Contents

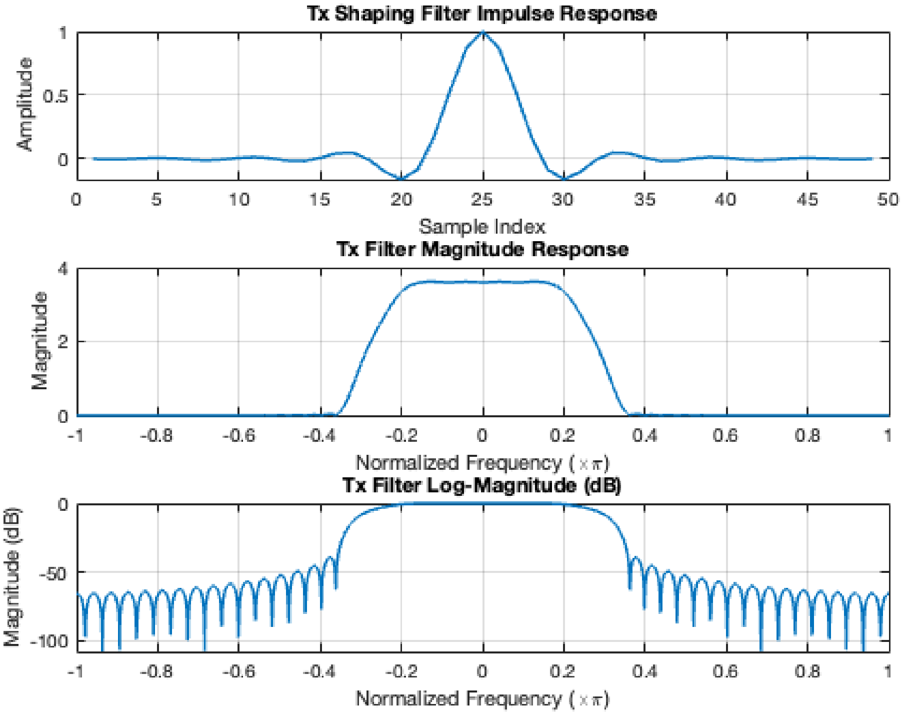
=== PLL Project===

```
close all force; clear; clc;
```

Part A: Transmit & Receive Filters

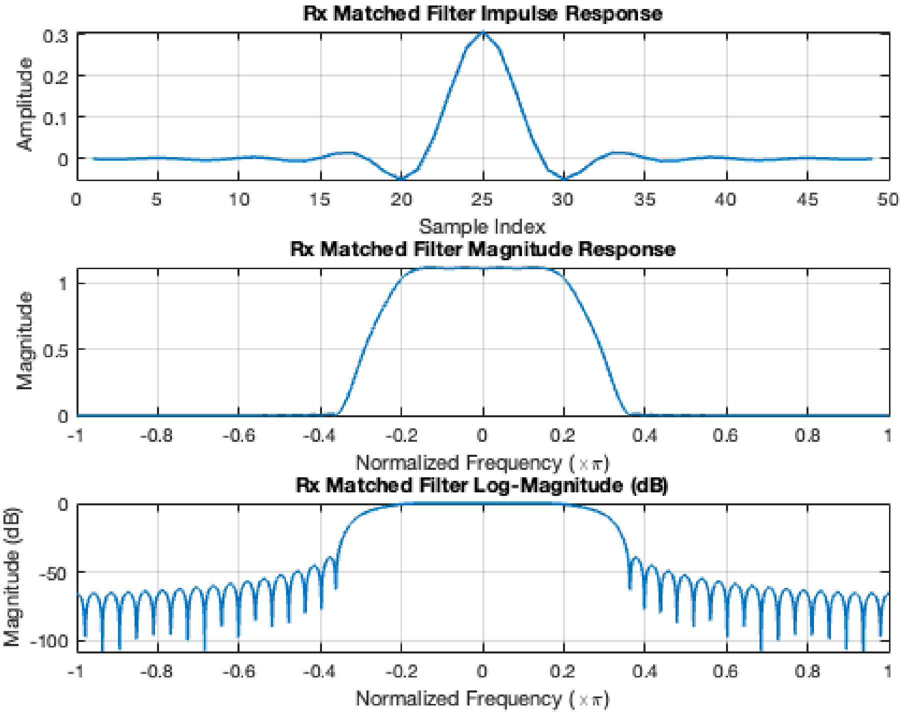
```
samplesPerSymbol   = 4;
rollOffFactor      = 0.4;
groupDelaySymbols  = 6;
symbolRate         = 1;  % normalized

% transmit sqrt-Nyquist shaping filter
txFilter = sqrtNyquistFilter(symbolRate, samplesPerSymbol, rollOffFactor, groupDelaySymbols);
txFilter = txFilter / max(abs(txFilter));

% receive matched filter
rxMatchedFilter = txFilter / (txFilter * txFilter');

% frequency axis for plotting
freqAxis = linspace(-1, 1, 2048);

figure;
subplot(3,1,1);
plot(txFilter, 'LineWidth',1.5); grid on;
title('Tx Shaping Filter Impulse Response');
xlabel('Sample Index'); ylabel('Amplitude');

subplot(3,1,2);
txFreqResp = abs(fftshift(fft(txFilter,2048)));
plot(freqAxis, txFreqResp, 'LineWidth',1.5); grid on;
title('Tx Filter Magnitude Response');
xlabel('Normalized Frequency (\times\pi)'); ylabel('Magnitude');

subplot(3,1,3);
plot(freqAxis, 20*log10(txFreqResp/max(txFreqResp)), 'LineWidth',1.5); grid on;
title('Tx Filter Log-Magnitude (dB)');
xlabel('Normalized Frequency (\times\pi)'); ylabel('Magnitude (dB)');
```

```matlab
figure;
subplot(3,1,1);
plot(rxMatchedFilter, 'LineWidth',1.5); grid on;
title('Rx Matched Filter Impulse Response');
xlabel('Sample Index'); ylabel('Amplitude');

subplot(3,1,2);
rxFreqResp = abs(fftshift(fft(rxMatchedFilter,2048)));
plot(freqAxis, rxFreqResp, 'LineWidth',1.5); grid on;
title('Rx Matched Filter Magnitude Response');
xlabel('Normalized Frequency (\times\pi)'); ylabel('Magnitude');

subplot(3,1,3);
plot(freqAxis, 20*log10(rxFreqResp/max(rxFreqResp)), 'LineWidth',1.5); grid on;
title('Rx Matched Filter Log-Magnitude (dB)');
xlabel('Normalized Frequency (\times\pi)'); ylabel('Magnitude (dB)');
```


Tx Shaping Filter Impulse Response

Tx Filter Magnitude Response

Tx Filter Log-Magnitude (dB)


Rx Matched Filter Impulse Response

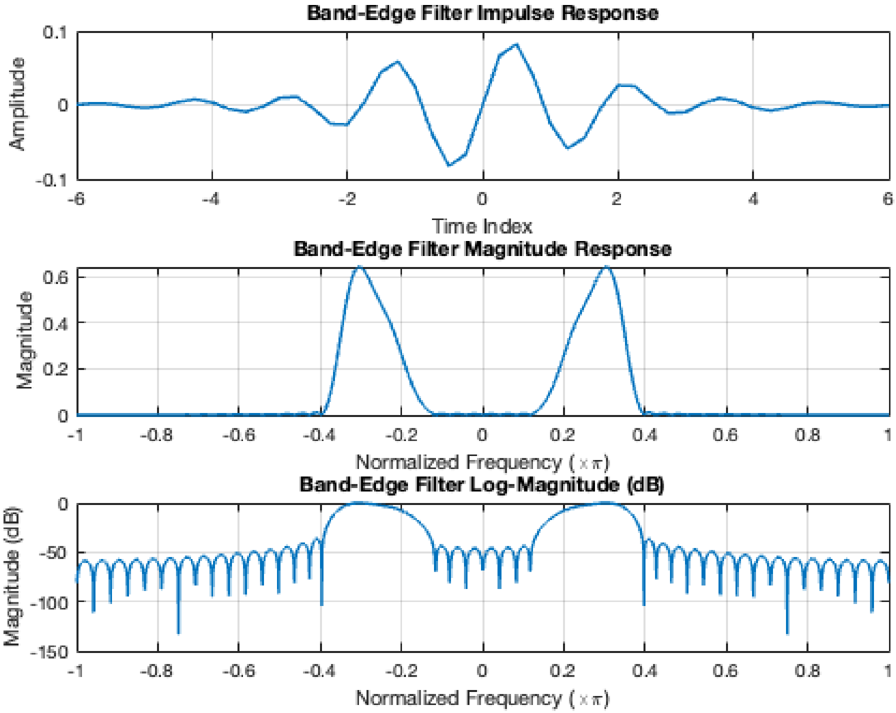
Rx Matched Filter Magnitude Response

Rx Matched Filter Log-Magnitude (dB)

Part B: Band-Edge Matched Filter

```
timeIndices = -groupDelaySymbols : 1/samplesPerSymbol : groupDelaySymbols;
windowKE    = kaiser(length(timeIndices), 3)';
bandEdgeFilter = rxMatchedFilter .* timeIndices .* windowKE;

figure;
subplot(3,1,1);
plot(timeIndices, bandEdgeFilter, 'LineWidth',1.5); grid on;
```

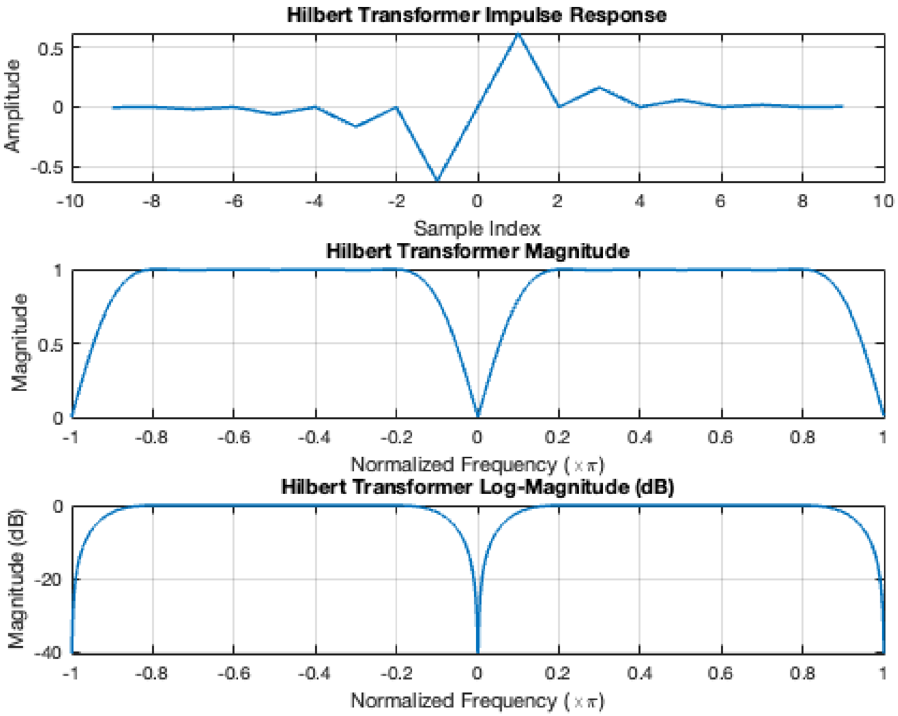
```
title('Band-Edge Filter Impulse Response');
xlabel('Time Index'); ylabel('Amplitude');

subplot(3,1,2);
beFreqResp = abs(fftshift(fft(bandEdgeFilter,2048)));
plot(freqAxis, beFreqResp, 'LineWidth',1.5); grid on;
title('Band-Edge Filter Magnitude Response');
xlabel('Normalized Frequency (\times\pi)'); ylabel('Magnitude');

subplot(3,1,3);
plot(freqAxis, 20*log10(beFreqResp/max(beFreqResp)), 'LineWidth',1.5); grid on;
title('Band-Edge Filter Log-Magnitude (dB)');
xlabel('Normalized Frequency (\times\pi)'); ylabel('Magnitude (dB)');
```



Part C: Hilbert Transformer

```
htOrder  = 9;
windowHB = 5;
hilbertCoeffs = (2/pi)*[-1/9 0 -1/7 0 -1/5 0 -1/3 0 -1 0 1 0 1/3 0 1/5 0 1/7 0 1/9];
hilbertWindow = kaiser(length(hilbertCoeffs), windowHB)';
hilbertFilter  = hilbertCoeffs .* hilbertWindow;

figure;
subplot(3,1,1);
tHT = -htOrder:htOrder;
plot(tHT, hilbertFilter, 'LineWidth',1.5); grid on;
title('Hilbert Transformer Impulse Response');
xlabel('Sample Index'); ylabel('Amplitude');

subplot(3,1,2);
htFreqResp = abs(fftshift(fft(hilbertFilter,2048)));
plot(freqAxis, htFreqResp, 'LineWidth',1.5); grid on;
title('Hilbert Transformer Magnitude');
xlabel('Normalized Frequency (\times\pi)'); ylabel('Magnitude');
```

```
subplot(3,1,3);
plot(freqAxis, 20*log10(htFreqResp/max(htFreqResp)), 'LineWidth',1.5); grid on;
title('Hilbert Transformer Log-Magnitude (dB)');
xlabel('Normalized Frequency (\times\pi)'); ylabel('Magnitude (dB)');
```



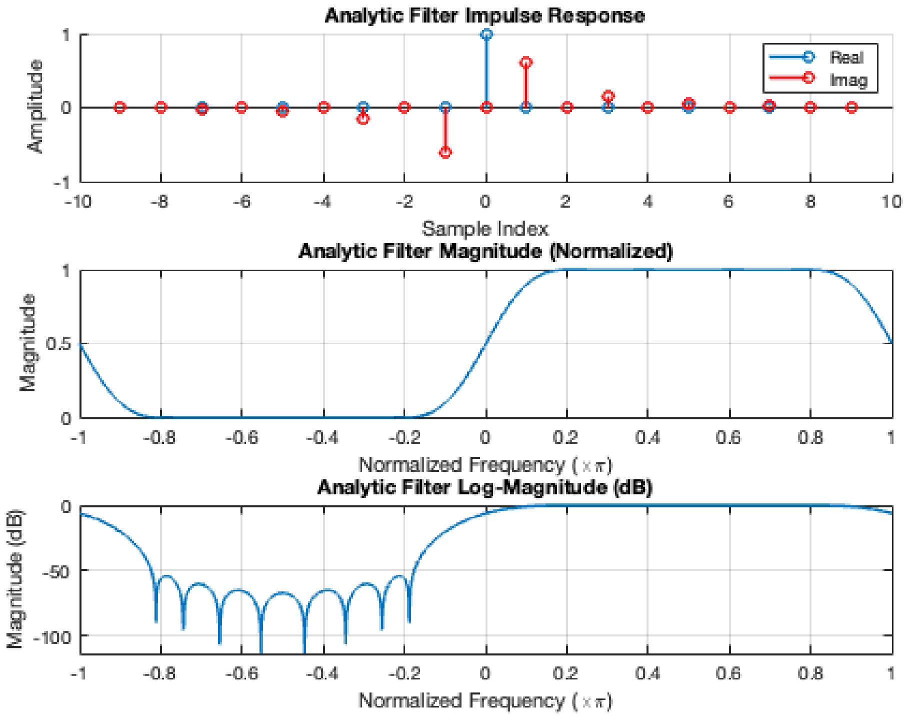
Hilbert Transformer Impulse Response

Hilbert Transformer Magnitude

Hilbert Transformer Log-Magnitude (dB)

Part D: Analytic Signal Filter

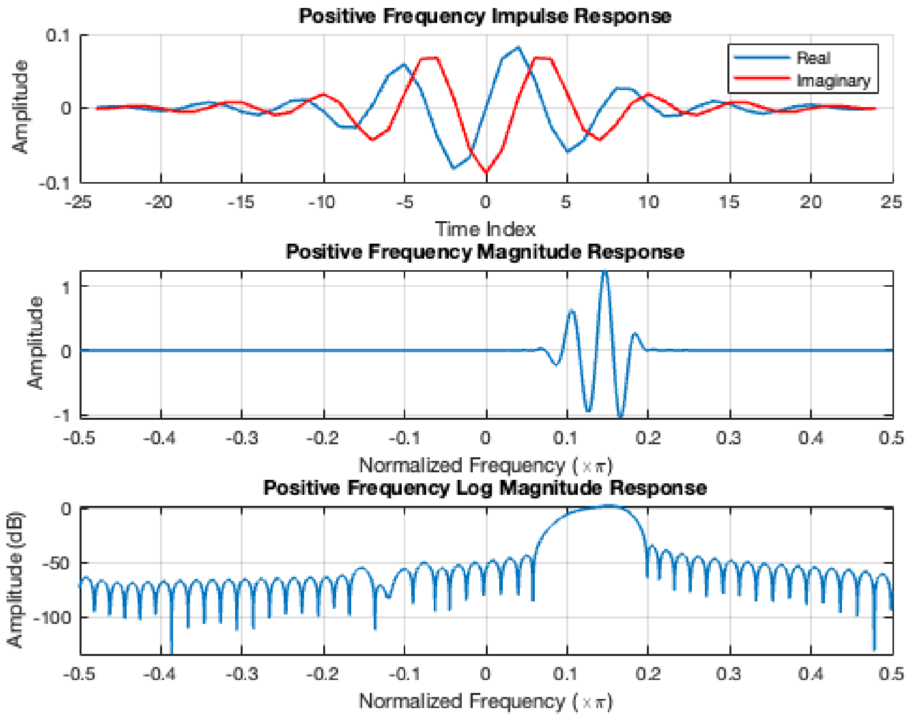
```
analyticFilter = 1j * hilbertFilter;
centerIndex    = htOrder + 1;
analyticFilter(centerIndex) = 1 + 1j*0;

figure;
subplot(3,1,1); hold on;
stem(tHT, real(analyticFilter), 'LineWidth',1.5);
stem(tHT, imag(analyticFilter), 'LineWidth',1.5,'Color','r');
hold off; grid on; legend('Real','Imag');
title('Analytic Filter Impulse Response');
xlabel('Sample Index'); ylabel('Amplitude');

subplot(3,1,2);
afFreqResp = abs(fftshift(fft(analyticFilter,2048)));
plot(freqAxis, afFreqResp/max(afFreqResp), 'LineWidth',1.5); grid on;
title('Analytic Filter Magnitude (Normalized)');
xlabel('Normalized Frequency (\times\pi)'); ylabel('Magnitude');

subplot(3,1,3);
plot(freqAxis, 20*log10(afFreqResp/max(afFreqResp)), 'LineWidth',1.5); grid on;
title('Analytic Filter Log-Magnitude (dB)');
xlabel('Normalized Frequency (\times\pi)'); ylabel('Magnitude (dB)');
```



Part E: Positive-Frequency Filter

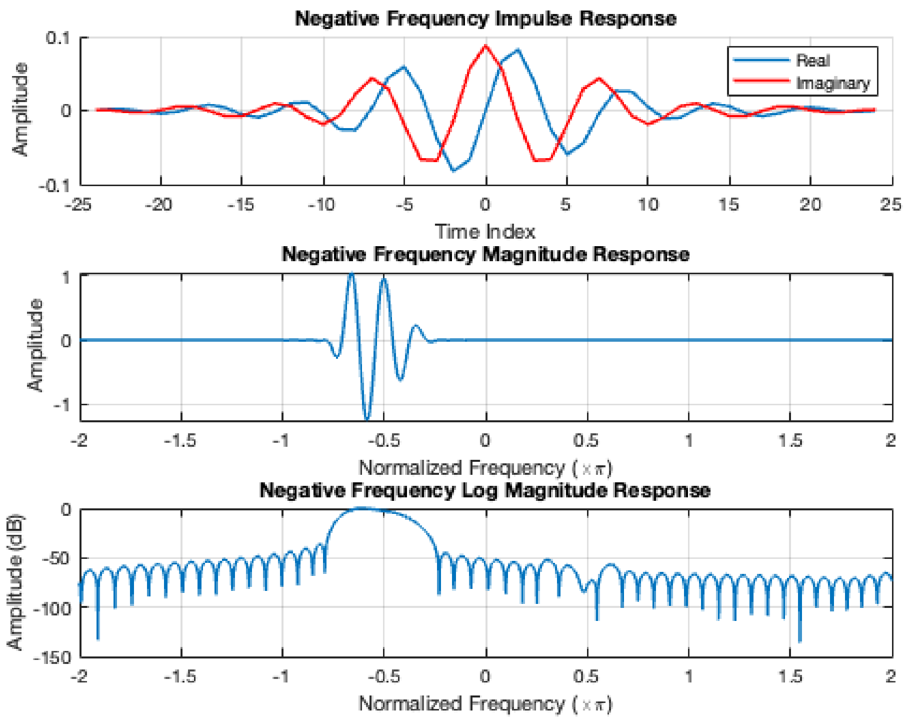
```
convResult          = conv(bandEdgeFilter, analyticFilter);
positiveFreqFilter  = convResult(10 : 58);  % extract taps 10–58

% Plot impulse response
figure;
subplot(3,1,1);
hold on;
timeSamples = linspace(-samplesPerSymbol*groupDelaySymbols, ...
                       samplesPerSymbol*groupDelaySymbols, ...
                       length(positiveFreqFilter));
plot(timeSamples, real(positiveFreqFilter), 'LineWidth',1.5);
plot(timeSamples, imag(positiveFreqFilter), 'LineWidth',1.5, 'Color','r');
hold off;
legend('Real','Imaginary');
grid on;
title('Positive Frequency Impulse Response');
xlabel('Time Index');
ylabel('Amplitude');

% Plot “magnitude” (imag part per example)
subplot(3,1,2);
posMag = abs(fftshift(fft(positiveFreqFilter,2048)));
plot(linspace(-0.5,0.5,2048), imag(fftshift(fft(positiveFreqFilter,2048))), 'LineWidth',1.5);
grid on;
title('Positive Frequency Magnitude Response');
xlabel('Normalized Frequency (\times\pi)');
ylabel('Amplitude');

% Plot log-magnitude
subplot(3,1,3);
plot(linspace(-0.5,0.5,2048), 20*log10(posMag), 'LineWidth',1.5);
grid on;
title('Positive Frequency Log Magnitude Response');
```

```
xlabel('Normalized Frequency (\times\pi)');
ylabel('Amplitude (dB)');
```



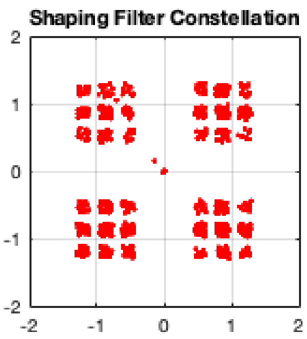
Part F: Negative-Frequency Filter

```
negativeFreqFilter = conj(positiveFreqFilter);

figure;
subplot(3,1,1);
hold on;
plot(timeSamples, real(negativeFreqFilter), 'LineWidth',1.5);
plot(timeSamples, imag(negativeFreqFilter), 'LineWidth',1.5, 'Color','r');
hold off;
legend('Real','Imaginary');
grid on;
title('Negative Frequency Impulse Response');
xlabel('Time Index');
ylabel('Amplitude');

% Plot “magnitude” (imag part per example)
subplot(3,1,2);
negMag = abs(fftshift(fft(negativeFreqFilter,2048)));
plot(linspace(-2,2,2048), imag(fftshift(fft(negativeFreqFilter,2048))), 'LineWidth',1.5);
grid on;
title('Negative Frequency Magnitude Response');
xlabel('Normalized Frequency (\times\pi)');
ylabel('Amplitude');

% Plot log-magnitude
subplot(3,1,3);
plot(linspace(-2,2,2048), 20*log10(negMag / max(negMag)), 'LineWidth',1.5);
grid on;
title('Negative Frequency Log Magnitude Response');
xlabel('Normalized Frequency (\times\pi)');
ylabel('Amplitude (dB)');
```

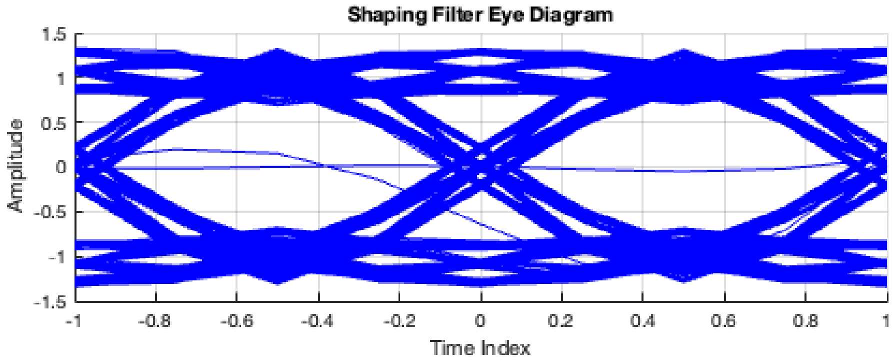


Part G: Shaping Filter Output & Eye Diagram

```
numSymbols    = 1100;
dataSymbols   = (2*round(rand(1,numSymbols))-1) + 1j*(2*round(rand(1,numSymbols))-1);
upsampledLen  = samplesPerSymbol * numSymbols;
txCoeffsMat   = reshape([zeros(1,samplesPerSymbol-1), txFilter], samplesPerSymbol, []);
shiftRegister = zeros(1, size(txCoeffsMat,2));
shapedSignal  = zeros(1, upsampledLen);

idx = 1;
for symIdx = 1:numSymbols
    shiftRegister = [dataSymbols(symIdx), shiftRegister(1:end-1)];
    for sampIdx = 1:samplesPerSymbol
        shapedSignal(idx) = shiftRegister * txCoeffsMat(sampIdx,:)';
        idx = idx + 1;
    end
end

figure;
subplot(2,1,1);
plot(shapedSignal(1:samplesPerSymbol:end), 'r.'); grid on; axis equal;
axis([-2 2 -2 2]);
title('Shaping Filter Constellation');

subplot(2,1,2); hold on;
for n = (groupDelaySymbols*2+2) : (samplesPerSymbol*2) : (upsampledLen - samplesPerSymbol)
    plot(linspace(-1,1,samplesPerSymbol*2+1), real(shapedSignal(n:n+samplesPerSymbol*2)), 'b');
end
hold off; grid on;
title('Shaping Filter Eye Diagram');
xlabel('Time Index'); ylabel('Amplitude');
```



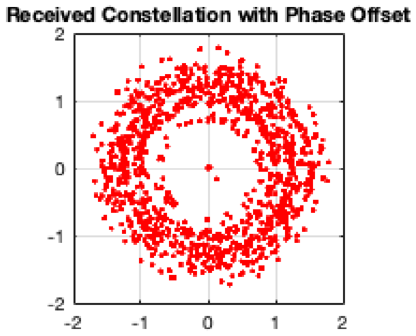
Shaping Filter Constellation

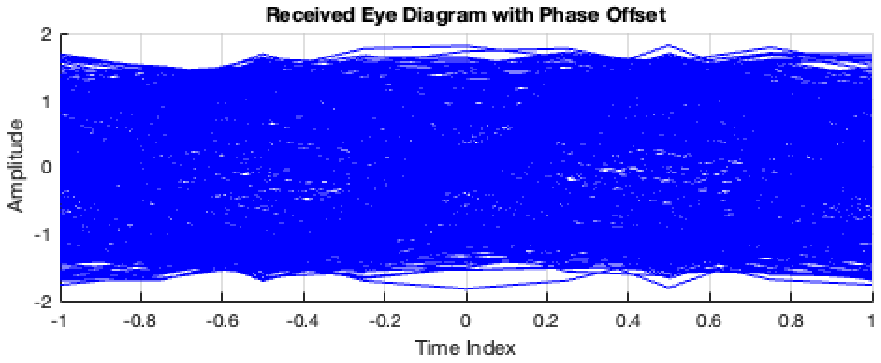


Shaping Filter Eye Diagram

Part H: Received Signal with Phase Offset

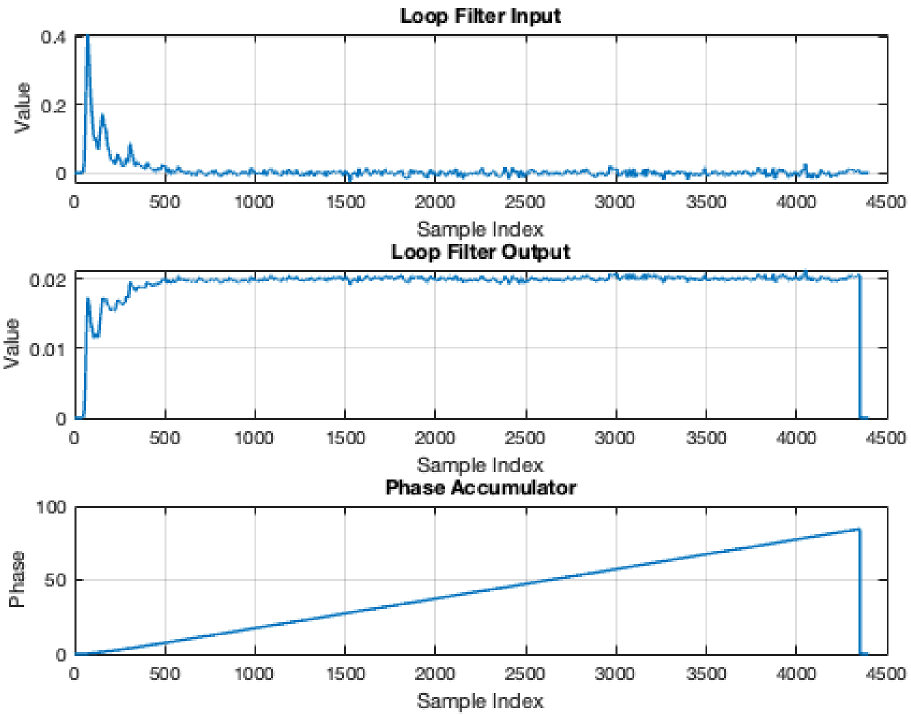
```
freqOffsetNorm = 0.02;
timeVec        = 0 : (upsampledLen-1);
receivedSignal = shapedSignal .* exp(1j*2*pi*timeVec*freqOffsetNorm);

figure;
subplot(2,1,1);
plot(receivedSignal(1:samplesPerSymbol:end), 'r.'); grid on; axis equal;
axis([-2 2 -2 2]);
title('Received Constellation with Phase Offset');

subplot(2,1,2); hold on;
for n = (groupDelaySymbols*2+2) : (samplesPerSymbol*2) : (upsampledLen - samplesPerSymbol)
    plot(linspace(-1,1,samplesPerSymbol*2+1), real(receivedSignal(n:n+samplesPerSymbol*2)), 'b');
end
hold off; grid on;
title('Received Eye Diagram with Phase Offset');
xlabel('Time Index'); ylabel('Amplitude');
```





Part I: PLL Loop Filter & Phase Tracking

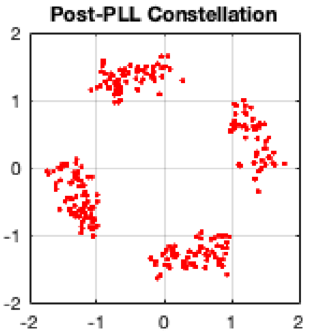
```
numTaps        = length(rxMatchedFilter);        % = 49
pllReg         = zeros(1, numTaps);              % proper 1×49 shift register
phaseAcc       = 0;
intState       = 0;
leakyState     = 0;
alphaLeak      = 0.95;
phaseStep0     = 2*pi/500;
dampingFactor = sqrt(2)/2;
ki             = (4*phaseStep0^2)/(1 + 2*dampingFactor*phaseStep0 + phaseStep0^2);
kp             = (4*dampingFactor*phaseStep0)/(1 + 2*dampingFactor*phaseStep0 + phaseStep0^2);

loopLen        = length(receivedSignal);
loopInLog      = zeros(1, loopLen);
loopOutLog     = zeros(1, loopLen);
phaseAccLog    = zeros(1, loopLen);
despunSignal = zeros(1, loopLen);     % <— pre-allocate

for n = 1:(loopLen - numTaps)
    % 1) Despin by current phase estimate
    despunSignal(n) = receivedSignal(n)*exp(-1j*2*pi*phaseAcc);
    % 2) Update PLL shift register (49 taps)
    pllReg = [despunSignal(n), pllReg(1:end-1)];
    % 3) Matched-filter output (you probably want to use this somewhere)
    matchOut = pllReg * rxMatchedFilter.';
    % 4) Band-edge filters power
    yBEp = pllReg * positiveFreqFilter.';
    yBEn = pllReg * negativeFreqFilter.';
    diffPower = abs(yBEp)^2 - abs(yBEn)^2;
    % 5) Loop-filter
    leakyState = alphaLeak*leakyState + (1-alphaLeak)*diffPower;
    intState    = intState + ki*leakyState;
    loopOut     = intState + kp*leakyState;
    % 6) Save logs
    loopInLog(n)    = leakyState;
    loopOutLog(n)   = loopOut;
    phaseAccLog(n) = phaseAcc;
    % 7) Update phase accumulator
    phaseAcc = phaseAcc + loopOut;
end

figure;
subplot(3,1,1);
plot(loopInLog, 'LineWidth',1.5); grid on;
title('Loop Filter Input');
xlabel('Sample Index'); ylabel('Value');

subplot(3,1,2);
plot(loopOutLog, 'LineWidth',1.5); grid on;
title('Loop Filter Output');
xlabel('Sample Index'); ylabel('Value');

subplot(3,1,3);
plot(phaseAccLog, 'LineWidth',1.5); grid on;
title('Phase Accumulator');
xlabel('Sample Index'); ylabel('Phase');
```

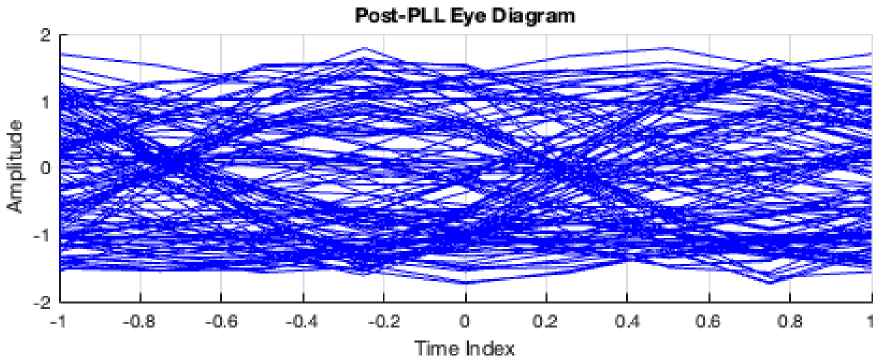


Part J: Post-PLL Output Constellation & Eye Diagram

```
figure;
subplot(2,1,1);
plot(despunSignal(2000:4:3000), 'r.'); grid on; axis equal;
axis([-2 2 -2 2]);
title('Post-PLL Constellation');

subplot(2,1,2); hold on;
for n = 2001:8:3000
    plot(linspace(-1,1,9), real(despunSignal(n:n+8)), 'b');
end
```

```
hold off; grid on;
title('Post-PLL Eye Diagram');
xlabel('Time Index'); ylabel('Amplitude');
```

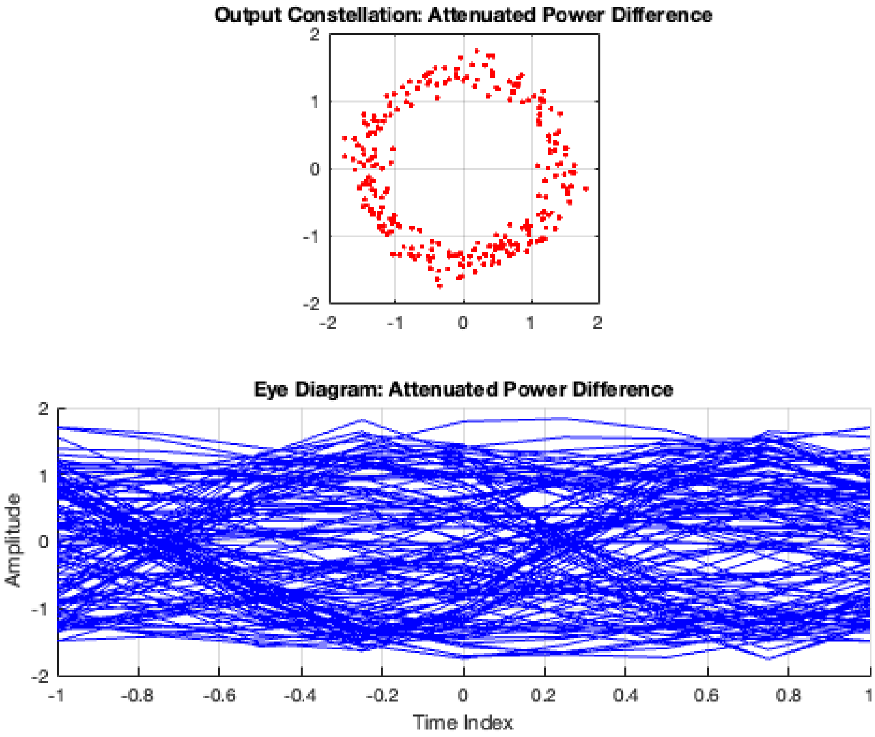




Part K: Attenuated PLL Loop (0.1 Power-Difference)

Pre-allocate

```
y2_k        = zeros(1, loopLen);
yy_k        = zeros(1, loopLen);
pllReg_k    = zeros(1, numTaps);
accum_k     = 0;                  % phase accumulator
intState_k  = 0;                  % loop filter integrator
accumLog_k  = zeros(1, loopLen);
lpIn_k      = 0;                  % leaky integrator input
lpInLog_k   = zeros(1, loopLen);
lpOutLog_k  = zeros(1, loopLen);
alphaLeak_k = 0.95;               % leaky integrator feedback

for nn = 1:(loopLen - numTaps)
    % 1) Despin by current phase estimate
    y2_k(nn) = receivedSignal(nn) * exp(-1j*2*pi*accum_k);

    % 2) Shift-register for BE filters & matched filter
    pllReg_k = [y2_k(nn), pllReg_k(1:end-1)];
    yy_k(nn) = pllReg_k * rxMatchedFilter';  % matched-filter output

    % 3) Band-edge filter outputs
    yBEp = pllReg_k * positiveFreqFilter.';
    yBEn = pllReg_k * negativeFreqFilter.';

    % 4) Scaled power difference (0.1 ×)
    diffPow_k = 0.1*(abs(yBEp)^2 - abs(yBEn)^2);
    % 5) Loop filter: leaky integrator + PI
    lpIn_k     = alphaLeak_k*lpIn_k + (1-alphaLeak_k)*diffPow_k;
    intState_k = intState_k + ki * lpIn_k;
    lpOut_k    = intState_k + kp * lpIn_k;

    % 6) Log everything
    lpInLog_k(nn)    = lpIn_k;
    lpOutLog_k(nn)   = lpOut_k;
    accumLog_k(nn)   = accum_k;

    % 7) Update phase accumulator
    accum_k = accum_k + lpOut_k;
end

% === Plot Attenuated PLL Results ===
figure;
subplot(2,1,1);
plot(y2_k(2000:4:3000), 'r.'); grid on; axis equal; axis([-2 2 -2 2]);
title('Output Constellation: Attenuated Power Difference');

subplot(2,1,2);
hold on;
for n = 2001:8:3000
    plot(-1:1/4:1, real(y2_k(n:n+8)), 'b');
end
hold off; grid on;
title('Eye Diagram: Attenuated Power Difference');
xlabel('Time Index'); ylabel('Amplitude');
```



=== Helper Function ===

```
function h = sqrtNyquistFilter(symbolRate, sampRate, alpha, delaySym)
    t = -delaySym*(1/symbolRate) : 1/sampRate : delaySym*(1/symbolRate);
    h = zeros(size(t));
    for i = 1:length(t)
```

```
        ti = t(i);
        if ti == 0
            h(i) = 1 - alpha + 4*alpha/pi;
        elseif abs(ti) == 1/(4*alpha*symbolRate)
            h(i) = (alpha/sqrt(2))*((1+2/pi)*sin(pi/(4*alpha)) + (1-2/pi)*cos(pi/(4*alpha)));
        else
            num = sin(pi*ti*symbolRate*(1-alpha)) + 4*alpha*ti*symbolRate*cos(pi*ti*symbolRate*(1+alpha));
            den = pi*ti*symbolRate*(1 - (4*alpha*ti*symbolRate)^2);
            h(i) = num/den;
        end
    end
    h = h / norm(h);
end
```

Published with MATLAB® R2023b