ANNEX 1

RECTANGULAR FIT: IMPROVED ARBENZ ALGORITHM

```
% FIT RECTANGLE
% Arbenz script (2008) modified by M.Monaco and F.Carnevale
% 2014
% Finding the best rectangle given four sets of points
% input: XY coordinates of the points sampled on the four sides
% output: vertices and errors

clear all; close all; clc;

% recall coordinates and associated errors of the points (X Y and err)
filename = 'CANO_R1.dat';
fid0 = fopen(filename,'r');
M = textscan(fid0,'%s %f %f %f %s %f %f %f %s %f %f %f %s %f %f
%f','CommentStyle','%');
label = char(M{1});
label = char(M{5});
label = char(M{9});
label = char(M{13});

XY1 = [M{2} M{3}];
XY2 = [M{6} M{7}];
XY3 = [M{10} M{11}];
XY4 = [M{14} M{15}];
err = [M{4}];
err1 = [M{4}];
err2 = [M{8}];
err3 = [M{12}];
err4 = [M{16}];

% SIDE 1
lato1= XY1;
Y1 = lato1(:,2);
N1 = length(lato1);
err1m = (sum(err1))/N1;
Y1m = mean(Y1);
sigma1 = std(Y1);

% SIDE 2
lato2 = XY2;
X2 = lato2(:,1);
N2 = length(lato2);
err2m = (sum(err2))/N2;
X2m = mean(X2);
sigma2 = std(X2);

% SIDE 3
lato3 = XY3;
Y3 = lato3(:,2);
N3 = length(lato3);
err3m = (sum(err3))/N3;
Y3m = mean(Y3);
```

```
sigma3 = std(Y3);

% SIDE 4
lato4 = XY4;
X4 = lato4(:,1);
N4 = length(lato4);
err4m = (sum(err4))/N4;
X4m = mean(X4);
sigma4 = std(X4);

% calculate the standard deviations (sigmaX and sigmaY)
% affecting the best fitting lines
  sigmaX = sqrt(sigma2^2 + sigma4^2);
  sigmaY = sqrt(sigma1^2 + sigma3^2);

% create vectors with the coordinates of the points sampled on the 4
% sides
Px = lato1(:,1);
Py = lato1(:,2);
Qx = lato2(:,1);
Qy = lato2(:,2);
Rx = lato3(:,1);
Ry = lato3(:,2);
Sx = lato4(:,1);
Sy = lato4(:,2);

% create zeros and ones vectors of the same length of the previous
   zp = zeros(size(Px)); op =  ones(size(Px));
   zq = zeros(size(Qx)); oq =  ones(size(Qx));
   zr = zeros(size(Rx)); or =  ones(size(Rx));
   zs = zeros(size(Sx)); os =  ones(size(Sx));

% create the matrix A representing the Least Square constraint
   A = [ op zp zp zp Px  Py
         zq oq zq zq Qy -Qx
         zr zr or zr Rx  Ry
         zs zs zs os Sy -Sx];

% create the vectors of constant terms c and of coefficients n (n1,
n2)
   [c, n] = clsq(A,2);

% calculate the four vertices of the best fitting rectangle
   B  = [n  [-n(2) n(1)]'];
   X = -B* [c([1 3 3 1])'; c([2 2 4 4])'];
   X';
   X = [X X(:,1)];

   pA = (X(:,1))';
   pB = (X(:,2))';
   pC = (X(:,3))';
   pD = (X(:,4))';

% display the coordinates of the 4 vertices
   disp('Vertices of the rectangle');
   disp('
');
   str1=sprintf('     A (%f,%f)' , pA(1),pA(2));
   disp(str1);
   str2=sprintf('     B (%f,%f)' , pB(1),pB(2));
   disp(str2);
   str3=sprintf('     C (%f,%f)' , pC(1),pC(2));
   disp(str3);
   str4=sprintf('     D (%f,%f)' , pD(1),pD(2));
   disp(str4);
   disp('                                                       ');

% calculate and display the errors associated to the vertex
coordinates
% (TWO values, sigma_x DIFFERENT FROM sigma_y)

   sigmaA = sqrt( sigma4^2 + sigma1^2 );
   sigmaB = sqrt( sigma2^2 + sigma1^2 );
   sigmaC = sqrt( sigma2^2 + sigma3^2 );
   sigmaD = sqrt( sigma4^2 + sigma3^2 );

   disp('Errors associated to the coordinates of the vertices');
   disp('                                                       ');
   str5=sprintf('     sigmaA (%f,%f) ' , sigma4,sigma1);
   disp(str5);
   str6=sprintf('     sigmaB (%f,%f) ' , sigma2,sigma1);
   disp(str6);
   str7=sprintf('     sigmaC (%f,%f) ' , sigma2,sigma3);
   disp(str7);
   str8=sprintf('     sigmaD (%f,%f) ' , sigma4,sigma3);
   disp(str8);
   disp('                                                       ');
   disp('

   disp('Fit Errors');
   disp('                                                       ');
   str9=sprintf('     sigmaX = %f     sigmaY = %f' , sigmaX , sigmaY);
   disp(str9);
   disp('                                                       ');

% calculate the side length (distances between the vertices)
   distAB = sqrt((pB(1) - pA(1))^2 + (pB(2) - pA(2))^2);
   distBC = sqrt((pB(1) - pC(1))^2 + (pB(2) - pC(2))^2);
   distCD = sqrt((pC(1) - pD(1))^2 + (pC(2) - pD(2))^2);
   distDA = sqrt((pA(1) - pD(1))^2 + (pA(2) - pD(2))^2);

% calculate the uncertainties related to the sides lenght
   sigmaAB1 = sqrt( ((pB(1)-pA(1))^2 * (sigmaX^2) + (pB(2)-pA(2))^2 *
(sigmaY^2)) / ((pB(1)-pA(1))^2 + (pB(2)-pA(2))^2) );
   sigmaBC1 = sqrt( ((pB(1)-pC(1))^2 * (sigmaX^2) + (pB(2)-pC(2))^2 *
(sigmaY^2)) / ((pB(1)-pC(1))^2 + (pB(2)-pC(2))^2) );
   sigmaCD1 = sqrt( ((pC(1)-pD(1))^2 * (sigmaX^2) + (pC(2)-pD(2))^2 *
(sigmaY^2)) / ((pC(1)-pD(1))^2 + (pC(2)-pD(2))^2) );
   sigmaDA1 = sqrt( ((pA(1)-pD(1))^2 * (sigmaX^2) + (pA(2)-pD(2))^2 *
(sigmaY^2)) / ((pA(1)-pD(1))^2 + (pA(2)-pD(2))^2) );

% display the sides lenght and the related uncertainties
   disp('                                                       ');
   disp('Lunghezza dei lati');
   disp('                                                       ');
   str20=sprintf('     AB1 = %f ± %f ' , distAB , sigmaAB1);
   disp(str20);
   str21=sprintf('     BC1 = %f ± %f ' , distBC , sigmaBC1);
   disp(str21);
   str22=sprintf('     CD1 = %f ± %f ' , distCD , sigmaCD1);
   disp(str22);
   str23=sprintf('     DA1 = %f ± %f ' , distDA , sigmaDA1);
   disp(str23);
   disp('                                                        ');
   disp('                                                        ');

% display the sides lenght and the related uncertainties
   disp('Lunghezza dei lati');
   disp('                                                        ');
   str10=sprintf('     AB = %f ± %f ' , distAB , sigmaAB);
   disp(str10);
   str11=sprintf('     BC = %f ± %f ' , distBC , sigmaBC);
   disp(str11);
   str12=sprintf('     CD = %f ± %f ' , distCD , sigmaCD);
   disp(str12);
   str13=sprintf('     DA = %f ± %f ' , distDA , sigmaDA);
   disp(str13);
   disp('                                                        ');
   str14=sprintf('     LATO1 = %f ± %f ' , distAB , sigmaLATO1);
   disp(str14);
   str15=sprintf('     LATO2 = %f ± %f ' , distBC , sigmaLATO2);
   disp(str15);
   disp('                                                        ');
   disp('                                                        ');

% calculate the single lines, if possible
   if all([sum(op)>1 sum(oq)>1 sum(or)>1 sum(os)>1]),
     [c1, n1] = clsq([op Px Py],2);
     [c2, n2] = clsq([oq Qx Qy],2);
     [c3, n3] = clsq([or Rx Ry],2);
     [c4, n4] = clsq([os Sx Sy],2);

% calculate the intersection points
     aaa = -[n1(1) n1(2); n2(1) n2(2)]\[c1; c2];
     bbb = -[n2(1) n2(2); n3(1) n3(2)]\[c2; c3];
     ccc = -[n3(1) n3(2); n4(1) n4(2)]\[c3; c4];
     ddd = -[n4(1) n4(2); n1(1) n1(2)]\[c4; c1];

     Y=[aaa bbb ccc ddd];
     Y';

     pA1 = (Y(:,1))';
     pB1 = (Y(:,2))';
     pC1 = (Y(:,3))';
     pD1 = (Y(:,4))';
   end

% display the intersection points
  disp('Punti di intersezione delle rette senza imporre 
ortogonalità');
  disp('                                                         ');
  str16=sprintf('     A1 (%f,%f)' , pA1(1),pA1(2));
  disp(str16);
  str17=sprintf('     B1 (%f,%f)' , pB1(1),pB1(2));
  disp(str17);
  str18=sprintf('     C1 (%f,%f)' , pC1(1),pC1(2));
  disp(str18);
  str19=sprintf('     D1 (%f,%f)' , pD1(1),pD1(2));
  disp(str19);
```

GEOMETRIC CIRCULAR FIT (RANIERI METHOD)

```
% GEOMETRIC CIRCLE FIT
% Ranieri Method
% codified by M.Monaco and F.Carnevale (2013)
% INPUT: coordinates (XY) of the points and related errors (err)
% OUPUT: coordinates of the centre (a±σa,b±σb), radius (r±σr) and
         percentage error

clear all; close all; clc;

% recall data from file.dat (XY and err)
filename = 'TB.dat';
fid0 = fopen(filename,'r');
M = textscan(fid0,'%s %f %f %f','CommentStyle','%');

label = char(M{1});
XY = [M{2} M{3}];
X = [M{2}];
Y = [M{3}];
err = [M{4}];

n = length(XY);
errm = (sum(M{4}))/n;

% calculate the number of combinations without repetition: n!/k!(n-k)!
C = combnk(1:n,3); % combinations of n elements in v taken k at a time
                   % C = COMBNK(v,k) produce a matrix with k columns
                   % and n!/k!(n-k)! rows
CsR = size(C,1);   % number of combinations without repetition

k=1;               % create a counter

for k=1:CsR        % cycle for each triplet of points
   for j=1:3       % cycle for determining the coordinates of triplets

       Xpt1 = X(C(k,1));
       Ypt1 = Y(C(k,1));
       Xpt2 = X(C(k,2));
       Ypt2 = Y(C(k,2));
       Xpt3 = X(C(k,3));
       Ypt3 = Y(C(k,3));

       pt1 = [Xpt1 Ypt1];
       pt2 = [Xpt2 Ypt2];
       pt3 = [Xpt3 Ypt3];
   end        % the input of the calcCircle function has been obtained

   [centre radius] = calcCircle(pt1,pt2,pt3);    % calculate ai, bi, ri

   cerchi(k,1:3) = [centre(1) centre(2) radius]; % for each circle

end

   ParM = mean(cerchi);
   Sigma = std(cerchi);
   EPercR = (Sigma(3)/ParM(3))*100;
   % elimination of the outliers because the St.Dev. (Sigma) is too 
high
   % selection of the circles with a and b included between
   % (best fit value - Sigma) and (best fit value + Sigma)

   h=0;
   for k=1:CsR
    if (abs(cerchi(k,1)-ParM(1))<=Sigma(1) && abs(cerchi(k,2)-
ParM(2))<=Sigma(2))

        h=h+1;
        cerchi1(h,1)=cerchi(k,1);
        cerchi1(h,2)=cerchi(k,2);
        cerchi1(h,3)=cerchi(k,3);
    end
  end                 % the matrix cerchi1 has been obtained
  clear h;

  % calculate average values (ParM1) and standard deviations (Sigma1)
  % of the matrix cerchi1
  ParM1 = mean(cerchi1);
  Sigma1 = std(cerchi1);

  % selection from the matrix cerchi1 of circles with a and b included
  % between (best fit value – Sigma1) and (best fit value + Sigma1)
  CsR2 = size(cerchi1,1);

   l=0;
   for k=1:CsR2
    if(abs(cerchi1(k,1)-ParM1(1))<= Sigma1(1) && abs(cerchi1(k,2)-
ParM1(2))<= Sigma1(2))

        l=l+1;
        cerchi2(l,1)=cerchi1(k,1);
        cerchi2(l,2)=cerchi1(k,2);
        cerchi2(l,3)=cerchi1(k,3);
    end
  end                 % the matrix cerchi2 has been obtained
  clear l;

  CsRF = size(cerchi2,1);

  ParM2 = mean(cerchi2);
  Sigma2 = std(cerchi2);
  EPercR2 = (Sigma2(3)/ParM2(3))*100;

  % display the results
  disp('RESULTS OF THE GEOMETRIC FIT');
  disp('                                                           ');
  str2=sprintf('a = %f ± %f' , ParM2(1),Sigma2(1));
  disp(str2);
  str3=sprintf('b = %f ± %f' , ParM2(2),Sigma2(2));
  disp(str3);
  str4=sprintf('r =  %f ± %f' , ParM2(3),Sigma2(3));
  disp(str4);
  disp('                                                           ');
  str12=sprintf('Perc_Error_Radius = %f' , EPercR2);
  disp(str12);
```

ALGEBRAIC CIRCULAR FIT: IMPROVED KASA ALGORITHM

```
% ALGEBRAIC KASA FIT with STANDARD DEVIATION SIGMA
% M. Monaco (2015)
% INPUT: coordinates (XY) of the points and related errors (err)
% OUTPUT: coordinates of the centre (a±σa,b±σb), radius (r±σr) and
          percentage error

clear all; close all; clc;

% recall data from file.dat (XY and err)
filename = 'TB1.dat';
fid0 = fopen(filename,'r');
M = textscan(fid0,'%s %f %f %f','CommentStyle','%');
label = char(M{1});
XY = [M{2} M{3}];
X = [M{2}];
Y = [M{3}];
err = [M{4}];
N = length(XY);
errm = (sum(M{4}))/N;

% KASA 1976
CircFitK = Kasa(XY,errm);

a = CircFitK(1);
b = CircFitK(2);
R = CircFitK(3);

N = length(XY);

% STANDARD DEVIATION of the RADIUS (SigmaR)
ris = 0;

for i=1:N
    dxi = XY(i,1)-a;
    dyi = XY(i,2)-b;

    ris = ris + (sqrt( dxi^2 + dyi^2 ) - R)^2;
end

varianza = ris/(N-1);
SigmaR = sqrt(varianza);
EPercR = (SigmaR/CircFitK(3))*100;

% STANDARD DEVIATION of the CENTRE COORDINATES (SigmaA and SigmaB)

    % calculate the two vectors of a and related deviations (ScartiA)
    A_1 = (-(-2*X) - sqrt((-2*X).^2 - 4*(X.^2+Y.^2-2*b*Y+b^2-R^2)))/2;
    A_2 = (-(-2*X) + sqrt((-2*X).^2 - 4*(X.^2+Y.^2-2*b*Y+b^2-R^2)))/2;

    ScartiA_1 = A_1 - a;
    ScartiA_2 = A_2 - a;

    AA = [A_1(:) , A_2(:)];                     % matrix AA (24x2)
    A = [A_1(:) ; A_2(:)];                      % matrix A (48)
```

```
 SA = [ScartiA_1(:) ; ScartiA_2(:)];        % matrix SCARTI A (48)
 AAA = [A(:) , SA(:)];                      % matrix AAA (48x2)
 CrAAA = size(AAA);            % dimension of the matrix AAA (48x2)

 % selection of the rows having the deviation minor than 1
 k=1;                            % create a counter
 h=0;
 for k=1:CrAAA
  if (abs(SA(k,1)) < 1)
     h=h+1;
     ScartiA(h,1)=SA(k,1);
  end
 end

 % calculate the matrix ScartiA and the value of the error SigmaA
 CrSCA = size(ScartiA,1);
 SigmaA1 = mean(ScartiA);
 SigmaA = abs(SigmaA1);

 % calculate the two vectors of b and related deviations (ScartiB)
 B_1 = (-(-2*Y) - sqrt((-2*Y).^2 - 4*(X.^2+Y.^2-2*a*X+a^2-R^2)))/2;
 B_2 = (-(-2*Y) + sqrt((-2*Y).^2 - 4*(X.^2+Y.^2-2*a*X+a^2-R^2)))/2;

 ScartiB_1 = B_1 - b;
 ScartiB_2 = B_2 - b;

 BB = [B_1(:) , B_2(:)];                    % matrix BB (24x2)
 B = [B_1(:) ; B_2(:)];                     % matrix B (48)
 SB = [ScartiB_1(:) ; ScartiB_2(:)];        % matrix SCARTI B (48)
 BBB = [B(:) , SB(:)];                      % matrix BBB (48x2)
 CrBBB = size(BBB);            % dimension of the matrix BBB (48x2)

 % selection of the rows having the deviation minor than 1
 k=1;                            % create a counter
 h=0;
 for k=1:CrBBB
  if (abs(SB(k,1)) < 1)
     h=h+1;
     ScartiB(h,1)=SB(k,1);
  end
 end

 % calculate the matrix ScartiB and the value of the error SigmaB
 CrSCB = size(ScartiB,1);
 SigmaB1 = mean(ScartiB);
 SigmaB = abs(SigmaB1);

% display the results
disp('RESULTS OF THE IMPROVED ALGEBRAIC KASA FIT');
disp('                                                           ');
str1=sprintf('a = %f ± %f'  , CircFitK(1),SigmaA);
disp(str1);
str2=sprintf('b = %f ± %f'  , CircFitK(2),SigmaB);
disp(str2);
str3=sprintf('r =  %f ± %f' , CircFitK(3),SigmaR);
disp(str3);
disp('                                                           ');
str4=sprintf('Perc_Error_Radius = %f' , EPercR);
disp(str4);
```

KASA ALGORITHM

```
function CircFitK = Kasa(XY,errm)

%------------------------------------------------------------------
% Input:  XY(n,2) is the array of coordinates of n points
%                              x(i)=XY(i,1), y(i)=XY(i,2)
%         errm is the bias of both coordinates
%
% Output: CircFitK = [a b R] is the fitting circle:
%                     center (a,b) and radius R
%------------------------------------------------------------------

P = [XY ones(size(XY,1),1)] \ [XY(:,1).^2 + XY(:,2).^2];
CircFitK = [P(1)/2 , P(2)/2 , sqrt((P(1)^2+P(2)^2)/4+P(3))];

X = XY(1);
X_quad = X^2;
Sxx = sum(X_quad);
N = length(XY);
xx = (1/N)* Sxx;
Xm = sum(X);
Xm_quad = Xm^2;
Mk = [-Xm , 0 , xx];

a = CircFitK(1);
b = CircFitK(2);
R = sqrt((P(1)^2+P(2)^2)/4+P(3));

ErrK = (CircFitK *((2*(errm^2))/R))-(((errm^2)/(xx-Xm_quad))*Mk);
ErrFit = (R*((2*(errm^2))/R))-(((errm^2)/(xx-Xm_quad))*Mk);

ScX = XY(:,1)-P(1)/2;
ScXQ = ScX.^2;
ScY = XY(:,2)-P(2)/2;
ScYQ = ScY.^2;

ris = 0;

for i=1:N
    dxi = XY(i,1)-a;
    dyi = XY(i,2)-b;

    ris = ris + (sqrt( dxi^2 + dyi^2 ) - R)^2;
end

varianza = ris/(N-1);
Sigma = sqrt(varianza);
EPercR = (sigma/CircFitK(3))*100;

end   %  Kasa
```

ALGEBRAIC CIRCULAR FIT: IMPROVED PRATT ALGORITHM

```
% ALGEBRAIC PRATT FIT with STANDARD DEVIATION SIGMA
% M. Monaco (2015)
% INPUT: coordinates (XY) of the points and related errors (err)
% OUTPUT: coordinates of the centre (a±σa,b±σb), radius (r±σr) and
percentage error

clear all; close all; clc;

% recall data from file.dat (XY and err)
filename = 'TB1.dat';
fid0 = fopen(filename,'r');
M = textscan(fid0,'%s %f %f %f','CommentStyle','%');
label = char(M{1});
XY = [M{2} M{3}];
X = [M{2}];
Y = [M{3}];
err = [M{4}];
N = length(XY);
errm = (sum(M{4}))/N;

% PRATT (1987)
CircFitP = Pratt(XY,errm);

a = CircFitP(1);
b = CircFitP(2);
R = CircFitP(3);

N = length(XY);

% STANDARD DEVIATION of the RADIUS (SigmaR)
ris = 0;

for i=1:N
    dxi = XY(i,1)-a;
    dyi = XY(i,2)-b;

    ris = ris + (sqrt( dxi^2 + dyi^2 ) - R)^2;
end

varianza = ris/(N-1);
SigmaR = sqrt(varianza);
EPercR = (SigmaR/CircFitP(3))*100;

% STANDARD DEVIATION of the CENTRE COORDINATES (SigmaA and SigmaB)

    % calculate the two vectors of a and related deviations (ScartiA)
    A_1 = (-(-2*X) - sqrt((-2*X).^2 - 4*(X.^2+Y.^2-2*b*Y+b^2-R^2)))/2;
    A_2 = (-(-2*X) + sqrt((-2*X).^2 - 4*(X.^2+Y.^2-2*b*Y+b^2-R^2)))/2;

    ScartiA_1 = A_1 - a;
    ScartiA_2 = A_2 - a;

    AA = [A_1(:) , A_2(:)];                   % matrix AA (24x2)
```

```
 A = [A_1(:) ; A_2(:)];                      % matrix A (48)
 SA = [ScartiA_1(:) ; ScartiA_2(:)];         % matrix SCARTI A (48)
 AAA = [A(:) , SA(:)];                       % matrix AAA (48x2)
 CrAAA = size(AAA);             % dimension of the matrix AAA (48x2)

 % selection of the rows having the deviation minor than 1
 k=1;                             % create a counter
 h=0;
 for k=1:CrAAA
  if (abs(SA(k,1)) < 1)
     h=h+1;
     ScartiA(h,1)=SA(k,1);
  end
 end

 % calculate the matrix ScartiA and the value of the error SigmaA
 CrSCA = size(ScartiA,1);
 SigmaA1 = mean(ScartiA);
 SigmaA = abs(SigmaA1);

 % calculate the two vectors of b and related deviations
 B_1 = (-(-2*Y) - sqrt((-2*Y).^2 - 4*(X.^2+Y.^2-2*a*X+a^2-R^2)))/2;
 B_2 = (-(-2*Y) + sqrt((-2*Y).^2 - 4*(X.^2+Y.^2-2*a*X+a^2-R^2)))/2;

 ScartiB_1 = B_1 - b;
 ScartiB_2 = B_2 - b;

 BB = [B_1(:) , B_2(:)];                     % matrix BB (24x2)
 B = [B_1(:) ; B_2(:)];                      % matrix B (48)
 SB = [ScartiB_1(:) ; ScartiB_2(:)];         % matrix SCARTI B (48)
 BBB = [B(:) , SB(:)];                       % matrice BBB (48x2)
 CrBBB = size(BBB);             % dimension of the matrix BBB (48x2)

 % selection of the rows having the deviations minor than 1
 k=1;                             % create a counter
 h=0;
 for k=1:CrBBB
  if (abs(SB(k,1)) < 1)
     h=h+1;
     ScartiB(h,1)=SB(k,1);
  end
 end

 % calculate the matrix ScartiB and the value of the error SigmaB
 CrSCB = size(ScartiB,1);
 SigmaB1 = mean(ScartiB);
 SigmaB = abs(SigmaB1);

% display the results
disp('RESULTS OF THE IMPROVED ALGEBRAIC PRATT FIT');
disp('                                                           ');
str1=sprintf('a = %f ± %f'  , CircFitP(1),SigmaA);
disp(str1);
str2=sprintf('b = %f ± %f'  , CircFitP(2),SigmaB);
disp(str2);
str3=sprintf('R =  %f ± %f' , CircFitP(3),SigmaR);
disp(str3);
disp('                                                           ');
str4=sprintf('Perc_Error_Radius = %f' , EPercR);
disp(str4);
```

PRATT ALGORITHM

```
function CircFitP = Pratt(XY,errm)

%--------------------------------------------------------------------
% Input:  XY(n,2) is the array of coordinates of n points
%                                    x(i)=XY(i,1), y(i)=XY(i,2)
%         errm is the bias of both coordinates
%
% Output: CircFitP = [a b R] is the fitting circle:
%                     center (a,b) and radius R
%--------------------------------------------------------------------

n = size(XY,1);        % number of data points

centroid = mean(XY);   % the centroid of the data set

% computing moments (note: all moments will be normed, i.e.
divided by n)

Mxx=0; Myy=0; Mxy=0; Mxz=0; Myz=0; Mzz=0;

for i=1:n
    Xi = XY(i,1) - centroid(1);  %  centering data
    Yi = XY(i,2) - centroid(2);  %  centering data
    Zi = Xi*Xi + Yi*Yi;
    Mxy = Mxy + Xi*Yi;
    Mxx = Mxx + Xi*Xi;
    Myy = Myy + Yi*Yi;
    Mxz = Mxz + Xi*Zi;
    Myz = Myz + Yi*Zi;
    Mzz = Mzz + Zi*Zi;
end

Mxx = Mxx/n;
Myy = Myy/n;
Mxy = Mxy/n;
Mxz = Mxz/n;
Myz = Myz/n;
Mzz = Mzz/n;

%    computing the coefficients of the characteristic polynomial

Mz = Mxx + Myy;
Cov_xy = Mxx*Myy - Mxy*Mxy;
Mxz2 = Mxz*Mxz;
Myz2 = Myz*Myz;

A2 = 4*Cov_xy - 3*Mz*Mz - Mzz;
A1 = Mzz*Mz + 4*Cov_xy*Mz - Mxz2 - Myz2 - Mz*Mz*Mz;
A0 = Mxz2*Myy + Myz2*Mxx - Mzz*Cov_xy - 2*Mxz*Myz*Mxy +
Mz*Mz*Cov_xy;
A22 = A2 + A2;

epsilon=1e-12;
ynew=1e+20;
IterMax=20;
```

```
xnew = 0;

%    Newton's method starting at x=0

for iter=1:IterMax
    yold = ynew;
    ynew = A0 + xnew*(A1 + xnew*(A2 + 4.*xnew*xnew));
    if (abs(ynew)>abs(yold))
        disp('Newton-Pratt goes wrong direction: |ynew| >
|yold|');
        xnew = 0;
        break;
    end
    Dy = A1 + xnew*(A22 + 16*xnew*xnew);
    xold = xnew;
    xnew = xold - ynew/Dy;
    if (abs((xnew-xold)/xnew) < epsilon), break, end
    if (iter >= IterMax)
        disp('Newton-Pratt will not converge');
        xnew = 0;
    end
    if (xnew<0.)
        fprintf(1,'Newton-Pratt negative root:  x=%f\n',xnew);
        xnew = 0;
    end
end

% computing the circle parameters

DET = xnew*xnew - xnew*Mz + Cov_xy;
Center = [Mxz*(Myy-xnew)-Myz*Mxy , Myz*(Mxx-xnew)-Mxz*Mxy]/DET/2;

CircFitP = [Center+centroid , sqrt(Center*Center'+Mz+2*xnew)];

a = CircFitP(1);
b = CircFitP(2);
R = sqrt(Center*Center'+Mz+2*xnew);

ErrP = CircFitP * ((2*(errm^2))/R);
ErrFit = R*((2*(errm^2))/R);


ris = 0;

for i=1:n
    dxi = XY(i,1)-a;
    dyi = XY(i,2)-b;

    ris = ris + (sqrt( dxi^2 + dyi^2 ) - R)^2;
end

varianza = ris/(n-1);
sigma = sqrt(varianza);
EPercR = (sigma/CircFitP(3))*100;

end    % Pratt
```

ALGEBRAIC CIRCULAR FIT: IMPROVED TAUBIN ALGORITHM

```
% ALGEBRAIC TAUBIN FIT with STANDARD DEVIATION SIGMA
% M. Monaco (2015)
% INPUT: coordinates (XY) of the points and related errors (err)
% OUTPUT: coordinates of the centre (a±σa,b±σb), radius (r±σr) and
          percentage error

clear all; close all; clc;

% recall data from file.dat (XY and err)
filename = 'TB1.dat';
fid0 = fopen(filename,'r');
M = textscan(fid0,'%s %f %f %f','CommentStyle','%');
label = char(M{1});
XY = [M{2} M{3}];
X = [M{2}];
Y = [M{3}];
err = [M{4}];
n = length(XY);
errm = (sum(M{4}))/n;

% TAUBIN 1991
CircFitT = Taubin(XY,errm);

a = CircFitT(1);
b = CircFitT(2);
R = CircFitT(3);

N = length(XY);

% STANDARD DEVIATION of the RADIUS (SigmaR)
ris = 0;

for i=1:N
    dxi = XY(i,1)-a;
    dyi = XY(i,2)-b;

    ris = ris + (sqrt( dxi^2 + dyi^2 ) - R)^2;
end

varianza = ris/(N-1);
SigmaR = sqrt(varianza);
EPercR = (SigmaR/CircFitT(3))*100;

% STANDARD DEVIATION of the CENTRE COORDINATES (SigmaA and SigmaB)

    % calculate the two vectors of a and related deviations (ScartiA)
    A_1 = (-(-2*X) - sqrt((-2*X).^2 - 4*(X.^2+Y.^2-2*b*Y+b^2-R^2)))/2;
    A_2 = (-(-2*X) + sqrt((-2*X).^2 - 4*(X.^2+Y.^2-2*b*Y+b^2-R^2)))/2;

    ScartiA_1 = A_1 - a;
    ScartiA_2 = A_2 - a;

    AA = [A_1(:) , A_2(:)];                     % matrix AA (24x2)
    A = [A_1(:) ; A_2(:)];                      % matrix A (48)
    SA = [ScartiA_1(:) ; ScartiA_2(:)];         % matrix ScartiA (48)
    AAA = [A(:) , SA(:)];                       % matrix AAA (48x2)
    CrAAA = size(AAA);            % dimension of the matrix AAA (48x2)

    % selection of the rows having the deviation minor than 1
    k=1;                            % create a counter
    h=0;
    for k=1:CrAAA
     if (abs(SA(k,1)) < 1)
        h=h+1;
        ScartiA(h,1)=SA(k,1);
     end
    end

    % calculate the matrix ScartiA and the value of the error SigmaA
    CrSCA = size(ScartiA,1);
    SigmaA1 = mean(ScartiA);
    SigmaA = abs(SigmaA1);

    % calculate the two vectors of b and related deviations (ScartiB)
    B_1 = (-(-2*Y) - sqrt((-2*Y).^2 - 4*(X.^2+Y.^2-2*a*X+a^2-R^2)))/2;
    B_2 = (-(-2*Y) + sqrt((-2*Y).^2 - 4*(X.^2+Y.^2-2*a*X+a^2-R^2)))/2;

    ScartiB_1 = B_1 - b;
    ScartiB_2 = B_2 - b;

    BB = [B_1(:) , B_2(:)];                     % matrix BB (24x2)
    B = [B_1(:) ; B_2(:)];                      % matrix B (48)
    SB = [ScartiB_1(:) ; ScartiB_2(:)];         % matrix ScartiB (48)
    BBB = [B(:) , SB(:)];                       % matrix BBB (48x2)
    CrBBB = size(BBB);            % dimension of the matrix BBB (48x2)

    % selection of the rows having the deviation minor than 1
    k=1;                            % create a counter
    h=0;
    for k=1:CrBBB
     if (abs(SB(k,1)) < 1)
        h=h+1;
        ScartiB(h,1)=SB(k,1);
     end
    end

    % calculate the matrix ScartiB and the value of the error SigmaB
    CrSCB = size(ScartiB,1);
    SigmaB1 = mean(ScartiB);
    SigmaB = abs(SigmaB1);

% display the results
   disp('RESULTS OF THE IMPROVED ALGEBRAIC TAUBIN FIT');
   disp('                                                          ');
   str1=sprintf('a = %f ± %f'  , CircFitT(1),SigmaA);
   disp(str1);
   str2=sprintf('b = %f ± %f'  , CircFitT(2),SigmaB);
   disp(str2);
   str3=sprintf('R =  %f ± %f' , CircFitT(3),SigmaR);
   disp(str3);
   disp('                                                          ');
   str4=sprintf('Errore_Perc_R = %f' , EPercR);
   disp(str4);
```

TAUBIN ALGORITHM

```
function CircFitT = Taubin(XY,errm)

%------------------------------------------------------------------
% Input:  XY(n,2) is the array of coordinates of n points
%                                  x(i)=XY(i,1), y(i)=XY(i,2)
%         errm is the bias of both coordinates
%
% Output: CircFitT = [a b R] is the fitting circle:
%                      center (a,b) and radius R
%------------------------------------------------------------------

n = size(XY,1);        % number of data points

centroid = mean(XY);   % the centroid of the data set

%     computing moments (note: all moments will be normed, i.e.
divided by n)

Mxx = 0; Myy = 0; Mxy = 0; Mxz = 0; Myz = 0; Mzz = 0;

for i=1:n
    Xi = XY(i,1) - centroid(1);  %  centering data
    Yi = XY(i,2) - centroid(2);  %  centering data
    Zi = Xi*Xi + Yi*Yi;
    Mxy = Mxy + Xi*Yi;
    Mxx = Mxx + Xi*Xi;
    Myy = Myy + Yi*Yi;
    Mxz = Mxz + Xi*Zi;
    Myz = Myz + Yi*Zi;
    Mzz = Mzz + Zi*Zi;
end

Mxx = Mxx/n;
Myy = Myy/n;
Mxy = Mxy/n;
Mxz = Mxz/n;
Myz = Myz/n;
Mzz = Mzz/n;

%    computing the coefficients of the characteristic polynomial

Mz = Mxx + Myy;
Cov_xy = Mxx*Myy - Mxy*Mxy;
A3 = 4*Mz;
A2 = -3*Mz*Mz - Mzz;
A1 = Mzz*Mz + 4*Cov_xy*Mz - Mxz*Mxz - Myz*Myz - Mz*Mz*Mz;
A0 = Mxz*Mxz*Myy + Myz*Myz*Mxx - Mzz*Cov_xy - 2*Mxz*Myz*Mxy +
Mz*Mz*Cov_xy;
A22 = A2 + A2;
A33 = A3 + A3 + A3;

xnew = 0;
ynew = 1e+20;
epsilon = 1e-12;
IterMax = 20;
```

```
% Newton's method starting at x=0

for iter=1:IterMax
    yold = ynew;
    ynew = A0 + xnew*(A1 + xnew*(A2 + xnew*A3));
    if abs(ynew) > abs(yold)
       disp('Newton-Taubin goes wrong direction: |ynew| >
|yold|');
       xnew = 0;
       break;
    end
    Dy = A1 + xnew*(A22 + xnew*A33);
    xold = xnew;
    xnew = xold - ynew/Dy;
    if (abs((xnew-xold)/xnew) < epsilon), break, end
    if (iter >= IterMax)
        disp('Newton-Taubin will not converge');
        xnew = 0;
    end
    if (xnew<0.)
        fprintf(1,'Newton-Taubin negative root:  x=%f\n',xnew);
        xnew = 0;
    end
end

%  computing the circle parameters

DET = xnew*xnew - xnew*Mz + Cov_xy;
Center = [Mxz*(Myy-xnew)-Myz*Mxy , Myz*(Mxx-xnew)-Mxz*Mxy]/DET/2;

CircFitT = [Center+centroid , sqrt(Center*Center'+Mz)];

a = CircFitT(1);
b = CircFitT(2);
R = sqrt(Center*Center'+Mz);

ErrT = CircFitT *(errm^2)/R;

ErrFit = R*(errm^2)/R;


ris = 0;

for i=1:n
    dxi = XY(i,1)-a;
    dyi = XY(i,2)-b;

    ris = ris + (sqrt( dxi^2 + dyi^2 ) - R)^2;
end

varianza = ris/(n-1);

sigma = sqrt(varianza);

EPercR = (sigma/CircFitT(3))*100;

end    %    Taubin
```

ALGEBRAIC CIRCULAR FIT: IMPROVED HYPERFIT ALGORITHM

```
% ALGEBRAIC HYPERFIT with STANDARD DEVIATION SIGMA
% M. Monaco (2015)
% INPUT: coordinates (XY) of the points and related errors (err)
% OUTPUT: coordinates of the centre (a±σa,b±σb), radius (r±σr) and
percentage error

clear all; close all; clc;

% recall data from file.dat (XY and err)
filename = 'TB1.dat';
fid0 = fopen(filename,'r');
M = textscan(fid0,'%s %f %f %f','CommentStyle','%');
label = char(M{1});
XY = [M{2} M{3}];
X = [M{2}];
Y = [M{3}];
err = [M{4}];
n = length(XY);
errm = (sum(M{4}))/n;

% HYPERFIT 2009
CircFitH = Hyper(XY);

a = CircFitH(1);
b = CircFitH(2);
R = CircFitH(3);

N = length(XY);

% STANDARD DEVIATION of the RADIUS (SigmaR)
ris = 0;

for i=1:N
    dxi = XY(i,1)-a;
    dyi = XY(i,2)-b;

    ris = ris + (sqrt( dxi^2 + dyi^2 ) - R)^2;
end

varianza = ris/(N-1);
SigmaR = sqrt(varianza);
EPercR = (SigmaR/CircFitH(3))*100;

% STANDARD DEVIATION of the CENTRE COORDINATES (SigmaA and SigmaB)

    % calculate the two vectors of a and related deviations (ScartiA)
    A_1 = (-(-2*X) - sqrt((-2*X).^2 - 4*(X.^2+Y.^2-2*b*Y+b^2-R^2)))/2;
    A_2 = (-(-2*X) + sqrt((-2*X).^2 - 4*(X.^2+Y.^2-2*b*Y+b^2-R^2)))/2;

    ScartiA_1 = A_1 - a;
    ScartiA_2 = A_2 - a;

    AA = [A_1(:) , A_2(:)];                     % matrix AA (24x2)
```

```
 A = [A_1(:) ; A_2(:)];                     % matrix A (48)
 SA = [ScartiA_1(:) ; ScartiA_2(:)];        % matrix ScartiA (48)
 AAA = [A(:) , SA(:)];                      % matrice AAA (48x2)
 CrAAA = size(AAA);            % dimension of the matrix AAA (48x2)

 % selection of the rows having the deviation minor than 1
 k=1;                            % create a counter
 h=0;
 for k=1:CrAAA
  if (abs(SA(k,1)) < 1)
     h=h+1;
     ScartiA(h,1)=SA(k,1);
  end
 end

 % calculate the matrix ScartiA and the value of the error SigmaA
 CrSCA = size(ScartiA,1);
 SigmaA1 = mean(ScartiA);
 SigmaA = abs(SigmaA1);

 % calculate the two vectors of b and related deviations (ScartiB)
 B_1 = (-(-2*Y) - sqrt((-2*Y).^2 - 4*(X.^2+Y.^2-2*a*X+a^2-R^2)))/2;
 B_2 = (-(-2*Y) + sqrt((-2*Y).^2 - 4*(X.^2+Y.^2-2*a*X+a^2-R^2)))/2;

 ScartiB_1 = B_1 - b;
 ScartiB_2 = B_2 - b;

 BB = [B_1(:) , B_2(:)];                    % matrix BB (24x2)
 B = [B_1(:) ; B_2(:)];                     % matrix B (48)
 SB = [ScartiB_1(:) ; ScartiB_2(:)];        % matrix ScartiB (48)
 BBB = [B(:) , SB(:)];                      % matrice BBB (48x2)
 CrBBB = size(BBB);            % dimension of the matrix BBB (48x2)

 % selection of the rows having the deviation minor than 1
 k=1;                            % create a counter
 h=0;
 for k=1:CrBBB
  if (abs(SB(k,1)) < 1)
     h=h+1;
     ScartiB(h,1)=SB(k,1);
  end
 end

 % calculate the matrix ScartiB and the value of the error SigmaB
 CrSCB = size(ScartiB,1);
 SigmaB1 = mean(ScartiB);
 SigmaB = abs(SigmaB1);

% display the results
disp('RESULTS OF THE IMPROVED ALGEBRAIC HYPERFIT');
disp('                                                          ');
str1=sprintf('a = %f ± %f'  , CircFitH(1),SigmaA);
disp(str1);
str2=sprintf('b = %f ± %f'  , CircFitH(2),SigmaB);
disp(str2);
str3=sprintf('r =  %f ± %f' , CircFitH(3),SigmaR);
disp(str3);
disp('                                                          ');
str4=sprintf('Perc_Error_Radius = %f' , EPercR);
disp(str4);
```

HYPERFIT ALGORITHM

```
function CircFitH = Hyper(XY)

%-----------------------------------------------------------------
%
% Algebraic circle fit with "hyperaccuracy" (with zero essential
bias)
%
% Input:  XY(n,2) is the array of coordinates of n points
%                              x(i)=XY(i,1), y(i)=XY(i,2)
%
% Output: CircFitH = [a b R] is the fitting circle:
%                    center (a,b) and radius R
%
% Note: this is a version optimized for speed, not for stability

%-----------------------------------------------------------------

X = XY(:,1);
Y = XY(:,2);
Z = X.*X + Y.*Y;
ZXY1 = [Z X Y ones(length(Z),1)];
M = ZXY1'*ZXY1;
S = mean(ZXY1);
K = [8*S(1) 4*S(2) 4*S(3) 2; 4*S(2) 1 0 0; 4*S(3) 0 1 0; 2 0 0
0];
KM = inv(K)*M;
[E,D] = eig(KM);
[Dsort,ID] = sort(diag(D));
if (Dsort(1)>0)
    disp('Error in Hyper: the smallest e-value is positive...')
end
if (Dsort(2)<0)
    disp('Error in Hyper: the second smallest e-value is
negative...')
end
A = E(:,ID(2));

CircFitH = zeros(1,3);
CircFitH(1:2) = -(A(2:3))'/A(1)/2;
CircFitH(3) = sqrt(A(2)*A(2)+A(3)*A(3)-4*A(1)*A(4))/abs(A(1))/2;

ris = 0;

N = length(XY);
a = CircFitH(1);
b = CircFitH(2);
R = CircFitH(3);

for i=1:N
    dxi = XY(i,1)-a;
    dyi = XY(i,2)-b;

    ris = ris + (sqrt( dxi^2 + dyi^2 ) - R)^2;
end
```

```
varianza = ris/(N-1);
sigma = sqrt(varianza);
EPercR = (sigma/CircFitH(3))*100;

end   %  Hyper
```