(i,j)=alfa.*((((deltaS(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
 end
```

```
for i = 1:i, ;
    for j =1:j, ;
         if deltaE(i,j) <= kf;
             if deltaE(i,j)  >= 0;
             cE(i,j) = 1-((deltaE(i,j)./kf).^n);
              end
                elseif deltaE(i,j) <= (kb+w);
                    if deltaE(i,j) >= (kb-w);
                     cE(i,j)=alfa.*((((deltaE(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end


for i = 1:i, ;
    for j =1:j, ;
         if deltaW(i,j) <= kf;
             if deltaW(i,j)  >= 0;
             cW(i,j) = 1-((deltaW(i,j)./kf).^n);
              end
                elseif deltaW(i,j) <= (kb+w);
                    if deltaW(i,j) >= (kb-w);
                     cW(i,j)=alfa.*((((deltaW(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end


72

 end
```

```
%diff = diff + lambda*(cE.*deltaE + cW.*deltaW + cN.*deltaN + cS.*deltaS);
diff = diff + lambda*(cE.*deltaE + cW.*deltaW );
%deltaN
%deltaS

s=i;% backup i
diff1=diff;
h=length(diff1);
for i = 3:h-2;

%    if  abs((diff1(i)-diff1(i-1))) < (kappa);

%         if  abs((diff1(i)-diff1(i+1))) < (kappa);
```

```
               diff(i)=median([ diff1(i-2) diff1(i-1) diff1(i)   diff1(i+1) diff1(i+2)↙
]);
%          end
%    end
end
gout=diff;
ee=abs(fft(gout))/length(gout);
dd=ee(:,(pontos/9):(pontos/2));

ddd=ee(:,(pontos/720):(pontos/9));
vetor(1,s)=((sum((ddd))))- ((sum((dd))));
erro(1,s)=sqrt((sum((im-io).^2))/(sum((diff-io).^2)));
NMSE(1,s)=(sum((diff-io).^2))/ (sum(io.^2));
end
fprintf('\n');
```

```
function [diff,vetor,erro,NMSE] = filtroW7m(im,niterations, kappa, lambda, option,p,i,↙
d,pontos,io,ganho)

Kp=p;
Ki=i;
Kd=d;




im = double(im);
[rows,cols] = size(im);
diff = im;

integral_erroN= [diff*0];
integral_erroS= [diff*0];
integral_erroE= [diff*0];
integral_erroW= [diff*0];
erroN_anterior= [diff*0];
erroS_anterior= [diff*0];
erroE_anterior= [diff*0];
erroW_anterior= [diff*0];
std_diff=std(diff);
std_diff_anterior=std(diff);
for i = 1:niterations




  diffl = zeros(rows+2, cols+2);
  diffl(2:rows+1, 2:cols+1) = diff;


  deltaNa = diffl(1:rows,2:cols+1) - diff;
  deltaSa = diffl(3:rows+2,2:cols+1) - diff;
  deltaEa = diffl(2:rows+1,3:cols+2) - diff;
  deltaWa = diffl(2:rows+1,1:cols) - diff;

  derivada_erroNa = deltaNa - erroN_anterior;
  derivada_erroSa = deltaSa - erroS_anterior;
  derivada_erroEa = deltaEa - erroE_anterior;
  derivada_erroWa = deltaWa - erroW_anterior;

  erroN_anterior = deltaNa; %GUARDA ERRO ANTERIOR
  erroS_anterior = deltaSa;
  erroE_anterior = deltaEa;
  erroW_anterior = deltaWa;

  integral_erroN = deltaNa + integral_erroN;
  integral_erroS = deltaSa + integral_erroS;
  integral_erroE = deltaEa + integral_erroE;
  integral_erroW = deltaWa + integral_erroW;

  deltaN = ((Kp*deltaNa)+(Ki*integral_erroN)+(Kd*derivada_erroNa));
```

```
deltaS = ((Kp*deltaSa)+(Ki*integral_erroS)+(Kd*derivada_erroSa));
deltaE = ((Kp*deltaEa)+(Ki*integral_erroE)+(Kd*derivada_erroEa));
deltaW = ((Kp*deltaWa)+(Ki*integral_erroW)+(Kd*derivada_erroWa));


if option == 1 %perona 1

  cN = 1./(1+(((deltaN.^2))/((kappa.^2))));
  cS = 1./(1+(((deltaS.^2))/((kappa.^2))));
  cE = 1./(1+(((deltaE.^2))/((kappa.^2))));
  cW = 1./(1+(((deltaW.^2))/((kappa.^2))));

elseif option == 2 % perona 2

  cN = exp(-(((deltaN.^2)/(2*(kappa.^2)))));
  cS = exp(-(((deltaS.^2)/(2*(kappa.^2)))));
  cE = exp(-(((deltaE.^2)/(2*(kappa.^2)))));
  cW = exp(-(((deltaW.^2)/(2*(kappa.^2)))));

elseif option == 3 % degrau

  cN = deltaN./deltaN;
  cS = deltaS./deltaS;
  cE = deltaE./deltaE;
  cW = deltaW./deltaW;

  cN((deltaN.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cS((deltaS.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cE((deltaE.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cW((deltaW.^2./(sqrt(2)*kappa).^2)>=0.5)=0;

  cN((deltaN.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cS((deltaS.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cE((deltaE.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cW((deltaW.^2./(sqrt(2)*kappa).^2)<0.5)=1;

elseif option == 4  % tukey

  cN = (1-((deltaN.^2)/(5*(kappa.^2)))).^2;
  cS = (1-((deltaS.^2)/(5*(kappa.^2)))).^2;
  cE = (1-((deltaE.^2)/(5*(kappa.^2)))).^2;
  cW = (1-((deltaW.^2)/(5*(kappa.^2)))).^2;

  cN(abs((deltaN.^2)/5)>((kappa.^2)))=0;
  cS(abs((deltaS.^2)/5)>((kappa.^2)))=0;
  cE(abs((deltaE.^2)/5)>((kappa.^2)))=0;
  cW(abs((deltaW.^2)/5)>((kappa.^2)))=0;

elseif option == 5 % huber

  cN=abs(kappa./(deltaN.*3.7));
  cS=abs(kappa./(deltaS.*3.7));
  cE=abs(kappa./(deltaE.*3.7));
  cW=abs(kappa./(deltaW.*3.7));

  cN(abs(deltaN.*3.7)<=kappa)=1;
```

```
    cS(abs(deltaS.*3.7)<=kappa)=1;
    cE(abs(deltaE.*3.7)<=kappa)=1;
    cW(abs(deltaW.*3.7)<=kappa)=1;

  elseif option == 6 % L1 norm

    cN = deltaN./deltaN;
    cS = deltaS./deltaS;
    cE = deltaE./deltaE;
    cW = deltaW./deltaW;

    cN((deltaN)<0)=-1;
    cS((deltaS)<0)=-1;
    cE((deltaE)<0)=-1;
    cW((deltaW)<0)=-1;

    cN((deltaN)>0)=1;
    cS((deltaS)>0)=1;
    cE((deltaE)>0)=1;
    cW((deltaW)>0)=1;



  elseif option == 7  % geman and mcclure

    cN = 2./((1+(.014.*deltaN.^2)).^2);
    cS = 2./((1+(.014.*deltaS.^2)).^2);
    cE = 2./((1+(.014.*deltaE.^2)).^2);
    cW = 2./((1+(.014.*deltaW.^2)).^2);

  elseif option == 8   % forwardbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=2;
    alfa=kf./(2.*(kb+w));

    cN =(1./(1+((deltaN./kf).^n))) - (alfa./(1+(((deltaN-kb)./w).^(2*m))));
    cS =(1./(1+((deltaS./kf).^n))) - (alfa./(1+(((deltaS-kb)./w).^(2*m))));
    cE =(1./(1+((deltaE./kf).^n))) - (alfa./(1+(((deltaE-kb)./w).^(2*m))));
    cW =(1./(1+((deltaW./kf).^n))) - (alfa./(1+(((deltaW-kb)./w).^(2*m))));

  elseif option == 9  % side

    cN = (exp(-((((abs(deltaN)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaN)))));
    cS = (exp(-((((abs(deltaS)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaS)))));
    cE = (exp(-((((abs(deltaE)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaE)))));
    cW = (exp(-((((abs(deltaW)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaW)))));

    cN((deltaN)==0)=0;
```

```
    cS((deltaS)==0)=0;
    cE((deltaE)==0)=0;
    cW((deltaW)==0)=0;


   elseif option == 10  % c2forwardandbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=1;
    alfa=kf./(2.*kb);

    cN = zeros(size(deltaN));
    cS = zeros(size(deltaS));
    cE = zeros(size(deltaE));
    cW = zeros(size(deltaW));


    [i,j]=size(deltaN);
    [i,j]=size(deltaS);
    [i,j]=size(deltaE);
    [i,j]=size(deltaW);


for i = 1:i, ;
    for j =1:j, ;
         if deltaN(i,j) <= kf;
             if deltaN(i,j)  >= 0;
             cN(i,j) = 1-((deltaN(i,j)./kf).^n);
              end
               elseif deltaN(i,j) <= (kb+w);
                   if deltaN(i,j) >= (kb-w);
                    cN(i,j)=alfa.*((((deltaN(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
 end

for i = 1:i, ;
    for j =1:j, ;
         if deltaS(i,j) <= kf;
             if deltaS(i,j)  >= 0;
             cS(i,j) = 1-((deltaS(i,j)./kf).^n);
              end
               elseif deltaS(i,j) <= (kb+w);
                   if deltaS(i,j) >= (kb-w);
                    cS(i,j)=alfa.*((((deltaS(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
 end
```

```
for i = 1:i, ;
    for j =1:j, ;
         if deltaE(i,j) <= kf;
             if deltaE(i,j)  >= 0;
             cE(i,j) = 1-((deltaE(i,j)./kf).^n);
              end
                elseif deltaE(i,j) <= (kb+w);
                    if deltaE(i,j) >= (kb-w);
                     cE(i,j)=alfa.*((((deltaE(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end


for i = 1:i, ;
    for j =1:j, ;
         if deltaW(i,j) <= kf;
             if deltaW(i,j)  >= 0;
             cW(i,j) = 1-((deltaW(i,j)./kf).^n);
              end
                elseif deltaW(i,j) <= (kb+w);
                    if deltaW(i,j) >= (kb-w);
                     cW(i,j)=alfa.*((((deltaW(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end


72

 end







%diff = diff + lambda*(cE.*deltaE + cW.*deltaW + cN.*deltaN + cS.*deltaS);
diff = diff + lambda*(cE.*deltaE + cW.*deltaW );
%deltaN
%deltaS

s=i;% backup i
diff1=diff;
h=length(diff1);
for i = 4:h-3;

%    if  abs((diff1(i)-diff1(i-1))) < (kappa);

%         if  abs((diff1(i)-diff1(i+1))) < (kappa);
```

```
                diff(i)=median([diff1(i-3) diff1(i-2) diff1(i-1) diff1(i)   diff1(i+1)↙
diff1(i+2) diff1(i+3) ]);
%          end
%    end
end
gout=diff;
ee=abs(fft(gout))/length(gout);
dd=ee(:,(pontos/9):(pontos/2));

ddd=ee(:,(pontos/720):(pontos/9));
vetor(1,s)=((sum((ddd))))- ((sum((dd))));
erro(1,s)=sqrt((sum((im-io).^2))/(sum((diff-io).^2)));
NMSE(1,s)=(sum((diff-io).^2))/ (sum(io.^2));
end
fprintf('\n');
```

```
function [diff,vetor,erro,NMSE] = filtroW9m(im,niterations, kappa, lambda, option,p,i,↙
d,pontos,io,ganho)

Kp=p;
Ki=i;
Kd=d;




im = double(im);
[rows,cols] = size(im);
diff = im;

integral_erroN= [diff*0];
integral_erroS= [diff*0];
integral_erroE= [diff*0];
integral_erroW= [diff*0];
erroN_anterior= [diff*0];
erroS_anterior= [diff*0];
erroE_anterior= [diff*0];
erroW_anterior= [diff*0];
std_diff=std(diff);
std_diff_anterior=std(diff);
for i = 1:niterations




  diffl = zeros(rows+2, cols+2);
  diffl(2:rows+1, 2:cols+1) = diff;


  deltaNa = diffl(1:rows,2:cols+1) - diff;
  deltaSa = diffl(3:rows+2,2:cols+1) - diff;
  deltaEa = diffl(2:rows+1,3:cols+2) - diff;
  deltaWa = diffl(2:rows+1,1:cols) - diff;

  derivada_erroNa = deltaNa - erroN_anterior;
  derivada_erroSa = deltaSa - erroS_anterior;
  derivada_erroEa = deltaEa - erroE_anterior;
  derivada_erroWa = deltaWa - erroW_anterior;

  erroN_anterior = deltaNa; %GUARDA ERRO ANTERIOR
  erroS_anterior = deltaSa;
  erroE_anterior = deltaEa;
  erroW_anterior = deltaWa;

  integral_erroN = deltaNa + integral_erroN;
  integral_erroS = deltaSa + integral_erroS;
  integral_erroE = deltaEa + integral_erroE;
  integral_erroW = deltaWa + integral_erroW;

  deltaN = ((Kp*deltaNa)+(Ki*integral_erroN)+(Kd*derivada_erroNa));
```

```
deltaS = ((Kp*deltaSa)+(Ki*integral_erroS)+(Kd*derivada_erroSa));
deltaE = ((Kp*deltaEa)+(Ki*integral_erroE)+(Kd*derivada_erroEa));
deltaW = ((Kp*deltaWa)+(Ki*integral_erroW)+(Kd*derivada_erroWa));


if option == 1 %perona 1

  cN = 1./(1+(((deltaN.^2))/((kappa.^2))));
  cS = 1./(1+(((deltaS.^2))/((kappa.^2))));
  cE = 1./(1+(((deltaE.^2))/((kappa.^2))));
  cW = 1./(1+(((deltaW.^2))/((kappa.^2))));

elseif option == 2 % perona 2

  cN = exp(-(((deltaN.^2)/(2*(kappa.^2)))));
  cS = exp(-(((deltaS.^2)/(2*(kappa.^2)))));
  cE = exp(-(((deltaE.^2)/(2*(kappa.^2)))));
  cW = exp(-(((deltaW.^2)/(2*(kappa.^2)))));

elseif option == 3 % degrau

  cN = deltaN./deltaN;
  cS = deltaS./deltaS;
  cE = deltaE./deltaE;
  cW = deltaW./deltaW;

  cN((deltaN.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cS((deltaS.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cE((deltaE.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cW((deltaW.^2./(sqrt(2)*kappa).^2)>=0.5)=0;

  cN((deltaN.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cS((deltaS.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cE((deltaE.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cW((deltaW.^2./(sqrt(2)*kappa).^2)<0.5)=1;

elseif option == 4  % tukey

  cN = (1-((deltaN.^2)/(5*(kappa.^2)))).^2;
  cS = (1-((deltaS.^2)/(5*(kappa.^2)))).^2;
  cE = (1-((deltaE.^2)/(5*(kappa.^2)))).^2;
  cW = (1-((deltaW.^2)/(5*(kappa.^2)))).^2;

  cN(abs((deltaN.^2)/5)>((kappa.^2)))=0;
  cS(abs((deltaS.^2)/5)>((kappa.^2)))=0;
  cE(abs((deltaE.^2)/5)>((kappa.^2)))=0;
  cW(abs((deltaW.^2)/5)>((kappa.^2)))=0;

elseif option == 5 % huber

  cN=abs(kappa./(deltaN.*3.7));
  cS=abs(kappa./(deltaS.*3.7));
  cE=abs(kappa./(deltaE.*3.7));
  cW=abs(kappa./(deltaW.*3.7));

  cN(abs(deltaN.*3.7)<=kappa)=1;
```

```
    cS(abs(deltaS.*3.7)<=kappa)=1;
    cE(abs(deltaE.*3.7)<=kappa)=1;
    cW(abs(deltaW.*3.7)<=kappa)=1;

  elseif option == 6 % L1 norm

    cN = deltaN./deltaN;
    cS = deltaS./deltaS;
    cE = deltaE./deltaE;
    cW = deltaW./deltaW;

    cN((deltaN)<0)=-1;
    cS((deltaS)<0)=-1;
    cE((deltaE)<0)=-1;
    cW((deltaW)<0)=-1;

    cN((deltaN)>0)=1;
    cS((deltaS)>0)=1;
    cE((deltaE)>0)=1;
    cW((deltaW)>0)=1;



  elseif option == 7  % geman and mcclure

    cN = 2./((1+(.014.*deltaN.^2)).^2);
    cS = 2./((1+(.014.*deltaS.^2)).^2);
    cE = 2./((1+(.014.*deltaE.^2)).^2);
    cW = 2./((1+(.014.*deltaW.^2)).^2);

  elseif option == 8   % forwardbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=2;
    alfa=kf./(2.*(kb+w));

    cN =(1./(1+((deltaN./kf).^n))) - (alfa./(1+(((deltaN-kb)./w).^(2*m))));
    cS =(1./(1+((deltaS./kf).^n))) - (alfa./(1+(((deltaS-kb)./w).^(2*m))));
    cE =(1./(1+((deltaE./kf).^n))) - (alfa./(1+(((deltaE-kb)./w).^(2*m))));
    cW =(1./(1+((deltaW./kf).^n))) - (alfa./(1+(((deltaW-kb)./w).^(2*m))));

  elseif option == 9  % side

    cN = (exp(-((((abs(deltaN)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaN)))));
    cS = (exp(-((((abs(deltaS)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaS)))));
    cE = (exp(-((((abs(deltaE)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaE)))));
    cW = (exp(-((((abs(deltaW)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaW)))));

    cN((deltaN)==0)=0;
```

```
    cS((deltaS)==0)=0;
    cE((deltaE)==0)=0;
    cW((deltaW)==0)=0;


   elseif option == 10  % c2forwardandbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=1;
    alfa=kf./(2.*kb);

    cN = zeros(size(deltaN));
    cS = zeros(size(deltaS));
    cE = zeros(size(deltaE));
    cW = zeros(size(deltaW));


    [i,j]=size(deltaN);
    [i,j]=size(deltaS);
    [i,j]=size(deltaE);
    [i,j]=size(deltaW);


for i = 1:i, ;
    for j =1:j, ;
         if deltaN(i,j) <= kf;
             if deltaN(i,j)  >= 0;
             cN(i,j) = 1-((deltaN(i,j)./kf).^n);
              end
               elseif deltaN(i,j) <= (kb+w);
                   if deltaN(i,j) >= (kb-w);
                    cN(i,j)=alfa.*((((deltaN(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
 end

for i = 1:i, ;
    for j =1:j, ;
         if deltaS(i,j) <= kf;
             if deltaS(i,j)  >= 0;
             cS(i,j) = 1-((deltaS(i,j)./kf).^n);
              end
               elseif deltaS(i,j) <= (kb+w);
                   if deltaS(i,j) >= (kb-w);
                    cS(i,j)=alfa.*((((deltaS(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
 end
```

```
for i = 1:i, ;
    for j =1:j, ;
         if deltaE(i,j) <= kf;
             if deltaE(i,j)  >= 0;
             cE(i,j) = 1-((deltaE(i,j)./kf).^n);
              end
                elseif deltaE(i,j) <= (kb+w);
                    if deltaE(i,j) >= (kb-w);
                     cE(i,j)=alfa.*((((deltaE(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end


for i = 1:i, ;
    for j =1:j, ;
         if deltaW(i,j) <= kf;
             if deltaW(i,j)  >= 0;
             cW(i,j) = 1-((deltaW(i,j)./kf).^n);
              end
                elseif deltaW(i,j) <= (kb+w);
                    if deltaW(i,j) >= (kb-w);
                     cW(i,j)=alfa.*((((deltaW(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end


72

 end







%diff = diff + lambda*(cE.*deltaE + cW.*deltaW + cN.*deltaN + cS.*deltaS);
diff = diff + lambda*(cE.*deltaE + cW.*deltaW );
%deltaN
%deltaS

s=i;% backup i
diff1=diff;
h=length(diff1);
for i = 5:h-4;

%    if  abs((diff1(i)-diff1(i-1))) < (kappa);

%         if  abs((diff1(i)-diff1(i+1))) < (kappa);
```

```
                diff(i)=median([diff1(i-4) diff1(i-3) diff1(i-2) diff1(i-1) diff1(i)↙
diff1(i+1) diff1(i+2) diff1(i+3) diff1(i+4)]);
%          end
%    end
end
gout=diff;
ee=abs(fft(gout))/length(gout);
dd=ee(:,(pontos/9):(pontos/2));

ddd=ee(:,(pontos/720):(pontos/9));
vetor(1,s)=((sum((ddd))))- ((sum((dd))));
erro(1,s)=sqrt((sum((im-io).^2))/(sum((diff-io).^2)));
NMSE(1,s)=(sum((diff-io).^2))/ (sum(io.^2));
end
fprintf('\n');
```

```
function [diff,vetor,erro,NMSE] = filtroW3mit(im,niterations, kappa, lambda, option,p,↙
i,d,pontos,io,ganho)

Kp=p;
Ki=i;
Kd=d;




im = double(im);
[rows,cols] = size(im);
diff = im;

integral_erroN= [diff*0];
integral_erroS= [diff*0];
integral_erroE= [diff*0];
integral_erroW= [diff*0];
erroN_anterior= [diff*0];
erroS_anterior= [diff*0];
erroE_anterior= [diff*0];
erroW_anterior= [diff*0];
std_diff=std(diff);
std_diff_anterior=std(diff);
for i = 1:niterations




  diffl = zeros(rows+2, cols+2);
  diffl(2:rows+1, 2:cols+1) = diff;


  deltaNa = diffl(1:rows,2:cols+1) - diff;
  deltaSa = diffl(3:rows+2,2:cols+1) - diff;
  deltaEa = diffl(2:rows+1,3:cols+2) - diff;
  deltaWa = diffl(2:rows+1,1:cols) - diff;

  derivada_erroNa = deltaNa - erroN_anterior;
  derivada_erroSa = deltaSa - erroS_anterior;
  derivada_erroEa = deltaEa - erroE_anterior;
  derivada_erroWa = deltaWa - erroW_anterior;

  erroN_anterior = deltaNa; %GUARDA ERRO ANTERIOR
  erroS_anterior = deltaSa;
  erroE_anterior = deltaEa;
  erroW_anterior = deltaWa;

  integral_erroN = deltaNa + integral_erroN;
  integral_erroS = deltaSa + integral_erroS;
  integral_erroE = deltaEa + integral_erroE;
  integral_erroW = deltaWa + integral_erroW;

  deltaN = ((Kp*deltaNa)+(Ki*integral_erroN)+(Kd*derivada_erroNa));
```

```
deltaS = ((Kp*deltaSa)+(Ki*integral_erroS)+(Kd*derivada_erroSa));
deltaE = ((Kp*deltaEa)+(Ki*integral_erroE)+(Kd*derivada_erroEa));
deltaW = ((Kp*deltaWa)+(Ki*integral_erroW)+(Kd*derivada_erroWa));


if option == 1 %perona 1

  cN = 1./(1+(((deltaN.^2))/((kappa.^2))));
  cS = 1./(1+(((deltaS.^2))/((kappa.^2))));
  cE = 1./(1+(((deltaE.^2))/((kappa.^2))));
  cW = 1./(1+(((deltaW.^2))/((kappa.^2))));

elseif option == 2 % perona 2

  cN = exp(-(((deltaN.^2)/(2*(kappa.^2)))));
  cS = exp(-(((deltaS.^2)/(2*(kappa.^2)))));
  cE = exp(-(((deltaE.^2)/(2*(kappa.^2)))));
  cW = exp(-(((deltaW.^2)/(2*(kappa.^2)))));

elseif option == 3 % degrau

  cN = deltaN./deltaN;
  cS = deltaS./deltaS;
  cE = deltaE./deltaE;
  cW = deltaW./deltaW;

  cN((deltaN.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cS((deltaS.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cE((deltaE.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cW((deltaW.^2./(sqrt(2)*kappa).^2)>=0.5)=0;

  cN((deltaN.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cS((deltaS.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cE((deltaE.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cW((deltaW.^2./(sqrt(2)*kappa).^2)<0.5)=1;

elseif option == 4  % tukey

  cN = (1-((deltaN.^2)/(5*(kappa.^2)))).^2;
  cS = (1-((deltaS.^2)/(5*(kappa.^2)))).^2;
  cE = (1-((deltaE.^2)/(5*(kappa.^2)))).^2;
  cW = (1-((deltaW.^2)/(5*(kappa.^2)))).^2;

  cN(abs((deltaN.^2)/5)>((kappa.^2)))=0;
  cS(abs((deltaS.^2)/5)>((kappa.^2)))=0;
  cE(abs((deltaE.^2)/5)>((kappa.^2)))=0;
  cW(abs((deltaW.^2)/5)>((kappa.^2)))=0;

elseif option == 5 % huber

  cN=abs(kappa./(deltaN.*3.7));
  cS=abs(kappa./(deltaS.*3.7));
  cE=abs(kappa./(deltaE.*3.7));
  cW=abs(kappa./(deltaW.*3.7));

  cN(abs(deltaN.*3.7)<=kappa)=1;
```

```
    cS(abs(deltaS.*3.7)<=kappa)=1;
    cE(abs(deltaE.*3.7)<=kappa)=1;
    cW(abs(deltaW.*3.7)<=kappa)=1;

  elseif option == 6 % L1 norm

    cN = deltaN./deltaN;
    cS = deltaS./deltaS;
    cE = deltaE./deltaE;
    cW = deltaW./deltaW;

    cN((deltaN)<0)=-1;
    cS((deltaS)<0)=-1;
    cE((deltaE)<0)=-1;
    cW((deltaW)<0)=-1;

    cN((deltaN)>0)=1;
    cS((deltaS)>0)=1;
    cE((deltaE)>0)=1;
    cW((deltaW)>0)=1;



  elseif option == 7  % geman and mcclure

    cN = 2./((1+(.014.*deltaN.^2)).^2);
    cS = 2./((1+(.014.*deltaS.^2)).^2);
    cE = 2./((1+(.014.*deltaE.^2)).^2);
    cW = 2./((1+(.014.*deltaW.^2)).^2);

  elseif option == 8   % forwardbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=2;
    alfa=kf./(2.*(kb+w));

    cN =(1./(1+((deltaN./kf).^n))) - (alfa./(1+(((deltaN-kb)./w).^(2*m))));
    cS =(1./(1+((deltaS./kf).^n))) - (alfa./(1+(((deltaS-kb)./w).^(2*m))));
    cE =(1./(1+((deltaE./kf).^n))) - (alfa./(1+(((deltaE-kb)./w).^(2*m))));
    cW =(1./(1+((deltaW./kf).^n))) - (alfa./(1+(((deltaW-kb)./w).^(2*m))));

  elseif option == 9  % side

    cN = (exp(-((((abs(deltaN)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaN)))));
    cS = (exp(-((((abs(deltaS)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaS)))));
    cE = (exp(-((((abs(deltaE)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaE)))));
    cW = (exp(-((((abs(deltaW)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaW)))));

    cN((deltaN)==0)=0;
```

```
    cS((deltaS)==0)=0;
    cE((deltaE)==0)=0;
    cW((deltaW)==0)=0;


   elseif option == 10  % c2forwardandbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=1;
    alfa=kf./(2.*kb);

    cN = zeros(size(deltaN));
    cS = zeros(size(deltaS));
    cE = zeros(size(deltaE));
    cW = zeros(size(deltaW));


    [i,j]=size(deltaN);
    [i,j]=size(deltaS);
    [i,j]=size(deltaE);
    [i,j]=size(deltaW);


for i = 1:i, ;
    for j =1:j, ;
         if deltaN(i,j) <= kf;
             if deltaN(i,j)  >= 0;
             cN(i,j) = 1-((deltaN(i,j)./kf).^n);
              end
               elseif deltaN(i,j) <= (kb+w);
                   if deltaN(i,j) >= (kb-w);
                    cN(i,j)=alfa.*((((deltaN(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
 end

for i = 1:i, ;
    for j =1:j, ;
         if deltaS(i,j) <= kf;
             if deltaS(i,j)  >= 0;
             cS(i,j) = 1-((deltaS(i,j)./kf).^n);
              end
               elseif deltaS(i,j) <= (kb+w);
                   if deltaS(i,j) >= (kb-w);
                    cS(i,j)=alfa.*((((deltaS(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
 end
```

```
for i = 1:i, ;
    for j =1:j, ;
         if deltaE(i,j) <= kf;
             if deltaE(i,j)  >= 0;
             cE(i,j) = 1-((deltaE(i,j)./kf).^n);
              end
                elseif deltaE(i,j) <= (kb+w);
                    if deltaE(i,j) >= (kb-w);
                     cE(i,j)=alfa.*((((deltaE(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end


for i = 1:i, ;
    for j =1:j, ;
         if deltaW(i,j) <= kf;
             if deltaW(i,j)  >= 0;
             cW(i,j) = 1-((deltaW(i,j)./kf).^n);
              end
                elseif deltaW(i,j) <= (kb+w);
                    if deltaW(i,j) >= (kb-w);
                     cW(i,j)=alfa.*((((deltaW(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end


72

 end
```

```
%diff = diff + lambda*(cE.*deltaE + cW.*deltaW + cN.*deltaN + cS.*deltaS);
diff = diff + lambda*(cE.*deltaE + cW.*deltaW );
%deltaN
%deltaS

s=i;% backup i
diff1=diff;
h=length(diff1);
for i = 2:h-1;

%    if  abs((diff1(i)-diff1(i-1))) < (kappa);

%         if  abs((diff1(i)-diff1(i+1))) < (kappa);
```

```
              diff(i)=median([diff1(i-1) diff1(i)  diff1(i+1)]);
%          end
%    end
end
gout=diff;
ee=abs(fft(gout))/length(gout);
dd=ee(:,(600):(pontos/2));

ddd=ee(:,(pontos/720):(600));
vetor(1,s)=((sum((ddd))))- ((sum((dd))));
erro(1,s)=sqrt((sum((im-io).^2))/(sum((diff-io).^2)));
NMSE(1,s)=(sum((diff-io).^2))/ (sum(io.^2));
end
fprintf('\n');
```

```
function [diff,vetor,erro,NMSE] = filtroW3p(im,niterations, kappa, lambda, option,p,i,↙
d,pontos,io,ganho)

Kp=p;
Ki=i;
Kd=d;



im = double(im);
[rows,cols] = size(im);
diff = im;

integral_erroN= [diff*0];
integral_erroS= [diff*0];
integral_erroE= [diff*0];
integral_erroW= [diff*0];
erroN_anterior= [diff*0];
erroS_anterior= [diff*0];
erroE_anterior= [diff*0];
erroW_anterior= [diff*0];
std_diff=std(diff);
std_diff_anterior=std(diff);
for i = 1:niterations



  diffl = zeros(rows+2, cols+2);
  diffl(2:rows+1, 2:cols+1) = diff;


  deltaNa = diffl(1:rows,2:cols+1) - diff;
  deltaSa = diffl(3:rows+2,2:cols+1) - diff;
  deltaEa = diffl(2:rows+1,3:cols+2) - diff;
  deltaWa = diffl(2:rows+1,1:cols) - diff;

  derivada_erroNa = deltaNa - erroN_anterior;
  derivada_erroSa = deltaSa - erroS_anterior;
  derivada_erroEa = deltaEa - erroE_anterior;
  derivada_erroWa = deltaWa - erroW_anterior;

  erroN_anterior = deltaNa; %GUARDA ERRO ANTERIOR
  erroS_anterior = deltaSa;
  erroE_anterior = deltaEa;
  erroW_anterior = deltaWa;

  integral_erroN = deltaNa + integral_erroN;
  integral_erroS = deltaSa + integral_erroS;
  integral_erroE = deltaEa + integral_erroE;
  integral_erroW = deltaWa + integral_erroW;

  deltaN = ((Kp*deltaNa)+(Ki*integral_erroN)+(Kd*derivada_erroNa));
```

```
deltaS = ((Kp*deltaSa)+(Ki*integral_erroS)+(Kd*derivada_erroSa));
deltaE = ((Kp*deltaEa)+(Ki*integral_erroE)+(Kd*derivada_erroEa));
deltaW = ((Kp*deltaWa)+(Ki*integral_erroW)+(Kd*derivada_erroWa));


if option == 1 %perona 1

  cN = 1./(1+(((deltaN.^2))/((kappa.^2))));
  cS = 1./(1+(((deltaS.^2))/((kappa.^2))));
  cE = 1./(1+(((deltaE.^2))/((kappa.^2))));
  cW = 1./(1+(((deltaW.^2))/((kappa.^2))));

elseif option == 2 % perona 2

  cN = exp(-(((deltaN.^2)/(2*(kappa.^2)))));
  cS = exp(-(((deltaS.^2)/(2*(kappa.^2)))));
  cE = exp(-(((deltaE.^2)/(2*(kappa.^2)))));
  cW = exp(-(((deltaW.^2)/(2*(kappa.^2)))));

elseif option == 3 % degrau

  cN = deltaN./deltaN;
  cS = deltaS./deltaS;
  cE = deltaE./deltaE;
  cW = deltaW./deltaW;

  cN((deltaN.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cS((deltaS.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cE((deltaE.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
  cW((deltaW.^2./(sqrt(2)*kappa).^2)>=0.5)=0;

  cN((deltaN.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cS((deltaS.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cE((deltaE.^2./(sqrt(2)*kappa).^2)<0.5)=1;
  cW((deltaW.^2./(sqrt(2)*kappa).^2)<0.5)=1;

elseif option == 4  % tukey

  cN = (1-((deltaN.^2)/(5*(kappa.^2)))).^2;
  cS = (1-((deltaS.^2)/(5*(kappa.^2)))).^2;
  cE = (1-((deltaE.^2)/(5*(kappa.^2)))).^2;
  cW = (1-((deltaW.^2)/(5*(kappa.^2)))).^2;

  cN(abs((deltaN.^2)/5)>((kappa.^2)))=0;
  cS(abs((deltaS.^2)/5)>((kappa.^2)))=0;
  cE(abs((deltaE.^2)/5)>((kappa.^2)))=0;
  cW(abs((deltaW.^2)/5)>((kappa.^2)))=0;

elseif option == 5 % huber

  cN=abs(kappa./(deltaN.*3.7));
  cS=abs(kappa./(deltaS.*3.7));
  cE=abs(kappa./(deltaE.*3.7));
  cW=abs(kappa./(deltaW.*3.7));

  cN(abs(deltaN.*3.7)<=kappa)=1;
```

```
    cS(abs(deltaS.*3.7)<=kappa)=1;
    cE(abs(deltaE.*3.7)<=kappa)=1;
    cW(abs(deltaW.*3.7)<=kappa)=1;

  elseif option == 6 % L1 norm

    cN = deltaN./deltaN;
    cS = deltaS./deltaS;
    cE = deltaE./deltaE;
    cW = deltaW./deltaW;

    cN((deltaN)<0)=-1;
    cS((deltaS)<0)=-1;
    cE((deltaE)<0)=-1;
    cW((deltaW)<0)=-1;

    cN((deltaN)>0)=1;
    cS((deltaS)>0)=1;
    cE((deltaE)>0)=1;
    cW((deltaW)>0)=1;



  elseif option == 7  % geman and mcclure

    cN = 2./((1+(.014.*deltaN.^2)).^2);
    cS = 2./((1+(.014.*deltaS.^2)).^2);
    cE = 2./((1+(.014.*deltaE.^2)).^2);
    cW = 2./((1+(.014.*deltaW.^2)).^2);

  elseif option == 8   % forwardbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=2;
    alfa=kf./(2.*(kb+w));

    cN =(1./(1+((deltaN./kf).^n))) - (alfa./(1+(((deltaN-kb)./w).^(2*m))));
    cS =(1./(1+((deltaS./kf).^n))) - (alfa./(1+(((deltaS-kb)./w).^(2*m))));
    cE =(1./(1+((deltaE./kf).^n))) - (alfa./(1+(((deltaE-kb)./w).^(2*m))));
    cW =(1./(1+((deltaW./kf).^n))) - (alfa./(1+(((deltaW-kb)./w).^(2*m))));

  elseif option == 9  % side

    cN = (exp(-((((abs(deltaN)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaN)))));
    cS = (exp(-((((abs(deltaS)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaS)))));
    cE = (exp(-((((abs(deltaE)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaE)))));
    cW = (exp(-((((abs(deltaW)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaW)))));

    cN((deltaN)==0)=0;
```

```
    cS((deltaS)==0)=0;
    cE((deltaE)==0)=0;
    cW((deltaW)==0)=0;


   elseif option == 10  % c2forwardandbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=1;
    alfa=kf./(2.*kb);

    cN = zeros(size(deltaN));
    cS = zeros(size(deltaS));
    cE = zeros(size(deltaE));
    cW = zeros(size(deltaW));


    [i,j]=size(deltaN);
    [i,j]=size(deltaS);
    [i,j]=size(deltaE);
    [i,j]=size(deltaW);


for i = 1:i, ;
    for j =1:j, ;
         if deltaN(i,j) <= kf;
             if deltaN(i,j)  >= 0;
             cN(i,j) = 1-((deltaN(i,j)./kf).^n);
              end
               elseif deltaN(i,j) <= (kb+w);
                   if deltaN(i,j) >= (kb-w);
                    cN(i,j)=alfa.*((((deltaN(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
 end

for i = 1:i, ;
    for j =1:j, ;
         if deltaS(i,j) <= kf;
             if deltaS(i,j)  >= 0;
             cS(i,j) = 1-((deltaS(i,j)./kf).^n);
              end
               elseif deltaS(i,j) <= (kb+w);
                   if deltaS(i,j) >= (kb-w);
                    cS(i,j)=alfa.*((((deltaS(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
 end
```

```
for i = 1:i, ;
    for j =1:j, ;
         if deltaE(i,j) <= kf;
             if deltaE(i,j)  >= 0;
             cE(i,j) = 1-((deltaE(i,j)./kf).^n);
              end
                elseif deltaE(i,j) <= (kb+w);
                    if deltaE(i,j) >= (kb-w);
                     cE(i,j)=alfa.*((((deltaE(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end


for i = 1:i, ;
    for j =1:j, ;
         if deltaW(i,j) <= kf;
             if deltaW(i,j)  >= 0;
             cW(i,j) = 1-((deltaW(i,j)./kf).^n);
              end
                elseif deltaW(i,j) <= (kb+w);
                    if deltaW(i,j) >= (kb-w);
                     cW(i,j)=alfa.*((((deltaW(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end


72

 end







%diff = diff + lambda*(cE.*deltaE + cW.*deltaW + cN.*deltaN + cS.*deltaS);
diff = diff + lambda*(cE.*deltaE + cW.*deltaW );
%deltaN
%deltaS

s=i;% backup i
diff1=diff;
h=length(diff1);
for i = 2:h-1;

%    if  abs((diff1(i)-diff1(i-1))) < (kappa);

%         if  abs((diff1(i)-diff1(i+1))) < (kappa);
```

```
              diff(i)=median([diff1(i-1) diff1(i)  diff1(i+1)]);
%          end
%    end
end
gout=diff;
ee=(fft(gout))/length(gout);
dd=ee(:,(pontos/39):(pontos/2));

ddd=ee(:,2:(pontos/39));
vetor(1,s)=(sum(abs(ddd)) - sum(abs(dd)));
erro(1,s)=sqrt((sum((im-io).^2))/(sum((diff-io).^2)));
NMSE(1,s)=(sum((diff-io).^2))/ (sum(io.^2));
end
fprintf('\n');
```

```
function diff = filtrosensorx(im, niterations, kappa, lambda, option,p,i,d)

Kp=p;
Ki=i;
Kd=d;



im = double(im);
[rows,cols] = size(im);
diff = im;

integral_erroN= [diff*0];
integral_erroS= [diff*0];
integral_erroE= [diff*0];
integral_erroW= [diff*0];
erroN_anterior= [diff*0];
erroS_anterior= [diff*0];
erroE_anterior= [diff*0];
erroW_anterior= [diff*0];
std_diff=std(diff);
std_diff_anterior=std(diff);

final=28.92;
inicial=42.66;

for i = 1:niterations;



  diffl = zeros(rows+2, cols+2);
  diffl(2:rows+1, 2:cols+1) = diff;


  deltaNa = diffl(1:rows,2:cols+1) - diff;
  deltaSa = diffl(3:rows+2,2:cols+1) - diff;
  deltaEa = diffl(2:rows+1,3:cols+2) - diff;
  deltaWa = diffl(2:rows+1,1:cols) - diff;

  derivada_erroNa = deltaNa - erroN_anterior;
  derivada_erroSa = deltaSa - erroS_anterior;
  derivada_erroEa = deltaEa - erroE_anterior;
  derivada_erroWa = deltaWa - erroW_anterior;

  erroN_anterior = deltaNa; %GUARDA ERRO ANTERIOR
  erroS_anterior = deltaSa;
  erroE_anterior = deltaEa;
  erroW_anterior = deltaWa;

  integral_erroN = deltaNa + integral_erroN;
  integral_erroS = deltaSa + integral_erroS;
  integral_erroE = deltaEa + integral_erroE;
```

```
  integral_erroW = deltaWa + integral_erroW;

  deltaN = ((Kp*deltaNa)+(Ki*integral_erroN)+(Kd*derivada_erroNa));
  deltaS = ((Kp*deltaSa)+(Ki*integral_erroS)+(Kd*derivada_erroSa));
  deltaE = ((Kp*deltaEa)+(Ki*integral_erroE)+(Kd*derivada_erroEa));
  deltaW = ((Kp*deltaWa)+(Ki*integral_erroW)+(Kd*derivada_erroWa));

  if option == 1 %perona 1

    cN = 1./(1+(((deltaN.^2))/((kappa.^2))));
    cS = 1./(1+(((deltaS.^2))/((kappa.^2))));
    cE = 1./(1+(((deltaE.^2))/((kappa.^2))));
    cW = 1./(1+(((deltaW.^2))/((kappa.^2))));

  elseif option == 2 % perona 2

    cN = exp(-(((deltaN.^2)/(2*(kappa.^2)))));
    cS = exp(-(((deltaS.^2)/(2*(kappa.^2)))));
    cE = exp(-(((deltaE.^2)/(2*(kappa.^2)))));
    cW = exp(-(((deltaW.^2)/(2*(kappa.^2)))));

  elseif option == 3 % degrau

    cN = deltaN./deltaN;
    cS = deltaS./deltaS;
    cE = deltaE./deltaE;
    cW = deltaW./deltaW;

    cN((deltaN.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
    cS((deltaS.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
    cE((deltaE.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
    cW((deltaW.^2./(sqrt(2)*kappa).^2)>=0.5)=0;

    cN((deltaN.^2./(sqrt(2)*kappa).^2)<0.5)=1;
    cS((deltaS.^2./(sqrt(2)*kappa).^2)<0.5)=1;
    cE((deltaE.^2./(sqrt(2)*kappa).^2)<0.5)=1;
    cW((deltaW.^2./(sqrt(2)*kappa).^2)<0.5)=1;

  elseif option == 4  % tukey

    cN = (1-((deltaN.^2)/(5*(kappa.^2)))).^2;
    cS = (1-((deltaS.^2)/(5*(kappa.^2)))).^2;
    cE = (1-((deltaE.^2)/(5*(kappa.^2)))).^2;
    cW = (1-((deltaW.^2)/(5*(kappa.^2)))).^2;

    cN(abs((deltaN.^2)/5)>((kappa.^2)))=0;
    cS(abs((deltaS.^2)/5)>((kappa.^2)))=0;
    cE(abs((deltaE.^2)/5)>((kappa.^2)))=0;
    cW(abs((deltaW.^2)/5)>((kappa.^2)))=0;

  elseif option == 5 % huber

    cN=abs(kappa./(deltaN.*3.7));
    cS=abs(kappa./(deltaS.*3.7));
    cE=abs(kappa./(deltaE.*3.7));
    cW=abs(kappa./(deltaW.*3.7));
```

```
    cN(abs(deltaN.*3.7)<=kappa)=1;
    cS(abs(deltaS.*3.7)<=kappa)=1;
    cE(abs(deltaE.*3.7)<=kappa)=1;
    cW(abs(deltaW.*3.7)<=kappa)=1;

  elseif option == 6 % L1 norm

    cN = deltaN./deltaN;
    cS = deltaS./deltaS;
    cE = deltaE./deltaE;
    cW = deltaW./deltaW;

    cN((deltaN)<0)=-1;
    cS((deltaS)<0)=-1;
    cE((deltaE)<0)=-1;
    cW((deltaW)<0)=-1;

    cN((deltaN)>0)=1;
    cS((deltaS)>0)=1;
    cE((deltaE)>0)=1;
    cW((deltaW)>0)=1;



  elseif option == 7  % geman and mcclure

    cN = 2./((1+(.014.*deltaN.^2)).^2);
    cS = 2./((1+(.014.*deltaS.^2)).^2);
    cE = 2./((1+(.014.*deltaE.^2)).^2);
    cW = 2./((1+(.014.*deltaW.^2)).^2);

  elseif option == 8   % forwardbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=2;
    alfa=kf./(2.*(kb+w));

    cN =(1./(1+((deltaN./kf).^n))) - (alfa./(1+(((deltaN-kb)./w).^(2*m))));
    cS =(1./(1+((deltaS./kf).^n))) - (alfa./(1+(((deltaS-kb)./w).^(2*m))));
    cE =(1./(1+((deltaE./kf).^n))) - (alfa./(1+(((deltaE-kb)./w).^(2*m))));
    cW =(1./(1+((deltaW./kf).^n))) - (alfa./(1+(((deltaW-kb)./w).^(2*m))));

  elseif option == 9  % side

    cN = (exp(-((((abs(deltaN)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaN)))));
    cS = (exp(-((((abs(deltaS)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaS)))));
    cE = (exp(-((((abs(deltaE)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaE)))));
    cW = (exp(-((((abs(deltaW)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaW)))));
```

```
    cN((deltaN)==0)=0;
    cS((deltaS)==0)=0;
    cE((deltaE)==0)=0;
    cW((deltaW)==0)=0;


   elseif option == 10  % c2forwardandbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=1;
    alfa=kf./(2.*kb);

    cN = zeros(size(deltaN));
    cS = zeros(size(deltaS));
    cE = zeros(size(deltaE));
    cW = zeros(size(deltaW));


    [i,j]=size(deltaN);
    [i,j]=size(deltaS);
    [i,j]=size(deltaE);
    [i,j]=size(deltaW);


for i = 1:i, ;
    for j =1:j, ;
         if deltaN(i,j) <= kf;
             if deltaN(i,j)  >= 0;
             cN(i,j) = 1-((deltaN(i,j)./kf).^n);
              end
               elseif deltaN(i,j) <= (kb+w);
                   if deltaN(i,j) >= (kb-w);
                    cN(i,j)=alfa.*((((deltaN(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
 end

for i = 1:i, ;
    for j =1:j, ;
         if deltaS(i,j) <= kf;
             if deltaS(i,j)  >= 0;
             cS(i,j) = 1-((deltaS(i,j)./kf).^n);
              end
               elseif deltaS(i,j) <= (kb+w);
                   if deltaS(i,j) >= (kb-w);
                    cS(i,j)=alfa.*((((deltaS(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
```

```
 end


for i = 1:i, ;
    for j =1:j, ;
         if deltaE(i,j) <= kf;
             if deltaE(i,j)  >= 0;
             cE(i,j) = 1-((deltaE(i,j)./kf).^n);
              end
                elseif deltaE(i,j) <= (kb+w);
                    if deltaE(i,j) >= (kb-w);
                     cE(i,j)=alfa.*((((deltaE(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end


for i = 1:i, ;
    for j =1:j, ;
         if deltaW(i,j) <= kf;
             if deltaW(i,j)  >= 0;
             cW(i,j) = 1-((deltaW(i,j)./kf).^n);
              end
                elseif deltaW(i,j) <= (kb+w);
                    if deltaW(i,j) >= (kb-w);
                     cW(i,j)=alfa.*((((deltaW(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end




 end




%diff = diff + lambda*(cE.*deltaE + cW.*deltaW + cN.*deltaN + cS.*deltaS);
diff = diff + lambda*(cE.*deltaE + cW.*deltaW );
%deltaN
%deltaS

diff(length(diff))=final;
diff(1)=inicial;

diff1=diff;
h=length(diff1);
for i = 2:h-1;
```

```
            diff(i)=median([diff1(i-1) diff1(i)  diff1(i+1)]);

 end
end
fprintf('\n');
```

```
function diff = filtrosensorxn(im, niterations, kappa, lambda, option,p,i,d)

Kp=p;
Ki=i;
Kd=d;




im = double(im);
[rows,cols] = size(im);
diff = im;

integral_erroN= [diff*0];
integral_erroS= [diff*0];
integral_erroE= [diff*0];
integral_erroW= [diff*0];
erroN_anterior= [diff*0];
erroS_anterior= [diff*0];
erroE_anterior= [diff*0];
erroW_anterior= [diff*0];
std_diff=std(diff);
std_diff_anterior=std(diff);

final=26.5668;
inicial=42.66;

for i = 1:niterations;




  diffl = zeros(rows+2, cols+2);
  diffl(2:rows+1, 2:cols+1) = diff;


  deltaNa = diffl(1:rows,2:cols+1) - diff;
  deltaSa = diffl(3:rows+2,2:cols+1) - diff;
  deltaEa = diffl(2:rows+1,3:cols+2) - diff;
  deltaWa = diffl(2:rows+1,1:cols) - diff;

  derivada_erroNa = deltaNa - erroN_anterior;
  derivada_erroSa = deltaSa - erroS_anterior;
  derivada_erroEa = deltaEa - erroE_anterior;
  derivada_erroWa = deltaWa - erroW_anterior;

  erroN_anterior = deltaNa; %GUARDA ERRO ANTERIOR
  erroS_anterior = deltaSa;
  erroE_anterior = deltaEa;
  erroW_anterior = deltaWa;

  integral_erroN = deltaNa + integral_erroN;
  integral_erroS = deltaSa + integral_erroS;
  integral_erroE = deltaEa + integral_erroE;
```

```
  integral_erroW = deltaWa + integral_erroW;

  deltaN = ((Kp*deltaNa)+(Ki*integral_erroN)+(Kd*derivada_erroNa));
  deltaS = ((Kp*deltaSa)+(Ki*integral_erroS)+(Kd*derivada_erroSa));
  deltaE = ((Kp*deltaEa)+(Ki*integral_erroE)+(Kd*derivada_erroEa));
  deltaW = ((Kp*deltaWa)+(Ki*integral_erroW)+(Kd*derivada_erroWa));

  if option == 1 %perona 1

    cN = 1./(1+(((deltaN.^2))/((kappa.^2))));
    cS = 1./(1+(((deltaS.^2))/((kappa.^2))));
    cE = 1./(1+(((deltaE.^2))/((kappa.^2))));
    cW = 1./(1+(((deltaW.^2))/((kappa.^2))));

  elseif option == 2 % perona 2

    cN = exp(-(((deltaN.^2)/(2*(kappa.^2)))));
    cS = exp(-(((deltaS.^2)/(2*(kappa.^2)))));
    cE = exp(-(((deltaE.^2)/(2*(kappa.^2)))));
    cW = exp(-(((deltaW.^2)/(2*(kappa.^2)))));

  elseif option == 3 % degrau

    cN = deltaN./deltaN;
    cS = deltaS./deltaS;
    cE = deltaE./deltaE;
    cW = deltaW./deltaW;

    cN((deltaN.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
    cS((deltaS.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
    cE((deltaE.^2./(sqrt(2)*kappa).^2)>=0.5)=0;
    cW((deltaW.^2./(sqrt(2)*kappa).^2)>=0.5)=0;

    cN((deltaN.^2./(sqrt(2)*kappa).^2)<0.5)=1;
    cS((deltaS.^2./(sqrt(2)*kappa).^2)<0.5)=1;
    cE((deltaE.^2./(sqrt(2)*kappa).^2)<0.5)=1;
    cW((deltaW.^2./(sqrt(2)*kappa).^2)<0.5)=1;

  elseif option == 4  % tukey

    cN = (1-((deltaN.^2)/(5*(kappa.^2)))).^2;
    cS = (1-((deltaS.^2)/(5*(kappa.^2)))).^2;
    cE = (1-((deltaE.^2)/(5*(kappa.^2)))).^2;
    cW = (1-((deltaW.^2)/(5*(kappa.^2)))).^2;

    cN(abs((deltaN.^2)/5)>((kappa.^2)))=0;
    cS(abs((deltaS.^2)/5)>((kappa.^2)))=0;
    cE(abs((deltaE.^2)/5)>((kappa.^2)))=0;
    cW(abs((deltaW.^2)/5)>((kappa.^2)))=0;

  elseif option == 5 % huber

    cN=abs(kappa./(deltaN.*3.7));
    cS=abs(kappa./(deltaS.*3.7));
    cE=abs(kappa./(deltaE.*3.7));
    cW=abs(kappa./(deltaW.*3.7));
```

```
    cN(abs(deltaN.*3.7)<=kappa)=1;
    cS(abs(deltaS.*3.7)<=kappa)=1;
    cE(abs(deltaE.*3.7)<=kappa)=1;
    cW(abs(deltaW.*3.7)<=kappa)=1;

  elseif option == 6 % L1 norm

    cN = deltaN./deltaN;
    cS = deltaS./deltaS;
    cE = deltaE./deltaE;
    cW = deltaW./deltaW;

    cN((deltaN)<0)=-1;
    cS((deltaS)<0)=-1;
    cE((deltaE)<0)=-1;
    cW((deltaW)<0)=-1;

    cN((deltaN)>0)=1;
    cS((deltaS)>0)=1;
    cE((deltaE)>0)=1;
    cW((deltaW)>0)=1;



  elseif option == 7  % geman and mcclure

    cN = 2./((1+(.014.*deltaN.^2)).^2);
    cS = 2./((1+(.014.*deltaS.^2)).^2);
    cE = 2./((1+(.014.*deltaE.^2)).^2);
    cW = 2./((1+(.014.*deltaW.^2)).^2);

  elseif option == 8   % forwardbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=2;
    alfa=kf./(2.*(kb+w));

    cN =(1./(1+((deltaN./kf).^n))) - (alfa./(1+(((deltaN-kb)./w).^(2*m))));
    cS =(1./(1+((deltaS./kf).^n))) - (alfa./(1+(((deltaS-kb)./w).^(2*m))));
    cE =(1./(1+((deltaE./kf).^n))) - (alfa./(1+(((deltaE-kb)./w).^(2*m))));
    cW =(1./(1+((deltaW./kf).^n))) - (alfa./(1+(((deltaW-kb)./w).^(2*m))));

  elseif option == 9  % side

    cN = (exp(-((((abs(deltaN)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaN)))));
    cS = (exp(-((((abs(deltaS)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaS)))));
    cE = (exp(-((((abs(deltaE)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaE)))));
    cW = (exp(-((((abs(deltaW)+(kappa./2)).*2.236)/(sqrt(2)*kappa))).^2)).*((1+(kappa./↙
(2.*abs(deltaW)))));
```

```
    cN((deltaN)==0)=0;
    cS((deltaS)==0)=0;
    cE((deltaE)==0)=0;
    cW((deltaW)==0)=0;


   elseif option == 10  % c2forwardandbackward

    kf=kappa./2;
    kb=4;
    w=.1;%para sinal 2d (imagem)w=1
    n=4;
    m=1;
    alfa=kf./(2.*kb);

    cN = zeros(size(deltaN));
    cS = zeros(size(deltaS));
    cE = zeros(size(deltaE));
    cW = zeros(size(deltaW));


    [i,j]=size(deltaN);
    [i,j]=size(deltaS);
    [i,j]=size(deltaE);
    [i,j]=size(deltaW);


for i = 1:i, ;
    for j =1:j, ;
         if deltaN(i,j) <= kf;
             if deltaN(i,j)  >= 0;
             cN(i,j) = 1-((deltaN(i,j)./kf).^n);
              end
               elseif deltaN(i,j) <= (kb+w);
                   if deltaN(i,j) >= (kb-w);
                    cN(i,j)=alfa.*((((deltaN(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
 end

for i = 1:i, ;
    for j =1:j, ;
         if deltaS(i,j) <= kf;
             if deltaS(i,j)  >= 0;
             cS(i,j) = 1-((deltaS(i,j)./kf).^n);
              end
               elseif deltaS(i,j) <= (kb+w);
                   if deltaS(i,j) >= (kb-w);
                    cS(i,j)=alfa.*((((deltaS(i,j)-kb)./w).^(2*m))-1);
                    end

          end
     end
```

```
 end


for i = 1:i, ;
    for j =1:j, ;
         if deltaE(i,j) <= kf;
             if deltaE(i,j)  >= 0;
             cE(i,j) = 1-((deltaE(i,j)./kf).^n);
              end
                elseif deltaE(i,j) <= (kb+w);
                    if deltaE(i,j) >= (kb-w);
                     cE(i,j)=alfa.*((((deltaE(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end


for i = 1:i, ;
    for j =1:j, ;
         if deltaW(i,j) <= kf;
             if deltaW(i,j)  >= 0;
             cW(i,j) = 1-((deltaW(i,j)./kf).^n);
              end
                elseif deltaW(i,j) <= (kb+w);
                    if deltaW(i,j) >= (kb-w);
                     cW(i,j)=alfa.*((((deltaW(i,j)-kb)./w).^(2*m))-1);
                     end

          end
     end
 end




 end




%diff = diff + lambda*(cE.*deltaE + cW.*deltaW + cN.*deltaN + cS.*deltaS);
diff = diff + lambda*(cE.*deltaE + cW.*deltaW );
%deltaN
%deltaS

diff(length(diff))=final;
diff(1)=inicial;

diff1=diff;
h=length(diff1);
for i = 2:h-1;
```

```
            diff(i)= diff1(i);

 end
end
fprintf('\n');
```

```
function y = mad(x)
% CALCULA EQ. 6.1;
[nrow,ncol] = size(x);
med = median(x);
y = abs(x - med(ones(nrow,1),:));
y = median(y);
```

```
function [QRSwave]=ECGwaveGen(bpm,duration,fs,amp)
%[QRSwave]=ECGwaveGen(bpm,dur,fs,amp) generates an artificial ECG/EKG waveform
%    Heart rate (bpm) sets the qrs event frequency (RR interval).
%    Duration of the entire waveform (dur) is in units of seconds.
%    Sample frequency (fs) sets the sample frequency in Hertz.
%    Amplitude (amp) of the QRS event is measured in micro Volts. The
%    waveform consists of a QRS complex and a T-wave. No attempt to
%    represent a P-wave has been made.
%
%    There are two additional parameters that can be changed from within the function.
%    They are the parameters that set the QRS width (default 0.1 secs) and the t-wave
%    amplitude (default 500 uV).

%Created January 22, 2001 by Floyd Harriott, primary email (fharriott@stellate.com),↙
secondary email (fsh@po.cwru.edu)
%Modified March 19, 2002 by Floyd Harriott, extended default duration so that default↙
settings produce a QRS event rather than
%   an error. Allows for the random insertion of PVCs.  This file must be edited to↙
include PVCs.

%Algorithm is based in part on the jounal article:
%Ruha, Antti and Seppo Nissila, "A Real-Time Microprocessor QRS Detector System with a↙
1-ms Timing Accuracy
%      for the Measurement of Ambulatory HRV", IEEE Trans. Biomed. Eng. Vol. 44, No. 3,↙
1997
%The artificial ECG signal they describe is based on the recommendations in the↙
Association for the Advancement
%of Medical Instrumentation (AAMI) "Standard for Cardiac Monitors, Heart Rate Meters↙
and Alarms (draft), Aug. 1981
%Feel free to make modifications, corrections and or suggestions.

if (exist('fs') ~= 1)  fs=  200;   end  %default value, Hz
if (exist('bpm') ~= 1)  bpm =  72;   end %default value, beats per minute
if (exist('amp') ~= 1)  amp = 1000;   end %default value, micro volts
if (exist('duration') ~= 1)  duration =  (60/bpm-0.35)+60/bpm+1/fs; end  %default value↙
gives one cycle, seconds

global t_line; %seconds
global sample_freq; % always equal to fs


%Changeable Parameters
d=0.1; %.07 to .120 seconds, QRS width
at=500; %amplitude of t-wave, 400 to 1200 uv

%Should not touch
org_amp=amp;
sample_freq=fs; %duplicated simply to make a global version
RR=(60/bpm); %RR interval, seconds
d1=0.4375*d;
d2=0.5*d;
d3=d-(d1+d2);
dt=0.180; %width of t wave, seconds
qt=0.35; %time from beginning of QRS to end of t-wave
t_line=0:1/fs:duration; %time line, seconds
QRS_wave=zeros( size(t_line) ); %QRS waveform
```

```
deadspace=RR-qt; %time between t-wave and next QRS
if deadspace < 0
    err_msg=['Bpm must be equal to or less than ' int2str(60/qt) ' inorder to fit one↙
cycle.'];
    error(err_msg);
end


%Calculate PVC parameters and segment
PVCchance=0.1; %How often does PVC happen., percent eg. 0.1=10%
PVCamp=amp;    %PVC amplitude, eg. same as normals (amp)
earlyfactor=0.25; %percentage, how much early should PVC happen then normal RR interval
PVCwidth=0.12; %seconds, QRS width of PVC, usually .12 to .17
PVCseg=[QRSpulse(d,60/((1-earlyfactor)*RR-0.4375*PVCwidth),fs,RandAmp(org_amp))↙
QRSpulse(PVCwidth,bpm*(1-earlyfactor),fs,PVCamp) QRSpulse(d,bpm,fs, RandAmp(org_amp))];↙
%PVC segment
tPVC=size(PVCseg,2)/fs; %amount of time taken up by PVC segment in seconds

t1=deadspace; %Where does the first QRS start? eg deadspace, or 0

%need enough time to display at least one interval.
if (t1+60/bpm+1/sample_freq > duration)
    err_msg=['The waveform length (duration) must be more than ' sprintf('%.2f%',↙
t1+60/bpm+1/sample_freq) ' second(s) in order to display one QRS event.'];
    error(err_msg);
end


%GENERATION LOOP
while ( t1+60/bpm+1/sample_freq <= duration) %space to insert another qrs pulse in time↙
line

    %amp=RandAmp(org_amp); %random size on qrs event
    amp=org_amp;



    %Segment 1  (Q-R)
    qrs_start=t1;
    t2=t1+d1;
    i_t1=time2index(t1); i_t2=time2index(t2);
    left=0; right=0.875*amp;
    m1=(right-left)/(t2-t1);
    QRS1=m1*index2time(i_t1:i_t2)-(m1*t1-left);
    QRSwave(i_t1:i_t2)=QRS1;

    %Segment 2  (R-?)
    t1=t2; t2=t1+d2;
    i_t1=time2index(t1); i_t2=time2index(t2);
    left=right; right=-.125*amp;
    m2=(right-left)/(t2-t1);
    QRS1=m2*index2time(i_t1:i_t2)-(m2*t1-left);
    QRSwave(i_t1:i_t2)=QRS1;


    %Segment 3 bottom_top (?-S)
```

```
    t1=t2; t2=t1+d3;
    i_t1=time2index(t1); i_t2=time2index(t2);
    left=right; right=0;
    if (i_t2-i_t1 >0) %at low sampling freq. there may be no sample for this segment
        m3=(right-left)/(t2-t1);
        QRS1=m3*index2time(i_t1:i_t2)-(m3*t1-left);
        QRS1=QRS1( find(QRS1<=0));
        QRSwave(i_t1:i_t1+size(QRS1,2)-1)=QRS1;
    elseif i_t2-i_t1==0
        m3=(right-left)/(t2-t1);
        QRS1=m3*index2time(i_t1:i_t2)-(m3*t1-left);
        QRSwave(i_t1)=QRS1(1);
    end

    %Segment 4, S-T interval
    t1=t2; t2=t1+qt+qrs_start-(dt+t2);
    i_t1=time2index(t1); i_t2=time2index(t2);
    left=right; right=0;

    %Segment 5, t-wave
    t1=t2; t2=t1+dt;
    i_t1=time2index(t1); i_t2=time2index(t2);
    t=-1:2/(i_t2-i_t1):1;
    QRS1=at*sqrt(1-t.^2);
    QRSwave(i_t1:i_t2)=QRS1;

    %Segment 6, remaining deadspace
    t1=t2; t2=t1+deadspace;
    i_t1=time2index(t1); i_t2=time2index(t2);


    %Do we insert a PVC here? Roll the die and find out.
    insertPVC=rand(1); %uncomment following 5 lines if PVCs are desired.
    %if insertPVC<=PVCchance & t2+tPVC+2/sample_freq <= duration %enough space to↙
insert PVC
     %   t1=t2; t2=t1+tPVC;
      %  i_t1=time2index(t1); i_t2=time2index(t2);
       % QRSwave(i_t1:i_t1+size(PVCseg,2)-1)=PVCseg;
    %end

    %stem(QRSwave); % view ECG waveform
    t1=t2; %end of this segment becomes beginning of next segment

end %while loop, appending qrs pulses

%_____________________________________%
function index=time2index(t)
%TIME2INDEX converts time (s) to an index value

global t_line;

indexArray=find(t_line>=t);
index=indexArray(1);

%_____________________________________%
function time=index2time(i)
```

```
%INDEX2TIME converts a time line index to a time value (seconds)
global sample_freq

time=(i-1).*1/sample_freq;


%_____________________________________________%
function RAmp=RandAmp(orgAmp)

RAmp=orgAmp+0.4*orgAmp*rand(1);
```

```
%#################################################################3333
%  function randalph = randalph([Nx,Ny],alpha,seed)
%
%  Generates a matrix with unit dispersion, zero-centered
%  i.i.d. alpha-stable random variables.
%
%  INPUTS
%       N=[Nx,Ny]: Matrix size
%       alpha: characteristic exponent
%       seed: specific seed value for generation
%
%  OUTPUT
%       randalph: Nx times Ny output matrix
%

% PLEASE REPORT BUGS AND COMMENTS TO:
%       Juan G. Gonzalez
%          Department of Electrical Engineering
%          University of Delaware
%          Newark, DE 19716, USA
%
%          e-mail: gonzalez@ee.udel.edu
%          Fax: (302) 831-4316
%          WWW: http://www.ee.udel.edu/~gonzalez/


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               Copyright (c) 1996                                    %
%                Juan G. Gonzalez                                     %
%                                                                     %
% This software can be used, copied, distributed or                   %
% altered provided that this notice is clearly                        %
% maintained.                                                         %
% The right to use or distribute this file for                        %
% profit or as part of any commercial product is                      %
% specifically reserved for the copyright owner.                      %
%                                                                     %
% Developed:    September 16, 1996                                    %
% Updated 1:                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



  function randalph = randalph(N,alpha)


%  if nargin > 2
%     randn('seed',seed);
%     rand('seed',seed);
 randn('state',0)
 rand('state',0)
%  end

  if alpha > 2
        error('Characteristic exponent must be less than or equal to 2 !!')
```

```
  elseif alpha == 2
        randalph = sqrt(2)*randn(N);
  elseif alpha == 1
        randalph = tan(pi*(rand(N)-.5*ones(N)));
  elseif alpha > 0
        u = pi*(rand(N)-.5*ones(N));
        alphac = 1-alpha;
        cosalph = cos(alphac*u);

         randalph =(cosalph./cos(u)).^(1/alpha).*sin(alpha*u)./cosalph.*(-log(rand↙
(N))).^(-alphac/alpha);
  else
        error('Characteristic exponent must be positive!!')
  end
%##################################################################
```

1.0e+003 *

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

```
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
  0.1000
  0.2000
  0.3000
  0.4000
  0.5000
  0.6000
  0.7000
  0.8000
  0.9000
  1.0000
  1.1000
  1.2000
  1.3000
  1.4000
  1.5000
  1.5500
  1.4500
  1.3500
  1.2500
  1.1500
  1.0500
  0.9500
  0.8500
  0.7500
  0.6500
  0.5500
  0.4500
  0.3500
  0.2500
  0.1500
  0.0500
 -0.0500
 -0.1500
 -0.2000
```

```
-0.1000
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
 0.1240
 0.1740
 0.2114
 0.2421
 0.2684
 0.2915
 0.3121
 0.3307
 0.3476
 0.3631
 0.3773
 0.3903
 0.4023
 0.4134
 0.4236
 0.4330
 0.4417
 0.4496
 0.4569
 0.4635
 0.4695
 0.4750
 0.4798
 0.4841
 0.4879
 0.4911
 0.4939
 0.4961
 0.4978
 0.4990
 0.4998
 0.5000
 0.4998
 0.4990
 0.4978
 0.4961
 0.4939
 0.4911
 0.4879
 0.4841
```

0.4798
0.4750
0.4695
0.4635
0.4569
0.4496
0.4417
0.4330
0.4236
0.4134
0.4023
0.3903
0.3773
0.3631
0.3476
0.3307
0.3121
0.2915
0.2684
0.2421
0.2114
0.1740
0.1240
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

QRSwave
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0.1000
0.2000
0.3000
0.4000
0.5000
0.6000
0.7000
0.8000
0.9000
1.0000
1.1000
1.2000
1.3000
1.4000

```
 1.5000
 1.5500
 1.4500
 1.3500
 1.2500
 1.1500
 1.0500
 0.9500
 0.8500
 0.7500
 0.6500
 0.5500
 0.4500
 0.3500
 0.2500
 0.1500
 0.0500
-0.0500
-0.1500
-0.2000
-0.1000
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
 0.1240
 0.1740
 0.2114
 0.2421
 0.2684
 0.2915
 0.3121
 0.3307
 0.3476
 0.3631
 0.3773
 0.3903
 0.4023
 0.4134
 0.4236
 0.4330
 0.4417
 0.4496
 0.4569
 0.4635
```

```
0.4695
0.4750
0.4798
0.4841
0.4879
0.4911
0.4939
0.4961
0.4978
0.4990
0.4998
0.5000
0.4998
0.4990
0.4978
0.4961
0.4939
0.4911
0.4879
0.4841
0.4798
0.4750
0.4695
0.4635
0.4569
0.4496
0.4417
0.4330
0.4236
0.4134
0.4023
0.3903
0.3773
0.3631
0.3476
0.3307
0.3121
0.2915
0.2684
0.2421
0.2114
0.1740
0.1240
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
```

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

```
       0
       0
       0
       0
       0
       0
  0.1000
  0.2000
  0.3000
  0.4000
  0.5000
  0.6000
  0.7000
  0.8000
  0.9000
  1.0000
  1.1000
  1.2000
  1.3000
  1.4000
  1.5000
  1.5500
  1.4500
  1.3500
  1.2500
  1.1500
  1.0500
  0.9500
  0.8500
  0.7500
  0.6500
  0.5500
  0.4500
  0.3500
  0.2500
  0.1500
  0.0500
 -0.0500
 -0.1500
 -0.2000
 -0.1000
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
```

```
0.1240
0.1740
0.2114
0.2421
0.2684
0.2915
0.3121
0.3307
0.3476
0.3631
0.3773
0.3903
0.4023
0.4134
0.4236
0.4330
0.4417
0.4496
0.4569
0.4635
0.4695
0.4750
0.4798
0.4841
0.4879
0.4911
0.4939
0.4961
0.4978
0.4990
0.4998
0.5000
0.4998
0.4990
0.4978
0.4961
0.4939
0.4911
0.4879
0.4841
0.4798
0.4750
0.4695
0.4635
0.4569
0.4496
0.4417
0.4330
0.4236
0.4134
0.4023
0.3903
0.3773
0.3631
0.3476
0.3307
0.3121
0.2915
0.2684
0.2421
0.2114
0.1740
0.1240
     0
     0
     0
     0
     0
```

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

```
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
       0
  0.1000
  0.2000
  0.3000
  0.4000
  0.5000
  0.6000
  0.7000
  0.8000
  0.9000
  1.0000
  1.1000
  1.2000
  1.3000
  1.4000
  1.5000
  1.5500
  1.4500
  1.3500
  1.2500
  1.1500
  1.0500
  0.9500
  0.8500
  0.7500
  0.6500
  0.5500
  0.4500
  0.3500
  0.2500
  0.1500
  0.0500
 -0.0500
 -0.1500
 -0.2000
 -0.1000
       0
       0
       0
       0
       0
       0
       0
```

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0.1240
0.1740
0.2114
0.2421
0.2684
0.2915
0.3121
0.3307
0.3476
0.3631
0.3773
0.3903
0.4023
0.4134
0.4236
0.4330
0.4417
0.4496
0.4569
0.4635
0.4695
0.4750
0.4798
0.4841
0.4879
0.4911
0.4939
0.4961
0.4978
0.4990
0.4998
0.5000
0.4998
0.4990
0.4978
0.4961
0.4939
0.4911
0.4879
0.4841
0.4798
0.4750
0.4695
0.4635
0.4569
0.4496
0.4417
0.4330

0.4236
0.4134
0.4023
0.3903
0.3773
0.3631
0.3476
0.3307
0.3121
0.2915
0.2684
0.2421
0.2114
0.1740
0.1240
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0.1000
0.2000
0.3000
0.4000
0.5000
0.6000
0.7000
0.8000
0.9000
1.0000
1.1000
1.2000
1.3000
1.4000
1.5000
1.5500
1.4500
1.3500
1.2500
1.1500
1.0500
0.9500

```
 0.8500
 0.7500
 0.6500
 0.5500
 0.4500
 0.3500
 0.2500
 0.1500
 0.0500
-0.0500
-0.1500
-0.2000
-0.1000
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
 0.1240
 0.1740
 0.2114
 0.2421
 0.2684
 0.2915
 0.3121
 0.3307
 0.3476
 0.3631
 0.3773
 0.3903
 0.4023
 0.4134
 0.4236
 0.4330
 0.4417
 0.4496
 0.4569
 0.4635
 0.4695
 0.4750
 0.4798
 0.4841
 0.4879
 0.4911
 0.4939
 0.4961
```

```
0.4978
0.4990
0.4998
0.5000
0.4998
0.4990
0.4978
0.4961
0.4939
0.4911
0.4879
0.4841
0.4798
0.4750
0.4695
0.4635
0.4569
0.4496
0.4417
0.4330
0.4236
0.4134
0.4023
0.3903
0.3773
0.3631
0.3476
0.3307
0.3121
0.2915
0.2684
0.2421
0.2114
0.1740
0.1240
  0
```

0
-0.3810
0
0
-0.3810
-0.3810
0.3810
0
-0.3810
0
0
-0.3810
0
0
-0.3810
0
0.3810
0
-0.3810
0
0
-0.3810
0
0
-0.3810
0
0.3810
0
-0.3810
0
0
-0.3810
0
0
-0.3810
0
0.3810
0
-0.3810
0
0
-0.3810

0
0
-0.3810
0
0.3810
0
-0.3810
0
0
-0.3810
0
0
-0.3810
0
0.3810
0
-0.3810
0
0
-0.3810
0
0
-0.3810
0
0.3810
0
-0.3810
0
0
-0.3810
0
0
-0.3810
0
0.3810
0
-0.3810
0
0
-0.3810
0
0
-0.3810

0
0.3810
0
-0.3810
0
0
-0.3810
0
0
-0.3810
0
0.3810
0
-0.3810
0
0
-0.3810
0
0
-0.3810
0
0.3810
0
-0.3810
0
0
-0.3810
0
0
-0.3810
-0.3810
0
0
-0.3810
0
0
-0.3810
0
0
-0.3810
0
0.3810
0

-0.3810
0
0
-0.3810
0
0
-0.3810
0
0.3810
0
-0.3810
0
0.3810
0.3810
0.7620
0.7620
0
0
0.3810
0
-0.3810
0
0
-0.3810
0
0
-0.3810
0
0.3810
0.7620
1.5240
2.6670
3.0480
2.2860
1.5240
0.7620
0
0
0.3810
0
-0.3810
0
0

-0.3810
0
0.3810
0.3810
1.5240
2.6670
3.0480
2.6670
3.4290
3.0480
1.9050
1.1430
0.7620
-0.3810
-0.3810
0
-0.3810
-0.3810
0
0.3810
0.7620
1.9050
3.0480
3.0480
3.0480
3.4290
3.4290
3.0480
3.4290
2.6670
1.1430
0.7620
0.3810
-0.3810
-0.3810
0
0
0.3810
1.9050
2.6670
2.6670
3.4290
3.4290

3.0480
3.4290
3.8100
3.4290
3.0480
3.0480
1.9050
0.3810
0
0
-0.3810
0
1.1430
2.2860
2.6670
3.4290
3.4290
3.0480
3.0480
3.4290
3.0480
3.0480
3.0480
2.6670
2.2860
1.9050
1.1430
0
0
0
0
0.7620
1.5240
1.5240
1.1430
1.1430
1.1430
0.7620
1.1430
1.5240
0.7620
0.3810
0.7620

0
0
0
0
-0.3810
0
0
0
0.3810
0.7620
0.7620
1.9050
6.0950
10.6670
14.0950
17.1430
19.0480
19.8100
20.1900
20.9520
20.1900
20.1900
20.9520
22.0950
24.3810
27.8100
31.6190
34.6670
37.7140
40.7620
42.6670
44.1900
45.3330
44.9520
43.8100
42.6670
41.1430
38.4760
36.1900
34.6670
33.1430
33.1430
35.0480

37.3330
39.6190
42.6670
44.9520
46.4760
47.6190
48.7620
48.7620
48.0000
47.2380
45.7140
43.0480
41.1430
38.8570
36.5710
35.0480
35.0480
35.8100
37.3330
40.3810
43.0480
45.3330
47.2380
48.3810
48.7620
48.7620
48.7620
47.6190
45.7140
43.8100
41.1430
38.0950
35.8100
34.6670
33.5240
33.9050
35.8100
37.7140
40.3810
43.0480
44.9520
46.0950
47.6190

48.0000
47.6190
46.8570
45.7140
43.8100
41.5240
39.6190
36.9520
34.6670
34.2860
34.6670
35.4290
37.7140
40.7620
43.4290
45.3330
47.6190
48.3810
48.7620
49.1430
48.3810
46.8570
45.3330
43.4290
40.3810
37.7140
35.8100
34.6670
33.9050
35.0480
37.3330
39.2380
42.2860
44.9520
46.4760
47.6190
48.3810
48.3810
46.8570
45.7140
44.1900
41.5240
39.2380

36.9520
34.2860
32.3810
32.0000
32.3810
33.1430
35.8100
38.0950
40.0000
42.2860
43.8100
44.1900
44.5710
45.3330
44.5710
43.0480
41.9050
40.0000
37.3330
34.6670
33.1430
31.6190
31.6190
33.1430
35.0480
37.3330
40.3810
42.6670
44.1900
45.3330
46.0950
45.7140
45.3330
44.5710
43.0480
40.7620
38.8570
36.1900
33.5240
32.0000
32.0000
32.3810
33.9050

36.1900
38.4760
40.3810
42.2860
43.4290
43.4290
43.8100
43.4290
42.2860
40.7620
39.6190
36.9520
33.9050
32.0000
30.4760
29.3330
29.7140
31.2380
33.1430
36.1900
39.2380
41.5240
43.0480
44.5710
44.9520
44.5710
43.8100
43.0480
40.7620
38.4760
36.1900
33.5240
31.2380
30.4760
30.4760
30.8570
33.1430
35.8100
38.0950
40.3810
42.6670
43.8100
44.1900

44.5710
44.1900
42.6670
41.1430
39.2380
36.1900
33.5240
31.6190
29.7140
28.5710
29.3330
30.8570
32.7620
35.8100
38.4760
40.0000
41.1430
42.2860
42.2860
41.9050
41.5240
40.3810
38.4760
36.1900
34.2860
31.2380
29.7140
29.3330
29.3330
30.0950
32.7620
35.4290
37.7140
40.3810
42.2860
43.0480
43.4290
43.8100
43.0480
41.5240
40.3810
38.4760
35.4290

33.1430
30.8570
28.9520
28.5710
29.7140
31.2380
33.5240
36.5710
38.8570
40.3810
41.9050
42.6670
42.2860
41.9050
41.5240
40.0000
37.3330
35.4290
32.7620
30.0950
28.5710
27.8100
27.4290
28.5710
31.6190
33.9050
36.1900
38.4760
40.0000
40.3810
41.1430
41.1430
40.0000
38.8570
37.7140
35.4290
32.3810
30.0950
28.1900
26.2860
26.2860
27.8100
29.3330

32.0000
35.0480
36.9520
38.4760
40.0000
40.3810
40.0000
39.6190
38.8570
37.3330
35.0480
33.1430
30.4760
27.8100
26.6670
25.9050
26.2860
28.1900
30.8570
33.1430
35.4290
38.0950
39.2380
40.0000
40.7620
40.7620
39.6190
38.4760
36.9520
34.2860
31.6190
29.7140
27.4290
25.9050
26.2860
27.4290
28.9520
31.6190
34.2860
36.1900
37.3330
38.8570
38.8570

38.4760
38.4760
37.3330
35.0480
33.5240
31.2380
28.5710
26.2860
25.1430
24.7620
24.7620
27.0480
29.7140
32.0000
34.6670
36.9520
37.7140
38.4760
39.2380
38.8570
37.3330
36.5710
34.6670
32.0000
29.3330
27.4290
25.1430
24.0000
25.1430
26.2860
28.1900
30.8570
33.9050
35.4290
37.3330
38.4760
38.4760
38.0950
38.0950
36.5710
34.2860
32.7620
30.0950

27.0480
25.1430
24.0000
23.6190
24.0000
26.6670
28.9520
30.8570
33.5240
35.4290
36.1900
36.9520
37.7140
36.5710
35.4290
34.6670
32.7620
30.0950
27.8100
25.9050
23.6190
22.8570
24.0000
25.1430
27.4290
30.8570
33.1430
35.0480
36.5710
37.7140
37.7140
37.7140
37.7140
35.8100
33.9050
32.0000
29.3330
26.2860
24.3810
23.2380
22.8570
24.0000
26.2860

28.1900
30.8570
33.5240
35.4290
35.8100
36.9520
37.3330
36.1900
35.4290
34.2860
32.0000
28.9520
26.6670
24.3810
22.0950
22.0950
22.8570
24.0000
26.6670
29.7140
32.0000
33.5240
35.4290
36.1900
35.8100
35.8100
35.0480
33.1430
31.2380
29.3330
26.6670
24.0000
22.4760