Listings

1 Write a MATLAB program to verify the Sampling Theorem.

```
function s = my_sinc(x)
    s = sin(pi * x) ./ (pi * x);
    s(x == 0) = 1;
end

fc1 = input('Frequency for signal 1: ');
a1 = input('Amplitude for signal 1: ');
fc2 = input('Frequency for signal 2: ');
a2 = input('Amplitude for signal 2: ');

n = 20;
tc1 = 1 / fc1; tc2 = 1 / fc2;
t = 0:1/(100*max(fc1, fc2)):n*max(tc1, tc2);

figure;
subplot(2, 1, 1);
xf1 = a1 * cos(2 * pi * fc1 * t);
plot(t, xf1, 'LineWidth', 1);
title(sprintf('Signal 1: f = %.2f Hz, A = %.2f', fc1, a1));
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(2, 1, 2);
xf2 = a2 * cos(2 * pi * fc2 * t);
plot(t, xf2, 'LineWidth', 1);
title(sprintf('Signal 2: f = %.2f Hz, A = %.2f', fc2, a2));
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

x_sum = a1 * cos(2 * pi * fc1 * t) + a2 * cos(2 * pi * fc2 * t);
f_max = max(fc1, fc2);
t_nq = 1 / (f_max * 2);
sampleTime = 0:t_nq:n*max(tc1, tc2);
sampleSignal = a1 * cos(2 * pi * fc1 * sampleTime) + a2 * cos(2 * pi * fc2 * sampleTime)
    ;

figure;
subplot(3, 1, 1);
plot(t, x_sum, 'LineWidth', 1);
title('Summation of Signal 1 and Signal 2');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(3, 1, 2);
stem(sampleTime, sampleSignal, 'r.'); hold on;
plot(t, x_sum, 'b--', 'LineWidth', 0.5);
title('Sampling of Summed Signal');
xlabel('Time (s)');
ylabel('Amplitude');
legend('Sampled Signal', 'Original Signal');
grid on;

subplot(3, 1, 3);
stem(sampleTime, sampleSignal, 'LineWidth', 1);
title('Sampled Signal Representation');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

x_sum_f = abs(fft(sampleSignal));

figure;
subplot(2, 1, 1);
plot(sampleTime, x_sum_f, 'LineWidth', 2);
title('Frequency Domain Representation of Sampled Signal');
xlabel('Frequency (Hz)');
```

```
ylabel('Amplitude');
grid on;

t_recon = t; x_recon = zeros(size(t_recon));
for i = 1:length(t_recon)
    for j = 1:length(sampleTime)
        x_recon(i) = x_recon(i) + sampleSignal(j) * my_sinc((t_recon(i) - sampleTime(j))
     / t_nq);
    end
end

subplot(2, 1, 2);
plot(t_recon, x_recon, 'r-', 'LineWidth', 1.5); hold on;
plot(t, x_sum, 'b--', 'LineWidth', 0.5);
title('Reconstructed Signal from Sampled Data');
xlabel('Time (s)');
ylabel('Amplitude');
legend('Reconstructed Signal', 'Original Signal');
grid on;
```

Program 1: MATLAB program to verify the Sampling Theorem

2 MATLAB program to compute the linear convolution of two discrete sequences.

```
function y = linearConv(x, h)
    l = length(x) + length(h) - 1;
    y = zeros(1, l);
    for n = 1:l
        for k = 1:length(x)
            hIndex = n - k + 1;
            if hIndex > 0 && hIndex <= length(h)
                y(n) = y(n) + x(k) * h(hIndex);
            end
        end
    end
end

x = str2num(input('Enter the value of x: ', 's'));
h = str2num(input('Enter the value of h: ', 's'));

disp('Custom Convolution:');
y_custom = linearConv(x, h)

disp('Built-in Convolution:');
y_builtin = conv(x, h)

figure;
subplot(3, 1, 1);
stem(x, 'filled'); title("Input Signal X[n]");
xlabel("n"); ylabel("Amplitude"); xticks(0:length(x)+1);

subplot(3, 1, 2);
stem(h, 'filled'); title("Impulse Response h[n]");
xlabel("n"); ylabel("Amplitude"); xticks(0:length(h)+1);

subplot(3, 1, 3);
stem(y_custom, 'filled'); title("Output Signal Y[n] (Convolution Result)");
xlabel("n"); ylabel("Amplitude"); xticks(0:length(y_custom)+1);
```

Program 2: MATLAB program to compute the linear convolution of two discrete sequences

3 MATLAB program to compute the circular convolution of two discrete sequences.

```
function y = circularConvolution(x, h)
    N = max(length(x), length(h));

    x = [x, zeros(1, N-length(x))];
    h = [h, zeros(1, N-length(h))];
    y = zeros(1, N);

    for n = 1:N
        for k = 1:N
            y(n) = y(n) + x(k) * h(mod(n-k, N) + 1);
        end
    end
end

fprintf("Enter the values of x:");
x = str2num(input('', 's'));

fprintf("Enter the values of h:");
h = str2num(input('', 's'));

fprintf("Circular convolution of x and h manually");
y=circularConvolution(x, h)

fprintf("Circular convolution of x and h (built in function)");
y=cconv(x,h, max(length(x), length(y)))
```

Program 3: MATLAB program to compute the circular convolution of two discrete sequences

4 MATLAB program to verify the commutative property of convolution

```
function y = linearConv(x, h)
        N = length(x);
        M = length(h);

        l = N+M-1;

        y = zeros(1, l);

        for n=1:l
            for k=1:N
                hIndex = n-k+1;
                if(hIndex > 0 && hIndex <= M)
                    y(n) = y(n) + x(k)*h(hIndex);
                end
            end
        end
end

function checkCommutativeProperty(x, h)
    fprintf("By Calculating customly: x*h ");
    y1 = linearConv(x, h)
    fprintf("By Calculating customly: h*x ");
    y2 = linearConv(h, x)

    if y1 == y2
        fprintf("Commutative  property is proved");
    else
        fprintf("Commutative  is not proved");
    end
end

fprintf("Enter the values of x:");
x = str2num(input('', 's'));
```

```

fprintf("Enter the values of h:");
h = str2num(input('', 's'));

checkCommutativeProperty(x, h)
```

Program 4: MATLAB program to verify the commutative property of convolution

5 MATLAB program to verify the distributive property of convolution.

```
function y = linearConv(x, h)
        N = length(x);
        M = length(h);

        l = N+M-1;

        y = zeros(1, l);

        for n=1:l
            for k=1:N
                hIndex = n-k+1;
                if(hIndex > 0 && hIndex <= M)
                    y(n) = y(n) + x(k)*h(hIndex);
                end
            end
        end
end

function checkDistributiveProperty(x, h1, h2)
    mx = max(length(h1), length(h2))
    h1(length(h1)+1:mx)=0
    h2(length(h2)+1:mx)=0

    fprintf("By Calculating customly: x*(h1+h2) ");
    y1 = linearConv(x, h1+h2)
    fprintf("By Calculating customly: x*h1+x*h2 ");
    y21 = linearConv(x, h1);
    y22 = linearConv(x, h2);
    y2 = y21 + y22

    if y1 == y2
        fprintf("Distributive  property is proved");
    else
        fprintf("Distributive is not proved");
    end
end

fprintf("Enter the values of x:");
x = str2num(input('', 's'));

fprintf("Enter the values of h1:");
h1 = str2num(input('', 's'));

fprintf("Enter the values of h2:");
h2 = str2num(input('', 's'));

checkDistributiveProperty(x, h1, h2)
```

Program 5: MATLAB program to verify the distributive property of convolution

6 MATLAB program to verify the associative property of convolution.

```
function y = linearConv(x, h)
        N = length(x);
        M = length(h);

        l = N+M-1;

        y = zeros(1, l);

        for n=1:l
            for k=1:N
                hIndex = n-k+1;
                if(hIndex > 0 && hIndex <= M)
                    y(n) = y(n) + x(k)*h(hIndex);
                end
            end
        end
end

function checkAssociativity(x, h1, h2)
    fprintf("By Calculating customly: (x*h1)*h2 ");
    y1 = linearConv( linearConv(x, h1) , h2)
    fprintf("By Calculating customly: (x)*(h1*h2) ");
    y2 = linearConv(x, linearConv(h1, h2))

    if y1 == y2
        fprintf("Associativity is proved");
    else
        fprintf("Associavitiy is not proved");
    end
end

fprintf("Enter the values of x:");
x = str2num(input('', 's'));

fprintf("Enter the values of h1:");
h1 = str2num(input('', 's'));

fprintf("Enter the values of h2:");
h2 = str2num(input('', 's'));

checkAssociativity(x, h1, h2)
```

Program 6: MATLAB program to verify the associative property of convolution