รายการอ้างอิง

1. Karl S., "Passive Position Location Estimation using the Extended Kalman Filter", IEEE Transactions on Aerospace and Electronics Systems Vol. AES-23 No. 4 (July 1987) : 558-567.
2. สุรเดช เคารพครู. การประมาณหาที่ตั้งแหล่งกำเนิดสัญญาณจากการวัดมุมทิศโดยใช้วิธีคาลแมนฟิลเตอร์แบบยืดขยาย สาขาวิชาวิศวกรรมไฟฟ้า จุฬาลงกรณ์มหาวิทยาลัย. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมไฟฟ้า บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2537.
3. Sullivan, M., "Simple methods overcome problems Least-Squares has in dealing with Outliers " Personal Engineering (October 1996) : 69-71.
4. Wu, W. and Cheng, P., "A Nonlinear IMM Algorithm for Maneuvering Target Tracking" IEEE Transactions on Aerospace and Electronics Systems AES-30 No.3 (Jul.1994) : 875-886.
5. Daeipour, E. and Bar-Shalom, Y., "An Interacting Multiple Model Approach for Target Tracking with Glint Noise", IEEE Transactions on Aerospace and Electronics Systems (November 1993) : 150-154.
6. Li and Bar-Shalom Y., " Performance Prediction of the Interacting Multiple Model Algorithm ", IEEE Transactions on Aerospace and Electronics Systems Vol. AES-29 No.3 (Jul.1993) : 755-765.
7. Alberto, L.G. Probability and Random Process for Electrical Engineering. 2nd ed. New York: Addison-Wesley, 1994. pp. 101-102, 113-117.
8. Curtis, S. Introduction to Electronic Warfare. Norwood: Artech House, 1986. pp. 109-129.
9. Averbuch, A.; Itzikowitz, S.; and Kapon, T. "Radar Target Tracking-Viterbi versus IMM" IEEE Transactions on Aerospace and Electronics Systems Vol. 27 No. 3 (May 1991) : 550-561.
10. PJD Gething. Radio Direction Finding and Superresolution. 2nd ed. London: Peter Peregrinus, 1991. pp. 296-301.
11. Wu, W., " Target Tracking with Glint Noise "IEEE Transactions on Aerospace and Electronics Systems Vol.29 No.1 (January 1993) : 174-185.
12. Torrieri, J. D., "Statistical Theory of Passive Location Systems", IEEE Transactions on Aerospace and Electronics Systems Vol AES-20 No.2 (March 1984) : 183-199.
13. Pian Totatrong. Robust High-Resolution Direction-Of-Arrival Estimatrion Signal Eigenvector Domain. Doctoral dissertations, Department of Electrical Engineering, Graduate School, University of Pittsburgh, 1993.

ภาคผนวก

ภาคผนวก ก

โปรแกรมจำลองบนเครื่องคอมพิวเตอร์
การประมาณที่ตั้งแหล่งกำเนิดสัญญาณ ด้วยอัลกอริทึมแบบไอเอ็มเอ็ม (IMM)

กรณี 3 DF เมื่อมี Laplacian Noise

```
%  IMM algorihm
%  call function of EKF, ipug1 and lnoise
clear;

% Set initial conditions

xe = 15;        ye = 15;        % Emitter's location

x1 = 0;         y1 = 0;         % DF1's location
x2 = 10;        y2 = 0;         % DF2's location
x3 = 20;        y3 = 0;         % DF3's location
XYk = [x1 x2 x3; y1 y2 y3];  % Observer location at state k

b1 = atan2((ye-y1),(xe-x1)); % Exact bearing from DF1 to the emitter
b2 = atan2((ye-y2),(xe-x2)); % Exact bearing from DF2 to the emitter
b3 = atan2((ye-y3),(xe-x3)); % Exact bearing from DF3 to the emitter

PNG = linspace(0.9,0.05,18); % Non-Gaussian probability
for gg = 1:18
   Png = PNG(gg)
   msd1 = 0.0175;          %  Model 1 standard deviation
   msd2 = 0.25; %  Model 2 standard deviation
```

```
DF1 = [x1 y1 b1 msd1 msd2]'; % DF1 data in vector form
DF2 = [x2 y2 b2 msd1 msd2]'; % DF2 data in vector form
DF3 = [x3 y3 b3 msd1 msd2]'; % DF3 data in vector form

p1k1 = 0.95;  % Initial weight of the first model, Gaussian Dist.
p2k1 = 0.05;  % Initial weight of the second model, Outlier Dist.

[x1k1,CovXY] = ipug1(DF1,DF3,PNG);   % call function of Initial Position Uncertainty
P1k1 = diag(diag(CovXY));
[x2k1,CovXY] = ipug1(DF1,DF3,PNG);   % call function of Initial Position Uncertainty
P2k1 = diag(diag(CovXY));
A = [1-Png Png; Png 1-Png]; % Model transition matrix
Phi = [1 0;0 1];          %  State transition matrix, Kalman filter
SteadyState = 1;
Observations = 40;
Monte = 100;
MErXY = eye(Monte,Observations);

t1 = clock;
for s = 1:Monte
  randn('seed',s);
  rand('seed',s);
  Runs = s;
  % call function of Lnoise
  N1 = lnoise(msd1,msd2,Png,Observations);   % Measurement noise DF 1
  N2 = lnoise(msd1,msd2,Png,Observations);   % Measurement noise DF 2
  N3 = lnoise(msd1,msd2,Png,Observations);   % Measurement noise DF 3
  B1 = b1+N1(:,1);      % Add noises to DF1
  B2 = b2+N2(:,1);      % Add noises to DF2
  B3 = b3+N3(:,1);      % Add noises to DF3
  Bk = [B1 B2 B3];
```

```
count = 0;
for Obsv = 1:Observations
  for df = 1:3
    count = count+1;

    yk = XYk(2,df); xk = XYk(1,df);   % observer location
    DFk = [xk yk Bk(Obsv,df)]';       % DF data at state k

% STEP 1 MIXING
p_1k = A(1,1)*p1k1 + A(1,2)*p2k1;     % Mixed weight of the 1st model
p_2k = A(2,1)*p1k1 + A(2,2)*p2k1;     % Mixed weight of the 2nd model
x_1k1 = A(1,1) * p1k1 * x1k1/p_1k + (A(1,2) * p2k1 * x2k1/p_1k);% Mixed estimate
x_2k1 = A(2,1) * p1k1 * x1k1/p_2k + (A(2,2) * p2k1 * x2k1/p_2k);% Mixed estimate

% Mixed covariance, 1st model
P_1k1 = A(1,1) * p1k1 * (P1k1 + (x1k1 - x_1k1) * (x1k1 - x_1k1)')/p_1k  + ...
        A(1,2) * p2k1 * (P2k1 + (x2k1 - x_1k1) * (x2k1 - x_1k1)')/p_1k;
% Mixed covariance, 2nd model
P_2k1 = A(2,1) * p1k1 * (P1k1 + (x1k1 - x_2k1) * (x1k1 - x_2k1)')/p_2k + ...
        A(2,2) * p2k1 * (P2k1 + (x2k1 - x_2k1) * (x2k1 - x_2k1)')/p_2k;

% STEP  2 FILTERING
% Put  x_ik1 and P_ik1 into corresponding Kalman filters to get the predicted
%    Pik, Sik and measurement updated XiK, PiK.
% For the Kalman filter:
%                    x_ik1 == X(k-1|k-1)
%                    P_ik1 == P(k-1|k-1)
%                       Xik == X(k|k-1)
%                       Pik == P(k|k-1)
%                       XiK == X(k|k)
%                       PiK == P(k|k)

% call EKF function
```

```
% Model 1, Gaussian
[X1K, P1K, P1k, S1k, v1k] =  fekf2(DFk, x_1k1, P_1k1, Phi, msd1);
% Model 2, Outlier
[X2K, P2K, P2k, S2k, v2k] =  fekf2(DFk, x_2k1, P_2k1, Phi, msd2);

% STEP 3 UPDATING STATISTICS
a1 = p_1k * (1/sqrt(S1k)) * exp(-0.5*v1k*v1k/S1k );
a2 = p_2k * (1/sqrt(S2k)) * exp(-0.5*v2k*v2k/S2k );
c = 1/(a1+a2);          % Nomenclature constant
p1k = c * a1; % Updated weight, model 1
p2k = c * a2; % Updated weight, model 2

% STEP 4 OUTPUT
Xk = p1k * X1K  + p2k * X2K;
Pk = p1k * (P1K + (X1K - Xk) * (X1K - Xk)')  + p2k * (P2K + (X2K - Xk) * (X2K - Xk)');

% Update parameters
p1k1 = p1k;
p2k1 = p2k;
x1k1 = X1K;
x2k1 = X2K;
P1k1 = P1K;
P2k1 = P2K;

ErX(count) = xe - Xk(1);
ErY(count) = ye - Xk(2);
ErXY(count) = (ErX(count)^2);
MErXY(s,count) = ErXY(count);
 end;  % End df for loop
end;    % End Obsv loop
end % End of Monte loop
```

```
AvgMErXY = sum(MErXY)/Monte;
for i = 1:Observations
  AvgAvg(i) = sum(AvgMErXY(1:i))/i
end
  ErrSNR(gg) = AvgAvg(3*Observations)

end % end PNG

  TotalVar = (1-PNG)*msd1^2 + 2*PNG*msd2^2;
  SNR = 10*log10(1./TotalVar);

figure
hold
plot(SNR,ErrSNR,'C')
xlabel('SNR(dB)')
ylabel('MS Position Errors')
grid
```

ภาคผนวก ข

โปรแกรมจำลองบนเครื่องคอมพิวเตอร์ วงจรกรองคาลมานแบบยืดขยาย (EKF)

```
function [Xkk,Pkk,Pkk1,Skk1,vk] = fekf(DFk,Xk1k1,Pk1k1,Phi,MSD)

%       FEKF   Extended Kalman Filter Function
%       [Xkk,Pkk] = fekf(DFk,Xk1k1,Pk1k1,Phi,MSD)
%
%       Inputs to the filter:
%       1) DFk is a vector whose elements are coordinates (X,Y) and Bk,
%          a measured bearing, of an active DF at state k; DFk = [X Y Bk]'.
%       2) Xk1k1 = X(k-1|k-1), a previous estimate at state k-1
%       3) Pk1k1 = P(k-1|k-1), an error covariance update at state k-1
%       4) Phi, a state transition matrix
%       5) MSD, Measurement Standard Deviation
%
%       Outputs of the filter:
%       1) Xkk = X(k|k), an estimate at state k, the current state
%       2) Pkk = P(k|k), an error covariance update at state k
%       3) Pkk1 = P(k|k-1), an error covariance extrapolation
%       4) Skk1 = S(k|k-1), a noise corresponding covariance extrapolation
%       The function was written specially for the method of applying EKF in
%       passive position location without process noises.

%Initialization
R = MSD^2;              % Measurement noise variance
q22 = 0.025^2;
Q = [0 0;0 q22];        % Process noise variance, can't set Q = 0,
                        %  so set q22 small as possible
Bk = DFk(3);            % Measurement at state k ; contaminated by noise
```

```
% Start EKF algorithm
%     Specify :
%         X(k|k) == Xkk
%         X(k-1|k-1) == Xk1k1
%         X(k|k-1) == Xkk1

Xkk1 = Phi*Xk1k1; % X(1|0) = Phi*X(0|0);
yekk1 = Xkk1(2); xekk1 = Xkk1(1);
yk = DFk(2); xk = DFk(1);  % observer location

u = (yekk1 - yk)/(xekk1-xk);
uy = (yekk1-yk);
ux = (xekk1-xk);

hXkk1 = atan2(uy,ux);   % h(X(k|k-1))

H11 = -u*((1+u^2) * (xekk1 - xk))^(-1);
H12 = ((1+u^2) * (xekk1 - xk))^(-1);
HXkk1 = [H11 H12];  % H(X(k|k-1))

Pkk1 = Phi*Pk1k1*Phi' + Q;    % P(k|k-1) , error covariance extrapolation

Skk1 = HXkk1*Pkk1*HXkk1'+R; % noise corresponding covariance extrapolation
vk = Bk - hXkk1;                % Estimated noise
K = Pkk1*HXkk1'*Skk1^(-1);      % Kalman gain

Xkk = Xkk1+K*(Bk - hXkk1);              % State estimate equation
Pkk = (eye(2) - K*HXkk1) * Pkk1;        % Error covariance update
```

ภาคผนวก ค

โปรแกรมจำลองบนเครื่องคอมพิวเตอร์ สัญญาณรบกวนที่ไม่เป็นแบบเกาส์

1) Laplacian Noise

```
function [N] = lnoise(SDG,SDL,PNG,Amount)

beta = SDL/sqrt(2);              % Laplacian Distribution Parameter

for i = 1:Amount

  if (rand < PNG)                % Check if Laplacian
      if(rand < 0.5)
        N(i) = beta*log(rand);
      else
        N(i) = -beta*log(rand);
      end
  else                           % Not Laplacian, so return Gaussian distribution
      N(i) = SDG*randn;
  end
end

N = N';  % Convert into vector form
```

2) Uniform Noise

```
function [N] = unoise(SDG,SDU,PNG,Amount)

beta = sqrt(3)*SDU;    % Uniform Distribution Parameter

for i = 1:Amount
  if (rand < PNG)      % Check if Uniform
      N(i) = (rand*(beta+beta))-beta;
  else                 % Not Uniform, so return Gaussian distribution
      N(i) = SDG*randn;
  end
end

N = N';  % Convert into vector form
```

โปรแกรมจำลองบนเครื่องคอมพิวเตอร์ การหา Initial Position Uncertainty

1) การหา IPU เมื่อมี Laplacian Noise

```
function  [X00,CovXY,NS] = ipug1(DF1,DF2,PNG)

% store each DFs X,Y coordinate
x1 = DF1(1);      x2 = DF2(1);
y1 = DF1(2);      y2 = DF2(2);

% store each DFs exact bearing to the emitter
b1 = DF1(3);      b2 = DF2(3);

% store each DFs' measurement standard deviations of gaussian noise
sdg1 = DF1(4);   sdg2 = DF2(4);

% store each DF's measurement standard deviations of non-gaussian noise
sdl1 = DF1(5);    sdl2 = DF2(5);

% store non-gaussian probability
NonGaussProb = PNG;

NS1 = lnoise(sdg1,sdl1,NonGauaaProb,1);
NS2 = lnoise(sdg2,sdl2,NonGaussProb,1);
NS = [NS1;NS2;];

% Add noise to bearing
beta1 = b1 + NS1;
beta2 = b2 + NS2;
```

```
theta = (atan2((y2-y1),(x2-x1))); % DF line slope

a = sqrt((x2-x1)^2 + (y2-y1)^2); % Distance between two DF's

alpha1 = beta1 +  theta; % Axis rotation
alpha2 = beta2 + theta;

d = (sin(alpha1 - alpha2))^(-2);
F11 = -a*d*sin(alpha2)*cos(alpha2);
F12 = a*d*sin(alpha1)*cos(alpha1);
F21 = -a*d*(sin(alpha2))^2;
F22 = a*d*(sin(alpha1))^2;
F = [F11 F12; F21 F22];

CovAlpha = diag([sdg1^2;sdg2^2]);
covxy  = F * CovAlpha * F';
B = [cos(theta) -sin(theta) ; sin(theta) cos(theta)];
CovXY = B' * covxy *B;

% Find Emitter Location
m1 = tan(beta1);
m2 = tan(beta2);
c1 = y1 - m1*x1;
c2 = y2 - m2*x2;
Xe = (c2 - c1)/(m1 - m2);
Ye = m1*Xe + c1;

X00 = [Xe Ye]';
```

2) การหา IPU เมื่อมี Uniform Noise

```
function  [X00,CovXY,NS] = ipug2(DF1,DF2,PNG)

% store each DFs X,Y coordinate
x1 = DF1(1);      x2 = DF2(1);
y1 = DF1(2);      y2 = DF2(2);

% store each DFs exact bearing to the emitter
b1 = DF1(3);      b2 = DF2(3);

% store each DFs' measurement standard deviations of gaussian noise
sdg1 = DF1(4);   sdg2 = DF2(4);

% store each DF's measurement standard deviations of non-gaussian noise
sdu1 = DF1(5);    sdu2 = DF2(5);

% store non-gaussian probability
NonGaussProb = PNG;

NS1 = unoise(sdg1,sdu1,NonGauaaProb,1);
NS2 = unoise(sdg2,sdu2,NonGaussProb,1);
NS = [NS1;NS2;];

% Add noise to bearing
beta1 = b1 + NS1;
beta2 = b2 + NS2;

theta = (atan2((y2-y1),(x2-x1))); % DF line slope

a = sqrt((x2-x1)^2 + (y2-y1)^2); % Distance between two DF's
```

```
alpha1 = beta1 +  theta; % Axis rotation
alpha2 = beta2 + theta;

d = (sin(alpha1 - alpha2))^(-2);
F11 = -a*d*sin(alpha2)*cos(alpha2);
F12 = a*d*sin(alpha1)*cos(alpha1);
F21 = -a*d*(sin(alpha2))^2;
F22 = a*d*(sin(alpha1))^2;
F = [F11 F12; F21 F22];

CovAlpha = diag([sdg1^2;sdg2^2]);
covxy  = F * CovAlpha * F';
B = [cos(theta) -sin(theta) ; sin(theta) cos(theta)];
CovXY = B' * covxy *B;

% Find Emitter Location
m1 = tan(beta1);
m2 = tan(beta2);
c1 = y1 - m1*x1;
c2 = y2 - m2*x2;
Xe = (c2 - c1)/(m1 - m2);
Ye = m1*Xe + c1;

X00 = [Xe Ye]';
```

ประวัติผู้เขียน

ร้อยเอก พชรพล สินธุวงศานนท์ เกิดวันที่ 27 กรกฎาคม พ.ศ. 2514 ที่กรุงเทพมหานคร สำเร็จการศึกษาปริญญาวิทยาศาสตรบัณฑิต (วทบ.ทบ) สาขาวิชา วิศวกรรมไฟฟ้า ภาควิชา วิศวกรรมไฟฟ้า โรงเรียนนายร้อยพระจุลจอมเกล้า ในปี พ.ศ. 2537 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต ที่ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปี พ.ศ. 2539 ปัจจุบันรับราชการที่ ศูนย์โทรคมนาคม กองการสื่อสาร กรมการทหารสื่อสาร ถนนพระราม 5 เขตดุสิต กรุงเทพมหานคร