รายการอ้างอิง

[1] Schleher, D.C. Introduction to Electronic Warfare. New York : Artech House, 1986.

[2] Kay, S.M. and Marple, S.L. Spectral Analysis: A Modern Perspective. Proceedings of the IEEE. Vol. 69 (Nov 1981) : 1380-1419.

[3] Johnson, J.H. The Application of Spectral Estimation Method to Bearing Estimation Problems. Proceedings of the IEEE. Vol. 70 (Sept 1982) : 1013-1028.

[4] สุรเดช เคารพครู. การประมาณหาที่ตั้งแหล่งกำเนิดสัญญาณจากการวัดมุมทิศโดยใช้วิธีคาลแมนฟิลเตอร์แบบยืดขยาย. วิทยานิพนธ์ปริญญามหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2539.

[5] เพียร โตท่าโรง. ระบบเครื่องหาทิศทางของสัญญาณวิทยุ. รายงานประจำปีศูนย์วิจัยและพัฒนาการทางทหาร กองบัญชาการทหารสูงสุด., 2537.

[6] Gething, P. Radio Direction Finding and Superresolution. London : Peter Peregrinus, 1978.

[7] Jenkins, H. Small Aperture Radio Direction-Finding. New York : Artech House, 1991.

[8] Johnson, D.H. and Dudgeon, D.E. Array Signal Processing: Concepts and Techniques. New Jersey : Prentice- Hall, 1993.

[9] Haykin, S. Adaptive Filter Theory. New Jersey : Prentice-Hall, 1996.

[10] Totarong, P. Robust-High-Resolution Direction of Arrival Estimation via Signal Eigenvector Domain. Dissertation of Doctor of Philosophy, University of Pittsburgh, August 1993.

[11] Kay, S.M. Modern Spectral Estimation: Theory and Application. New Jersey : Prentice-Hall, 1988.

[12] Makoul, J. Linear Prediction: A tutorial Review. Proceedings of the IEEE. Vol. 63 (April 1975) : 561-580.

[13] Hahn, S.L. Hilbert Transforms: The Transforms and Application Handbook. (n.p.) : CRC Press, 1996.

[14] Long, S.R. Huang, N.E. and Christensen, E.M. The Hilbert Techniques: An Alternate Approach For Non-Steady Time Series Analysis. IEEE Geoscience and Remote Sensing Society Newsletter. March 1995.

ภาคผนวก

ภาคผนวก ก

โปรแกรมจำลองบนเครื่องคอมพิวเตอร์
แบบจำลองของแหล่งกำเนิดสัญญาณ

<u>กรณีแบบจำลองของสัญญาณไซนูซอยด์</u>

```
clear
  %DEFINE MATRIX OF DELAYS,NOISE COEFFICIANS,SINUSOID SIGNAL
    bound1=input('Enter  lower limit to plot(in degree)=     ');
    bounds1=bound1*pi/180;
    bound2=input('Enter  upper limit to plot(in degree)=     ');
    bounds2=bound2*pi/180;
    L=input('Number of Sources=    ');
    m=input('Number of sensor=    ');
          if L >= m
               error('Number of sensors must greater than sources');
          end
               disp('If you do not want to add noise enter 1    ')
               disp('If you want to add noise enter 2    ')
    noi=input('Enter number =    ');
          if noi==2
               snr_dB=input('Enter signal to noise ratio(dB)=       ');
               snr=10^(snr_dB/10);
          end
    DELAY=zeros(m,1);
    FREQ=zeros(m,1);
    dummy=FREQ;
          for b=1:L
               fc=input('Enter Carrier Frequency(Hz)=    ');
               theta_chk(b)=input('Enter DOA (in degree)=  ');
               theta(b)=theta_chk(b)*pi/180;
               f(b)=fc;
```

```
            FREQ(:,b)=(f(b))*ones(size(dummy));
         if theta_chk>=0
            for a=1:m
               delay(a)=((a-1)/(2*fc))*sin(theta(b));
            end
         end
         if theta_chk<0
            for a=1:m
               delay(a)=-((a-1)/(2*fc))*sin(theta(b));
            end
         end
            delay=reshape(delay,m,1);
            DELAY(:,b)=[delay];
        end
%DEFINE MATRIX OF PHASE AND AMPLITUDE
  PHASE_=zeros(m,1);
  dummy=PHASE_;
        for b=1:L
            phase_chk(b)=input('Enter Initial Phase(in degree)=   ');
            phase(b)=phase_chk(b)*pi/180;
            PHASE_(:,b)=(phase(b))*ones(size(dummy));
        end
  AMP=zeros(m,1);
  dummy=AMP;
        for b=1:L
            amp(b)=input('Enter Initial Amplitude=   ');
            AMP(:,b)=(amp(b))*ones(size(dummy));
          if noi==2
            noi_eff(b)=amp(b)/sqrt(snr);
          end
        end
 % PARAMETERS OF DIRECTION VECTOR
  FREQ=2*pi*FREQ;
```

```
DELAY;
AMP;
PHASE=PHASE_;
n=input('Number of Snapshots=    ');
        for d=1:L
             for e=1:m
                 dummy=DELAY(e,d);
                 es=[es;dummy];
                 dummy=FREQ(e,d);
                 em=[em;dummy];
             end
        end
     u=es;
        for g=1:n-1
             u=[u es];
        end
     kk=em;
        for g=1:n-1
             kk=[kk em];
        end
 ste=input('samping rate=   ');
%TIME+DELAY
 count=0;
        for c=ste:ste:n*ste
             count=count+1;
             dummy=((c*1*ones(L*m,1))+u(:,1)).*kk(:,1);
             dummy=reshape(dummy,L*m,1);
             an=[an,dummy];
        end
 n=count;
 dummy=reshape(PHASE,L*m,1);
        for c=1:n
             bn=[bn,dummy];
```

```
        end
  dummy=reshape(AMP,L*m,1);
        for c=1:n
            cn=[cn,dummy];
        end
  FREQ=an; PHASE=bn; AMP=cn;
  index=FREQ+PHASE;
  x=exp(j*index);
  x=x.*AMP;
%NOISE ADDITIONS
      if noi==2
         for g=1:L
             dummy=noi_eff(g)*randn(m,n);
             NOISE=[NOISE;dummy];
         end
         NOI=NOISE.*j;
         NOISE=NOISE+NOI;
         y=x+NOISE;
         x=y;
      end
   xx1=x;
%COMPOUND SEVERAL  NOISE-ADDITIONED SIGNALS TO EACH SENSOR
      if L > 1
         for u=1:n
           for v=1:L-1
             for t=1:m
                 aa(t,u)=x(t,u);
                 aa(t,u)=aa(t,u)+x(t+m*v,u);
             end
           end
         end
         x=aa;
      else
```

```
        x=xx1;
    end
```

<u>กรณีแบบจำลองของสัญญาณไซนูซอยด์แบบพัลส์ ลักษณะที่ 1</u>

```
clear
  %DEFINE MATRIX OF DELAYS,NOISE COEFFICIANS,PULSE SINUSOID SIGNALS
    bound1=input('Enter lower limit to plot(in degree)=    ');
    bounds1=bound1*pi/180;
    bound2=input('Enter upper limit to plot(in degree)=    ');
    bounds2=bound2*pi/180;
    L=input('Number of Sources=   ');
    m=input('Number of sensor=   ');
    decay=input('Enter decay= ');deca=decay;
        if L >= m
            error('Number of sensors must greater than sources');
        end
            disp('If you do not want to add noise enter 1   ')
            disp('If you want to add noise enter 2   ')
    noi=input('Enter number =   ');
        if noi==2
            snr_dB=input('Enter signal to noise ratio(dB)=     ');
            snr=10^(snr_dB/10);
        end
    DELAY=zeros(m,1);
    FREQ1=zeros(m,1);
    dummy1=FREQ1;
        for b=1:L
            fc=input('Enter Carrier Frequency(Hz)=   ');
            theta_chk(b)=input('Enter DOA (in degree)=  ');
            theta(b)=theta_chk(b)*pi/180;
            f(b)=fc;
            FREQ1(:,b)=(f(b))*ones(size(dummy1));
```

```
            if theta_chk>=0
               for a=1:m
                   delay(a)=((a-1)/(2*(fc)))*sin(theta(b));
               end
            end
            if theta_chk<0
               for a=1:m
                   delay(a)=-((a-1)/(2*(fc)))*sin(theta(b));
               end
            end
               delay=reshape(delay,m,1);
               DELAY(:,b)=[delay];
          end
%DEFINE MATRIX OF PHASE AND AMPLITUDE
   PHASE_1=zeros(m,1);
   dummy1=PHASE_1;
          for b=1:L
                phase_chk1(b)=input('Enter Initial Carrier Phase(in degree)=    ');
                phase1(b)=phase_chk1(b)*pi/180;
                PHASE_1(:,b)=(phase1(b))*ones(size(dummy1));
          end
   AMP=zeros(m,1);
   dummy=AMP;
          for b=1:L
                amp(b)=input('Enter Initial Amplitude=    ');
                AMP(:,b)=(amp(b))*ones(size(dummy));
            if noi==2
                noi_eff(b)=amp(b)/sqrt(snr);
            end
          end
%PARAMETERS OF DIRECTION VECTOR
   FREQ1=2*pi*FREQ1;
   DELAY;
```

```
AMP;
PHASE1=PHASE_1;
n=input('Number of Snapshots=    ');
        for d=1:L
             for e=1:m
                 dummy=DELAY(e,d);
                 es=[es;dummy];
                 dummy=FREQ1(e,d);
                 ec=[ec;dummy];
             end
        end
    u=es;
        for g=1:n-1
             u=[u es];
        end
    kc=ec;
        for g=1:n-1
             kc=[kc ec];
        end
 ste=input('samping rate =    ');
%(TIME+DELAY)*2PI*F & (TIME+DELAY)*DECAY
 count=0;
        for c=ste:ste:n*ste
             count=count+1;
             dummy1=((c*1*ones(L*m,1))+u(:,1)).*(kc(:,1));
             dummy1=reshape(dummy1,L*m,1);
             an1=[an1,dummy1];
             dummy2=((c*1*ones(L*m,1))+u(:,1));
             dummy2=reshape(dummy2,L*m,1);
             an2=[an2,dummy2];
             dummy3=((c*1*ones(L*m,1))+u(:,1))*decay;
             dummy3=reshape(dummy3,L*m,1);
             DECAY=[DECAY,dummy3];
```

```
        end
 nnn=count;
 n=nnn;
 dummy=reshape(PHASE1,L*m,1);
        for c=1:n
             bn1=[bn1,dummy];
        end
 dummy=reshape(AMP,L*m,1);
        for c=1:n
             cn=[cn,dummy];
        end
 FREQ1=an1;t=an2; PHASE1=bn1; AMP=cn;
 index1=FREQ1+PHASE1;
 decay=DECAY;
 x1=exp(j*index1);
 decay=exp(decay);
 x=x1.*decay.*t;
 x=x.*AMP;
 x=x/max(max(real(x)));
%NOISE ADDITIONS
      if noi==2
         for g=1:L
              dummy=noi_eff(g)*randn(m,n);
              NOISE=[NOISE;dummy];
         end
         NOI=NOISE.*j;
         NOISE=NOISE+NOI;
         y=x+NOISE;x=y;
      end
  xx1=x;
%COMPOUND SEVERAL  NOISE-ADDITIONED SIGNALS TO EACH SENSOR
      if L > 1
         for u=1:n
```

```
        for v=1:L-1
          for t=1:m
                aa(t,u)=x(t,u);
                aa(t,u)=aa(t,u)+x(t+m*v,u);
          end
        end
      end
      x=xx2;
   else
      x=xx1;
   end
```

<u>กรณีแบบจำลองของสัญญาณไซนูซอยด์แบบพัลส์ ลักษณะที่ 2</u>

```
clear
   %DEFINE MATRIX OF DELAYS,NOISE COEFFICIANS,PULSE SINUSOID SIGNAL
     bound1=input('Enter lower limit to plot(in degree)=     ');
     bounds1=bound1*pi/180;
     bound2=input('Enter upper limit to plot(in degree)=     ');
     bounds2=bound2*pi/180;
     L=input('Number of Sources=    ');
     m=input('Number of sensor=    ');
     decay=input('Enter decay= ');deca=decay;
           if L >= m
                error('Number of sensors must greater than sources');
           end
                disp('If you do not want to add noise enter 1    ')
                disp('If you want to add noise enter 2    ')
     noi=input('Enter number =    ');
           if noi==2
                snr_dB=input('Enter signal to noise ratio(dB)=       ');
                snr=10^(snr_dB/10);
           end
```

```
DELAY=zeros(m,1);
FREQ1=zeros(m,1);
FREQ2=zeros(m,1);
dummy1=FREQ1;
dummy2=FREQ2;
      for b=1:L
          fc=input('Enter Carrier Frequency(Hz)=   ');
          fm=input('Enter Envelope Frequency(Hz)=   ');
          theta_chk(b)=input('Enter DOA (in degree)=  ');
          theta(b)=theta_chk(b)*pi/180;
          f(b)=fc;
          FREQ1(:,b)=(f(b))*ones(size(dummy1));
          f(b)=fm;
          FREQ2(:,b)=(f(b))*ones(size(dummy2));
        if theta_chk>=0
          for a=1:m
             delay(a)=((a-1)/(2*(fc)))*sin(theta(b));
          end
        end
        if theta_chk<0
          for a=1:m
             delay(a)=-((a-1)/(2*(fc)))*sin(theta(b));
          end
        end
          delay=reshape(delay,m,1);
          DELAY(:,b)=[delay];
      end
%DEFINE MATRIX OF PHASE AND AMPLITUDE
 PHASE_1=zeros(m,1);
 PHASE_2=zeros(m,1);
 dummy1=PHASE_1;
 dummy2=PHASE_2;
      for b=1:L
```

```
            phase_chk1(b)=input('Enter Initial Carrier Phase(in degree)=    ');
            phase1(b)=phase_chk1(b)*pi/180;
            PHASE_1(:,b)=(phase1(b))*ones(size(dummy1));
            phase_chk2(b)=input('Enter Initial Envelope Phase(in degree)=    ');
            phase2(b)=phase_chk2(b)*pi/180;
            PHASE_2(:,b)=(phase2(b))*ones(size(dummy2));
         end
  AMP=zeros(m,1);
  dummy=AMP;
         for b=1:L
            amp(b)=input('Enter Initial Amplitude=    ');
            AMP(:,b)=(amp(b))*ones(size(dummy));
          if noi==2
            noi_eff(b)=amp(b)/sqrt(snr);
          end
         end
%PARAMETERS OF DIRECTION VECTOR
  FREQ1=2*pi*FREQ1;
  FREQ2=2*pi*FREQ2;
  DELAY;
  AMP;
  PHASE1=PHASE_1;
  PHASE2=PHASE_2;
  n=input('Number of Snapshots=    ');
         for d=1:L
              for e=1:m
                 dummy=DELAY(e,d);
                 es=[es;dummy];
                 dummy=FREQ1(e,d);
                 ec=[ec;dummy];
                 dummy=FREQ2(e,d);
                 em=[em;dummy];
              end
```

```
        end
    u=es;
        for g=1:n-1
            u=[u es];
        end
    kc=ec;
        for g=1:n-1
            kc=[kc ec];
        end
    km=em;
        for g=1:n-1
            km=[km em];
        end
  ste=input('samping rate =     ');
%(TIME+DELAY)*2PI*F & (TIME+DELAY)*DECAY
  count=0;
        for c=ste:ste:n*ste
            count=count+1;
            dummy1=((c*1*ones(L*m,1))+u(:,1)).*(kc(:,1)+km(:,1));
            dummy1=reshape(dummy1,L*m,1);
            an1=[an1,dummy1];
            dummy2=((c*1*ones(L*m,1))+u(:,1)).*(kc(:,1)-km(:,1));
            dummy2=reshape(dummy2,L*m,1);
            an2=[an2,dummy2];
            dummy3=((c*1*ones(L*m,1))+u(:,1))*decay;
            dummy3=reshape(dummy3,L*m,1);
            DECAY=[DECAY,dummy3];
        end
  nnn=count;
  n=nnn;
  dummy=reshape(PHASE1,L*m,1);
        for c=1:n
            bn1=[bn1,dummy];
```

```
        end
  dummy=reshape(PHASE2,L*m,1);
        for c=1:n
            bn2=[bn2,dummy];
        end
  dummy=reshape(AMP,L*m,1);
        for c=1:n
            cn=[cn,dummy];
        end
  FREQ1=an1;FREQ2=an2; PHASE1=bn1;PHASE2=bn2; AMP=cn;
  index1=FREQ1+(PHASE1+PHASE2);
  index2=FREQ2+(PHASE1-PHASE2);
  decay=DECAY;
  x1=exp(j*index1);
  x2=exp(j*index2);
  decay=exp(decay);
  x_=(x1+x2);
  x=x_.*decay;
  x=x.*AMP;
  x=x/max(max(real(x)));
%NOISE ADDITIONS
      if noi==2
        for g=1:L
            dummy=noi_eff(g)*randn(m,n);
            NOISE=[NOISE;dummy];
        end
        NOI=NOISE.*j;
        NOISE=NOISE+NOI;
        y=x+NOISE;x=y;
      end
   xx1=x;
%COMPOUND SEVERAL  NOISE-ADDITIONED SIGNALS TO EACH SENSOR
      if L > 1
```

```
    for u=1:n
      for v=1:L-1
        for t=1:m
              aa(t,u)=x(t,u);
              aa(t,u)=aa(t,u)+x(t+m*v,u);
        end
      end
    end
    x=aa;
else
    x=xx1;
end
```

ภาคผนวก ข

โปรแกรมจำลองบนเครื่องคอมพิวเตอร์
วิธีการประมาณค่าทิศทางการมาถึงของแหล่งกำเนิดสัญญาณ

<u>วิธีบีมฟอร์มเมอร์ (Beamformer Method)</u>

```
%DIRECTION OF ARRIVAL FROM BEAMFORMER METHOD
          step=0.01*pi/180;
          count=0;
             for thet=bounds1:step:bounds2
                     count=count+1;
                     tao=0;
                if theta_chk>=0
                     for a=1:m
                        t=(a-1)/(2*(fc))*sin(thet);
                        tao=[tao;t];
                     end
                end
                if theta_chk<0
                     for a=1:m
                        t=-(a-1)/(2*(fc))*sin(thet);
                        tao=[tao;t];
                     end
                end
                     tao=tao(2:m+1,:);
                     indexbf=2*pi*(fc)*tao;
                     d=exp(i*indexbf)./sqrt(m);
                     dc=conj(d);
                     xc=conj(x);
                     sigma=x*(xc.')/(n);
                     s(count)=((dc.')*sigma)*d;
             end
```

```
ang=bounds1:step:bounds2;
ang=ang*180/pi;
maxi=max(abs(s));
s=10*log10((abs(s))/maxi);
thet=bounds1:step:bounds2;
   for fin=1:count
      if s(fin)==0
           thetabe=thet(fin);
      end
   end
thetaBe=thetabe*180/pi
figure
plot(ang,s,'k');
axis([bound1 bound2 -40 0]);
title(' Classical Beamformer');
xlabel('Direction of Arrival (degree)');
ylabel('db from peak');
```

<u>วิธีการประมาณพันธะเชิงเส้น (Linear Prediction Method) ด้วย Levinson Algorithm</u>

```
%PREPARE DATA FOR LEVINSON ALGORITHM
          if L>1
               x=xx2;
          else
               x=xx1;
          end
   step=0.01*pi/180;
   lag=m;
   mode=input('Enter mode =  ');
   ip=input('Order of Lavinson=   ');
   xl=x;
          for c=1:n
               x=xl(:,c);
```

```
y=x;
r = corr(m,lag,mode,x,y);
[a,aa,rho]=levin(r,ip);
al=[al;a];
end
a=al;
%DIRECTION OF ARRIVAL FROM LINEAR PREDICTION (LEVINSON ALGORITHM)
for c=1:n
a=[1 al(c,:)];
count=0;
for p=bounds1:step:bounds2
a= reshape(a,ip+1,1);
d=ones(1);
if theta_chk>=0
for e=1:ip
indexb=(-e)*pi*sin(p);
dummy=(exp(i*indexb));
d=[d dummy];
end
end
if theta_chk<0
for e=1:ip
indexb=(e)*pi*sin(p);
dummy=(exp(i*indexb));
d=[d dummy];
end
end
sr=1/(abs(d*a))^2;
count=count+1;
s(count)=sr;
end
ab=[ab;s];
maxi=max(abs(s));
```

```
        p=bounds1:step:bounds2;
        p=p*180/pi;
        s=10*log10((abs(s))/maxi);
        for fin=1:count
            if s(fin)==0
                theta_L=p(fin);
                thetaL=[thetaL;theta_L];
            end
        end
        plot(p,s,'k');
        axis([bound1 bound2 -40 0]);
        hold on
end
thetaL
  if n>1
      s1=mean(ab);
      maxi=max(abs(s1));
      p=bounds1:step:bounds2;
      p=p*180/pi;
      s1=10*log10((abs(s1))/maxi);
     for fin=1:count
         if s1(fin)==0
             theta_L2=p(fin);
         end
     end
      figure
      plot(p,s1,'k');
      axis([bound1 bound2 -40 0]);
      DOA_LEVINSON=mean(thetaL)
  end
title('Linear Prediction');
xlabel('Direction of Arrival (degree)');
ylabel('db from peak');
```

วิธีการประมาณพันธะเชิงเส้น (Linear Prediction Method) ด้วย Burg Algorithm

```
%PREPARE DATA FOR BURG ALGORITHM
            if L>1
                x=xx2;
            else
                x=xx1;
            end
        step=0.01*pi/180;
        ip=input('Order of Autoregressive=      ');
        xa=x;
            for c=1:n
                x=xa(:,c);
                dummy=burg(x,ip,m);
                au=[au;dummy];
            end
        a=au;
   % DIRECTION OF ARRIVAL FROM LINEAR PREDICTION (BURG ALGORITHM)
        for c=1:n
            a=[1 au(c,:)];
            count=0;
          for p=bounds1:step:bounds2
                a= reshape(a,ip+1,1);
                d=ones(1);
            if theta_chk>=0
                for e=1:ip
                    indexb=(-e)*pi*sin(p);
                    dummy=(exp(i*indexb));
                    d=[d dummy];
                end
            end
            if theta_chk<0
                for e=1:ip
```

```
                indexb=(e)*pi*sin(p);
                dummy=(exp(i*indexb));
                d=[d dummy];
            end
        end
            sr=1/(abs(d*a))^2;
            count=count+1;
            s(count)=sr;
    end
        ab=[ab;s];
        maxi=max(abs(s));
        p=bounds1:step:bounds2;
        p=p*180/pi;
        s3=10*log10((abs(s))/maxi);
        for fin=1:count
            if s3(fin)==0
                theta_B=p(fin);
                thetaB=[thetaB;theta_B];
            end
        end
        plot(p,s3,'k');
        axis([bound1 bound2 -80 0]);
        hold on
end
thetaB
    if n>1
        s31=mean(ab);
        maxi=max(abs(s31));
        p=bounds1:step:bounds2;
        p=p*180/pi;
        s31=10*log10((abs(s31))/maxi);
        for fin=1:count
            if s31(fin)==0
```

```
                thetaBurg=p(fin);
            end
        end
        figure
        plot(p,s31,'k');
        axis([bound1 bound2 -120 0]);
        DOABURG=mean(thetaB)
    end
title('Linear Prediction');
xlabel('Direction of Arrival (degree)');
ylabel('db from peak');
```

ภาคผนวก ค

โปรแกรมจำลองบนเครื่องคอมพิวเตอร์สำหรับ Correlation Estimate

```
function r = corr(n,lag,mode,x,y)
% CROSS-CORRELATION OF X AND Y
% LAG IS NUMBER OF CORRELATION SAMPLES DESIRED
% MODE=0 IS UNBIASED ESTIMATES
% MODE=OTHERS ARE BIASED ESTIMATES
% R IS VECTOR OF CORRELATION ESTIMATES
% N IS NUMBER OF DATA POINTS
% SYNTAX: r=CORR(n,lag,mode,x,y)
    for k=0:lag-1
        nk=n-k;
        sum=0;
      for b=1:nk
          sum=sum+conj(x(b))*y(b+k);
      end
      if mode==0
          r(k+1)=sum/(n-k);
      end
      if mode~=0
          r(k+1)=sum/n;
      end
    end
```

ภาคผนวก ง

โปรแกรมจำลองบนเครื่องคอมพิวเตอร์สำหรับ Levinson Algorithm

```
function [a,aa,rho]=levin(r,ip)
% LEVINSON ALGORITHM
% R IS VECTOR OF CORRELATION ESTIMATES
% ip=ORDER OF YULE-WALKER EQUATIONS TO BE SOLVED
% SYNTAX: [a,aa,rho]=levin(r,ip)
    rho0=r(1);
    aa(1,1)=-r(2)/r(1);
    rho(1)=(1-abs(aa(1,1))^2)*r(1);
    a(1)=aa(1,1);
        if ip==1
            break;
        end
        for l=2:ip
            b=-r(l+1);
            for k=1:l-1
                b=b-aa(k,l-1)*r(l+1-k);
            end
            aa(l,l)=b/rho(l-1);
            for k=1:l-1
                aa(k,l)=aa(k,l-1)+aa(l,l)*conj(aa(l-k,l-1));
            end
            rho(l)=(1-abs(aa(l,l))^2)*rho(l-1);
        end
        for l=1:ip
            a(l)=aa(l,ip);
        end
```

ภาคผนวก จ

โปรแกรมจำลองบนเครื่องคอมพิวเตอร์สำหรับ Burg Algorithm

```
function a = burg(x,ip,n)
% BURG (HARMONIC) ALGORITHM
% SYNTAX: a = BURG(x,ip,n)
% ip=ORDER OF AUTOREGRESSIVE PROCESS
% n=NUMBER OF DATA POINTS
% x=DATA POINTS
    rho0=0;
        for b=1:n
            rho0=rho0+(abs(x(b))^2)/n;
        end
        for b=2:n
            efk1(b)=x(b);
            ebk1(b-1)=x(b-1);
        end
        for k=1:ip
            sumn=0;
            sumd=0;
          for b=k+1:n
              sumn=sumn+efk1(b)*conj(ebk1(b-1));
              sumd=sumd+abs(efk1(b))^2+abs(ebk1(b-1))^2;
          end
            aa(k,k)=-2*sumn/sumd;
          if k==1
              rho(k)=(1-abs(aa(k,k))^2)*rho0;
          end
          if k>1
              rho(k)=(1-abs(aa(k,k))^2)*rho(k-1);
          end
```

```
        if ip==1
            break;
        end
        if k==1
            for b=k+2:n
                efk(b)=efk1(b)+aa(k,k)*ebk1(b-1);
                ebk(b-1)=ebk1(b-2)+conj(aa(k,k))*efk1(b-1);
            end
            for b=k+2:n
                efk1(b)=efk(b);
                ebk1(b-1)=ebk(b-1);
            end
        else
            for c=1:k-1
                aa(c,k)=aa(c,k-1)+aa(k,k)*conj(aa(k-c,k-1));
            end
            for b=k+2:n
                efk(b)=efk1(b)+aa(k,k)*ebk1(b-1);
                ebk(b-1)=ebk1(b-2)+conj(aa(k,k))*efk1(b-1);
            end
            for b=k+2:n
                efk1(b)=efk(b);
                ebk1(b-1)=ebk(b-1);
            end
        end
    end
sig2=rho(ip);
    for b=1:ip
        a(b)=aa(b,ip);
    end
```

ประวัติผู้เขียน

ร้อยเอก เอกวัฒน กิรติรัตนพฤกษ์ เกิดเมื่อวันพุธที่ 2 พฤษภาคม พ.ศ. 2516 ที่ จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาปริญญาวิทยาศาสตรบัณฑิต (สาขา วิศวกรรมไฟฟ้าสื่อสาร) จากโรงเรียนนายร้อยพระจุลจอมเกล้า รุ่นที่ 42 เมื่อปี พ.ศ. 2538 และเข้ารับการศึกษาต่อในหลักสูตร วิศวกรรมศาสตรมหาบัณฑิต สาขา วิศวกรรมไฟฟ้า ภาควิชา วิศวกรรมไฟฟ้า จุฬาลงกรณ์มหาวิทยาลัย เมื่อปี พ.ศ. 2539 ปัจจุบันปฏิบัติราชการเป็น ผู้บังคับกองร้อย กองพันนักเรียน โรงเรียนทหารสื่อสาร ค่ายกำแพงเพชรอัครโยธิน จังหวัดสมุทรสาคร