```
clf
clear
%Initial conditions:
fc = 2;
theta = 5;


phaseOut(1) = 0;    %initalize output phase to zero
for k = 1:60        %Run for this many iterations
phaseIn(k) = (2*pi*fc*k + theta);
if k == 1
    error(k) = phaseIn(k);      %initial error is equal to the input phase
    errorPrime(k) = 0;          %errorPrime can only be calculated for k >= 2
else
    error(k) = (phaseIn(k)-phaseOut(k));
    errorPrime(k) = error(k) - error(k-1);
end

%Construct Membership Functions
%error - LN MN SN, Z, SP MP LP (-50 to 50 rad)

if error(k) <= -40
LN1(k) = 1;
elseif error(k) > -40 && error(k) < -30
LN1(k) = -error(k) -3;
elseif error(k) >= -30
LN1(k) = 0;
end

if error(k) > -40  && error(k) <= -30
MN1(k) = error(k) + 4 ;
elseif error(k) > -30 && error(k) < -20
MN1(k) = -error(k) - 2;
else
MN1(k) = 0;
end

if error(k) > -30 && error(k) <= -20
SN1(k) = error(k) + 3;
elseif error(k) > -20 && error(k) < -10
SN1(k) = 1;
elseif error(k) >= -10 && error(k) < 0
SN1(k) = -error(k);
else
SN1(k) = 0;
end

if error(k) > -10 && error(k) < 0
Z1(k) = error(k) + 1;
elseif error(k) >= 0 && error(k) < 10
Z1(k) = -error(k) + 1;
else
Z1(k) = 0;
end

if error(k) > 0 && error(k) < 10
```

```
SP1(k) = error(k);
elseif error(k) >= 10 && error(k) < 20
SP1(k) = 1;
elseif error(k) >= 20 && error(k) < 30
SP1(k) = -error(k) + 3;
else
SP1(k) = 0;
end

if error(k) > 20 && error(k) < 30
MP1(k) = error(k) - 2;
elseif error(k) >= 30  && error(k) < 40
MP1(k) = -error(k) + 4 ;
else
MP1(k) = 0;
end

if error(k) <= 30
LP1(k) = 0;
elseif error(k) > 30 && error(k) < 40
LP1(k) = error(k) -3;
elseif error(k) >= 40
LP1(k) = 1;
end

%errorPrime - LN MN SN, Z, SP MP LP  (-2 rad/cycle to 2 rad/cycle)

if errorPrime(k) <= -1.6
LN2(k) = 1;
elseif errorPrime(k) > -1.6 && errorPrime(k) < -1.2
LN2(k) = -errorPrime(k)*2.5 -3;
elseif errorPrime(k) >= -1.2
LN2(k) = 0;
end

if errorPrime(k) > -1.6  && errorPrime(k) <= -1.2
MN2(k) = errorPrime(k)*2.5 + 4 ;
elseif errorPrime(k) > -1.2 && errorPrime(k) < -.8
MN2(k) = -errorPrime(k)*2.5 - 2;
else
MN2(k) = 0;
end

if errorPrime(k) > -1.2 && errorPrime(k) <= -.8
SN2(k) = errorPrime(k)*2.5 + 3;
elseif errorPrime(k) > -.8 && errorPrime(k) < -.4
SN2(k) = 1;
elseif errorPrime(k) >= -.4 && errorPrime(k) < 0
SN2(k) = -errorPrime(k)*2.5;
else
SN2(k) = 0;
end

if errorPrime(k) > -.4 && errorPrime(k) < 0
Z2(k) = errorPrime(k)*2.5 + 1;
elseif errorPrime(k) >= 0 && errorPrime(k) < .4
```

```
Z2(k) = -errorPrime(k)*2.5 + 1;
else
Z2(k) = 0;
end

if errorPrime(k) > 0 && errorPrime(k) < .4
SP2(k) = errorPrime(k)*2.5;
elseif errorPrime(k) >= .4 && errorPrime(k) < .8
SP2(k) = 1;
elseif errorPrime(k) >= .8 && errorPrime(k) < 1.2
SP2(k) = -errorPrime(k)*2.5 + 3;
else
SP2(k) = 0;
end

if errorPrime(k) > .8 && errorPrime(k) < 1.2
MP2(k) = errorPrime(k)*2.5 - 2;
elseif errorPrime(k) >= 1.2  && errorPrime(k) < 1.6
MP2(k) = -errorPrime(k)*2.5 + 4 ;
else
MP2(k) = 0;
end

if errorPrime(k) <= 1.2
LP2(k) = 0;
elseif errorPrime(k) > 1.2 && errorPrime(k) < 1.6
LP2(k) = errorPrime(k)*2.5 -3;
elseif errorPrime(k) >= 1.6
LP2(k) = 1;
end


%Construct 49 rules for each variable G1 and G2 in a 7x7 FAM table. Entries
%are PI control variables G1 and G2

%Initalize membership values to zero

LNValueG1 = zeros(7);
MNValueG1 = zeros(7);
SNValueG1 = zeros(7);
ZValueG1 = zeros(7);
SPValueG1 = zeros(7);
MPValueG1 = zeros(7);
LPValueG1 = zeros(7);

LNValueG2 = zeros(7);
MNValueG2 = zeros(7);
SNValueG2 = zeros(7);
ZValueG2 = zeros(7);
SPValueG2 = zeros(7);
MPValueG2 = zeros(7);
LPValueG2 = zeros(7);

%FAM TABLE

%Row 1
```

```
if LN1(k) ~= 0
    if LN2(k) ~=0
        LNValueG1(1,1) = min([LN1(k) LN2(k)]);
        LNValueG2(1,1) = min([LN1(k) LN2(k)]);
    end
    if MN2(k) ~= 0
        LNValueG1(1,2) = min([LN1(k) MN2(k)]);
        LNValueG2(1,2) = min([LN1(k) MN2(k)]);
    end
    if SN2(k) ~= 0
        MNValueG1(1,3) = min([LN1(k) SN2(k)]);
        MNValueG2(1,3) = min([LN1(k) SN2(k)]);
    end
     if Z2(k) ~=0
        MNValueG1(1,4) = min([LN1(k) Z2(k)]);
        MNValueG2(1,4) = min([LN1(k) Z2(k)]);
    end
    if SP2(k) ~= 0
        SNValueG1(1,5) = min([LN1(k) SP2(k)]);
        SNValueG2(1,5) = min([LN1(k) SP2(k)]);
    end
    if MP2(k) ~= 0
        SNValueG1(1,6) = min([LN1(k) MP2(k)]);
        SNValueG2(1,6) = min([LN1(k) MP2(k)]);
    end
    if LP2(k) ~= 0
        ZValueG1(1,7) = min([LN1(k) LP2(k)]);
        ZValueG2(1,7) = min([LN1(k) LP2(k)]);
    end
end

%Row 2
if MN1(k) ~= 0
    if LN2(k) ~=0
        MNValueG1(2,1) = min([MN1(k) LN2(k)]);
        LNValueG2(2,1) = min([MN1(k) LN2(k)]);
    end
    if MN2(k) ~= 0
        MNValueG1(2,2) = min([MN1(k) MN2(k)]);
        MNValueG2(2,2) = min([MN1(k) MN2(k)]);
    end
    if MN2(k) ~= 0
        MNValueG1(2,3) = min([MN1(k) SN2(k)]);
        MNValueG2(2,3) = min([MN1(k) SN2(k)]);
    end
     if Z2(k) ~=0
        SNValueG1(2,4) = min([MN1(k) Z2(k)]);
        SNValueG2(2,4) = min([MN1(k) Z2(k)]);
     end
    if SP2(k) ~= 0
        SNValueG1(2,5) = min([MN1(k) SP2(k)]);
        SNValueG2(2,5) = min([MN1(k) SP2(k)]);
    end
    if MP2(k) ~= 0
        ZValueG1(2,6) = min([MN1(k) MP2(k)]);
        ZValueG2(2,6) = min([MN1(k) MP2(k)]);
```

```
    end
    if LP2(k) ~= 0
        ZValueG1(2,7) = min([MN1(k) LP2(k)]);
        SPValueG2(2,7) = min([MN1(k) LP2(k)]);
    end
end

%Row 3
if SN1(k) ~= 0
    if LN2(k) ~=0
        SNValueG1(3,1) = min([SN1(k) LN2(k)]);
        MNValueG2(3,1) = min([SN1(k) LN2(k)]);
    end
    if MN2(k) ~= 0
        SNValueG1(3,2) = min([SN1(k) MN2(k)]);
        MNValueG2(3,2) = min([SN1(k) MN2(k)]);
    end
    if SN2(k) ~= 0
        SNValueG1(3,3) = min([SN1(k) SN2(k)]);
        SNValueG2(3,3) = min([SN1(k) SN2(k)]);
    end
     if Z2(k) ~=0
        SNValueG1(3,4) = min([SN1(k) Z2(k)]);
        SNValueG2(3,4) = min([SN1(k) Z2(k)]);
    end
    if SP2(k) ~= 0
        ZValueG1(3,5) = min([SN1(k) SP2(k)]);
        ZValueG2(3,5) = min([SN1(k) SP2(k)]);
    end
    if MP2(k) ~= 0
        ZValueG1(3,6) = min([SN1(k) MP2(k)]);
        SPValueG2(3,6) = min([SN1(k) MP2(k)]);
    end
    if LP2(k) ~= 0
        ZValueG1(3,7) = min([SN1(k) LP2(k)]);
        SPValueG2(3,7) = min([SN1(k) LP2(k)]);
    end
end

%Row 4
if Z1(k) ~= 0
    if LN2(k) ~=0
        ZValueG1(4,1) = min([Z1(k) LN2(k)]);
        MNValueG2(4,1) = min([Z1(k) LN2(k)]);
    end
    if MN2(k) ~= 0
        ZValueG1(4,2) = min([Z1(k) MN2(k)]);
        SNValueG2(4,2) = min([Z1(k) MN2(k)]);
    end
    if SN2(k) ~= 0
        ZValueG1(4,3) = min([Z1(k) SN2(k)]);
        SNValueG2(4,3) = min([Z1(k) SN2(k)]);
    end
     if Z2(k) ~=0
        ZValueG1(4,4) = min([Z1(k) Z2(k)]);
        ZValueG2(4,4) = min([Z1(k) Z2(k)]);
```

```
    end
    if SP2(k) ~= 0
        ZValueG1(4,5) = min([Z1(k) SP2(k)]);
        SPValueG2(4,5) = min([Z1(k) SP2(k)]);
    end
    if MP2(k) ~= 0
        ZValueG1(4,6) = min([Z1(k) MP2(k)]);
        SPValueG2(4,6) = min([Z1(k) MP2(k)]);
    end
    if LP2(k) ~= 0
        ZValueG1(4,7) = min([Z1(k) LP2(k)]);
        MPValueG2(4,7) = min([Z1(k) LP2(k)]);
    end
end

%Row 5
if SP1(k) ~= 0
    if LN2(k) ~=0
        ZValueG1(5,1) = min([SP1(k) LN2(k)]);
        SNValueG2(5,1) = min([SP1(k) LN2(k)]);
    end
    if MN2(k) ~= 0
        ZValueG1(5,2) = min([SP1(k) MN2(k)]);
        SNValueG2(5,2) = min([SP1(k) MN2(k)]);
    end
    if SN2(k) ~= 0
        ZValueG1(5,3) = min([SP1(k) SN2(k)]);
        ZValueG2(5,3) = min([SP1(k) SN2(k)]);
    end
     if Z2(k) ~=0
        SPValueG1(5,4) = min([SP1(k) Z2(k)]);
        SPValueG2(5,4) = min([SP1(k) Z2(k)]);
    end
    if SP2(k) ~= 0
        SPValueG1(5,5) = min([SP1(k) SP2(k)]);
        MPValueG2(5,5) = min([SP1(k) SP2(k)]);
    end
    if MP2(k) ~= 0
        SPValueG1(5,6) = min([SP1(k) MP2(k)]);
        MPValueG2(5,6) = min([SP1(k) MP2(k)]);
    end
    if LP2(k) ~= 0
        SPValueG1(5,7) = min([SP1(k) LP2(k)]);
        LPValueG2(5,7) = min([SP1(k) LP2(k)]);
    end
end

%Row 6
if MP1(k) ~= 0
    if LN2(k) ~=0
        ZValueG1(6,1) = min([MP1(k) LN2(k)]);
        SNValueG2(6,1) = min([MP1(k) LN2(k)]);
    end
    if MN2(k) ~= 0
        ZValueG1(6,2) = min([MP1(k) MN2(k)]);
        ZValueG2(6,2) = min([MP1(k) MN2(k)]);
```

```
    end
    if SN2(k) ~= 0
        SPValueG1(6,3) = min([MP1(k) SN2(k)]);
        SPValueG2(6,3) = min([MP1(k) SN2(k)]);
    end
     if Z2(k) ~=0
        SPValueG1(6,4) = min([MP1(k) Z2(k)]);
        SPValueG2(6,4) = min([MP1(k) Z2(k)]);
    end
    if SP2(k) ~= 0
        MPValueG1(6,5) = min([MP1(k) SP2(k)]);
        MPValueG2(6,5) = min([MP1(k) SP2(k)]);
    end
    if MP2(k) ~= 0
        MPValueG1(6,6) = min([MP1(k) MP2(k)]);
        MPValueG2(6,6) = min([MP1(k) MP2(k)]);
    end
    if LP2(k) ~= 0
        MPValueG1(6,7) = min([MP1(k) LP2(k)]);
        LPValueG2(6,7) = min([MP1(k) LP2(k)]);
    end
end

%Row 7
if LP1(k) ~= 0
    if LN2(k) ~=0
        ZValueG1(7,1) = min([LP1(k) LN2(k)]);
        ZValueG2(7,1) = min([LP1(k) LN2(k)]);
    end
    if MN2(k) ~= 0
        SPValueG1(7,2) = min([LP1(k) MN2(k)]);
        SPValueG2(7,2) = min([LP1(k) MN2(k)]);
    end
    if SN2(k) ~= 0
        SPValueG1(7,3) = min([LP1(k) SN2(k)]);
        SPValueG2(7,3) = min([LP1(k) SN2(k)]);
    end
     if Z2(k) ~=0
        MPValueG1(7,4) = min([LP1(k) Z2(k)]);
        MPValueG2(7,4) = min([LP1(k) Z2(k)]);
    end
    if SP2(k) ~= 0
        MPValueG1(7,5) = min([LP1(k) SP2(k)]);
        MPValueG2(7,5) = min([LP1(k) SP2(k)]);
    end
    if MP2(k) ~= 0
        LPValueG1(7,6) = min([LP1(k) MP2(k)]);
        LPValueG2(7,6) = min([LP1(k) MP2(k)]);
    end
    if LP2(k) ~= 0
        LPValueG1(7,7) = min([LP1(k) LP2(k)]);
        LPValueG2(7,7) = min([LP1(k) LP2(k)]);
    end
end

%Determine membership value of each input partition
```

```
LNFinalValueG1 = max(max(LNValueG1));
MNFinalValueG1 = max(max(MNValueG1));
SNFinalValueG1 = max(max(SNValueG1));
ZFinalValueG1 = max(max(ZValueG1));
SPFinalValueG1 = max(max(SPValueG1));
MPFinalValueG1 = max(max(MPValueG1));
LPFinalValueG1 = max(max(LPValueG1));

LNFinalValueG2 = max(max(LNValueG2));
MNFinalValueG2 = max(max(MNValueG2));
SNFinalValueG2 = max(max(SNValueG2));
ZFinalValueG2 = max(max(ZValueG2));
SPFinalValueG2 = max(max(SPValueG2));
MPFinalValueG2 = max(max(MPValueG2));
LPFinalValueG2 = max(max(LPValueG2));


%Initialize fuzzy output envelopes to zero
envelopeLNG1 = zeros(1,1000);
envelopeMNG1 = zeros(1,1000);
envelopeSNG1 = zeros(1,1000);
envelopeZG1 = zeros(1,1000);
envelopeSPG1 = zeros(1,1000);
envelopeMPG1 = zeros(1,1000);
envelopeLPG1 = zeros(1,1000);

envelopeLNG2 = zeros(1,1000);
envelopeMNG2 = zeros(1,1000);
envelopeSNG2 = zeros(1,1000);
envelopeZG2 = zeros(1,1000);
envelopeSPG2 = zeros(1,1000);
envelopeMPG2 = zeros(1,1000);
envelopeLPG2 = zeros(1,1000);

envelopeG1 = zeros(1,1000);
envelopeG2 = zeros(1,1000);


% Partition Output Space
% Construct 7 membership functions for G1 and G2
% The shape is defined here, but the universes of discourse can
    %easily be changed later

G1 = linspace(-20,-12,200);
for j = 1:200
    if G1(j) <= -16
        envelopeLNG1(j) = min(LNFinalValueG1,1);
    elseif G1(j) > -16 && G1(j) < -12
        envelopeLNG1(j) = min(LNFinalValueG1,-G1(j)*(0.25) -3);
    end
end


G1 = linspace(-16,-8,200);
for j = 1:200
```

```
    if G1(j) >= -16 && G1(j) < -12
        envelopeMNG1(j+100) = min(MNFinalValueG1,G1(j)*(0.25) +4);
    elseif G1(j) >= -12 && G1(j) < -8
        envelopeMNG1(j+100) = min(MNFinalValueG1,-G1(j)*(0.25) -2);
    end
end

G1 = linspace(-12,0,300);
for j = 1:300
    if G1(j) > -12 && G1(j) <= -8
        envelopeSNG1(j+200) = min(SNFinalValueG1,G1(j)*(0.25) + 3);
    elseif G1(j) > -8 && G1(j) < -4
        envelopeSNG1(j+200) = min(SNFinalValueG1,1);
    elseif G1(j) >= -4 && G1(j) < 0
        envelopeSNG1(j+200) = min(SNFinalValueG1,-G1(j)*(0.25));
    end
end

G1 = linspace(-4,4,200);
for j = 1:200
    if G1(j) > -4 && G1(j) < 0
        envelopeZG1(j+400) = min(ZFinalValueG1,G1(j)*(0.25) + 1);
    elseif G1(j) >= 0 && G1(j) < 4
        envelopeZG1(j+400) = min(ZFinalValueG1,-G1(j)*(0.25) + 1);
    end
end

G1 = linspace(0,12,300);
for j = 1:300
    if G1(j) > 0 && G1(j) < 4
        envelopeSPG1(j+500) = min(SPFinalValueG1,G1(j)*(0.25));
    elseif G1(j) >= 4 && G1(j) < 8
        envelopeSPG1(j+500) = min(SPFinalValueG1,1);
    elseif G1(j) >= 8 && G1(j) < 12
        envelopeSPG1(j+500) = min(SPFinalValueG1,-G1(j)*(0.25)+3);
    end
end

G1 = linspace(8,16,200);
for j = 1:200
    if G1(j) > 8 && G1(j) < 12
        envelopeMPG1(j+700) = min(MPFinalValueG1,G1(j)*(0.25) - 2);
    elseif G1(j) >= 12  && G1(j) < 16
        envelopeMPG1(j+700) = min(MPFinalValueG1,-G1(j)*(0.25) + 4);
    end
end

G1 = linspace(12,20,200);
for j = 1:200
    if G1(j) > 12 && G1(j) < 16
        envelopeLPG1(j+800) = min(LPFinalValueG1,G1(j)*(0.25) -3);
    elseif G1(j) >= 16
        envelopeLPG1(j+800) = min(LPFinalValueG1,1);
    end
end
```

```
G2 = linspace(-6,-3.6,200);
for j = 1:200
    if G2(j) <= -4.8
        envelopeLNG2(j) = min(LNFinalValueG2,1);
    elseif G2(j) > -4.8 && G2(j) < -3.6
        envelopeLNG2(j) = min(LNFinalValueG2,-G2(j)*(5/6) -3);
    end
end


G2 = linspace(-4.8,-2.4,200);
for j = 1:200
    if G2(j) >= -4.8 && G2(j) < -3.6
        envelopeMNG2(j+100) = min(MNFinalValueG2,G2(j)*(5/6) +4);
    elseif G2(j) >= -3.6 && G2(j) < -2.4
        envelopeMNG2(j+100) = min(MNFinalValueG2,-G2(j)*(5/6) -2);
    end
end

G2 = linspace(-3.6,0,300);
for j = 1:300
    if G2(j) > -3.6 && G2(j) <= -2.4
        envelopeSNG2(j+200) = min(SNFinalValueG2,G2(j)*(5/6) + 3);
    elseif G2(j) > -2.4 && G2(j) < -1.2
        envelopeSNG2(j+200) = min(SNFinalValueG2,1);
    elseif G2(j) >= -1.2 && G2(j) < 0
        envelopeSNG2(j+200) = min(SNFinalValueG2,-G2(j)*(5/6));
    end
end

G2 = linspace(-1.2,1.2,200);
for j = 1:200
    if G2(j) > -1.2 && G2(j) < 0
        envelopeZG2(j+400) = min(ZFinalValueG2,G2(j)*(5/6) + 1);
    elseif G2(j) >= 0 && G2(j) < 1.2
        envelopeZG2(j+400) = min(ZFinalValueG2,-G2(j)*(5/6) + 1);
    end
end

G2 = linspace(0,3.6,300);
for j = 1:300
    if G2(j) > 0 && G2(j) < 1.2
        envelopeSPG2(j+500) = min(SPFinalValueG2,G2(j)*(5/6));
    elseif G2(j) >= 1.2 && G2(j) < 2.4
        envelopeSPG2(j+500) = min(SPFinalValueG2,1);
    elseif G2(j) >= 2.4 && G2(j) < 3.6
        envelopeSPG2(j+500) = min(SPFinalValueG2,-G2(j)*(5/6) + 3);
    end
end

G2 = linspace(2.4,4.8,200);
for j = 1:200
    if G2(j) > 2.4 && G2(j) < 3.6
        envelopeMPG2(j+700) = min(MPFinalValueG2,G2(j)*(5/6) - 2);
    elseif G2(j) >= 3.6  && G2(j) < 4.8
```

```
        envelopeMPG2(j+700) = min(MPFinalValueG2,-G2(j)*(5/6) + 4);
    end
end

G2 = linspace(3.6,6,200);
for j = 1:200
    if G2(j) > 3.6 && G2(j) < 4.8
        envelopeLPG2(j+800) = min(LPFinalValueG2,G2(j)*(5/6) -3);
    elseif G2(j) >= 4.8
        envelopeLPG2(j+800) = min(LPFinalValueG2,1);
    end
end

%Create output variable envelopes for finding the centroid.
for i = 1:1000
    envelopeG1(i) = max([envelopeLNG1(i) envelopeMNG1(i) envelopeSNG1(i) envelopeZG1(i) enve
    envelopeG2(i) = max([envelopeLNG2(i) envelopeMNG2(i) envelopeSNG2(i) envelopeZG2(i) enve
end

%Defuzzification (centroid method)

summationMuG1 = 0; summationMuG1TimesG1 = 0;
summationMuG2 = 0; summationMuG2TimesG2 = 0;

uG1 = linspace(-1,1,1000);
uG2 = linspace(-2.5,2.5,1000);
for h = 1:1000
MuG1(h) = envelopeG1(h);
MuG2(h) = envelopeG2(h);

summationMuG1 = summationMuG1 + MuG1(h);
summationMuG1TimesG1 = summationMuG1TimesG1 + MuG1(h)*uG1(h);

summationMuG2 = summationMuG2 + MuG2(h);
summationMuG2TimesG2 = summationMuG2TimesG2 + MuG1(h)*uG2(h);
end
g1(k) = summationMuG1TimesG1/summationMuG1;     %centroid
g2(k) = summationMuG2TimesG2/summationMuG2;

envelopeG1Graph{k} = envelopeG1; %keep track of envelopes for future reference
envelopeG2Graph{k} = envelopeG2;

%Fuzzy PI Control
y(k) =  g1(k) + sum(g2);

%VCO
if k==1
    phaseOut(k+1) = 0;  %intialize output phase to zero
else
phaseOut(k+1)=phaseOut(k)+y(k); %lock to input
end

k %print iteration #
end

plot(error,'LineWidth',2)
```

```
title('Error vs Iteration Number');
xlabel('Iteration Number');
ylabel('Error');
```