

Listings

1	MATLAB program to verify the Sampling Theorem	1
2	MATLAB program to compute the linear convolution of two discrete sequences .	2
3	MATLAB program to compute the circular convolution of two discrete sequences	3
4	MATLAB program to verify the commutative property of convolution for two sequences A and B	3
5	MATLAB program to verify the associative property of convolution	4

1 Write a MATLAB program to verify the Sampling Theorem.

```
1 function s = my_sinc(x)
2     s = sin(pi * x) ./ (pi * x);
3     s(x == 0) = 1;
4 end
5
6 fc1 = input('Frequency for signal 1: ');
7 a1 = input('Amplitude for signal 1: ');
8 fc2 = input('Frequency for signal 2: ');
9 a2 = input('Amplitude for signal 2: ');
10
11 n = 20;
12 tc1 = 1 / fc1; tc2 = 1 / fc2;
13 t = 0:1/(100*max(fc1, fc2)):n*max(tc1, tc2);
14
15 figure;
16 subplot(2, 1, 1);
17 xf1 = a1 * cos(2 * pi * fc1 * t);
18 plot(t, xf1, 'LineWidth', 1);
19 title(sprintf('Signal 1: f = %.2f Hz, A = %.2f', fc1, a1));
20 xlabel('Time (s)');
21 ylabel('Amplitude');
22 grid on;
23
24 subplot(2, 1, 2);
25 xf2 = a2 * cos(2 * pi * fc2 * t);
26 plot(t, xf2, 'LineWidth', 1);
27 title(sprintf('Signal 2: f = %.2f Hz, A = %.2f', fc2, a2));
28 xlabel('Time (s)');
29 ylabel('Amplitude');
30 grid on;
31
32 x_sum = a1 * cos(2 * pi * fc1 * t) + a2 * cos(2 * pi * fc2 * t);
33 f_max = max(fc1, fc2);
34 t_nq = 1 / (f_max * 2);
35 sampleTime = 0:t_nq:n*max(tc1, tc2);
36 sampleSignal = a1 * cos(2 * pi * fc1 * sampleTime) + a2 * cos(2 * pi * fc2 *
    sampleTime);
37
38 figure;
39 subplot(3, 1, 1);
40 plot(t, x_sum, 'LineWidth', 1);
41 title('Summation of Signal 1 and Signal 2');
42 xlabel('Time (s)');
43 ylabel('Amplitude');
44 grid on;
45
46 subplot(3, 1, 2);
47 stem(sampleTime, sampleSignal, 'r.');
```

hold on;

```
48 plot(t, x_sum, 'b--', 'LineWidth', 0.5);
49 title('Sampling of Summed Signal');
50 xlabel('Time (s)');
51 ylabel('Amplitude');
52 legend('Sampled Signal', 'Original Signal');
53 grid on;
54
55 subplot(3, 1, 3);
56 stem(sampleTime, sampleSignal, 'LineWidth', 1);
57 title('Sampled Signal Representation');
58 xlabel('Time (s)');
59 ylabel('Amplitude');
60 grid on;
61
62 x_sum_f = abs(fft(sampleSignal));
63
64 figure;
65 subplot(2, 1, 1);
66 plot(sampleTime, x_sum_f, 'LineWidth', 2);
```

```

67 title('Frequency Domain Representation of Sampled Signal');
68 xlabel('Frequency (Hz)');
69 ylabel('Amplitude');
70 grid on;
71
72 t_recon = t; x_recon = zeros(size(t_recon));
73 for i = 1:length(t_recon)
74     for j = 1:length(sampleTime)
75         x_recon(i) = x_recon(i) + sampleSignal(j) * my_sinc((t_recon(i) -
76             sampleTime(j)) / t_nq);
77     end
78 end
79 subplot(2, 1, 2);
80 plot(t_recon, x_recon, 'r-', 'LineWidth', 1.5); hold on;
81 plot(t, x_sum, 'b--', 'LineWidth', 0.5);
82 title('Reconstructed Signal from Sampled Data');
83 xlabel('Time (s)');
84 ylabel('Amplitude');
85 legend('Reconstructed Signal', 'Original Signal');
86 grid on;

```

Program 1: MATLAB program to verify the Sampling Theorem

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

```

1 function y = linearConv(x, h)
2     l = length(x) + length(h) - 1;
3     y = zeros(1, l);
4     for n = 1:l
5         for k = 1:length(x)
6             hIndex = n - k + 1;
7             if hIndex > 0 && hIndex <= length(h)
8                 y(n) = y(n) + x(k) * h(hIndex);
9             end
10        end
11    end
12 end
13
14 x = str2num(input('Enter the value of x: ', 's'));
15 h = str2num(input('Enter the value of h: ', 's'));
16
17 disp('Custom Convolution:');
18 y_custom = linearConv(x, h)
19
20 disp('Built-in Convolution:');
21 y_builtin = conv(x, h)
22
23 figure;
24 subplot(3, 1, 1);
25 stem(x, 'filled'); title("Input Signal X[n]");
26 xlabel("n"); ylabel("Amplitude"); xticks(0:length(x)+1);
27
28 subplot(3, 1, 2);
29 stem(h, 'filled'); title("Impulse Response h[n]");
30 xlabel("n"); ylabel("Amplitude"); xticks(0:length(h)+1);
31
32 subplot(3, 1, 3);
33 stem(y_custom, 'filled'); title("Output Signal Y[n] (Convolution Result)");
34 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 A and B.

```
1 function y = linearConv(x, h)
2     N = length(x);
3     M = length(h);
4     l = N + M - 1;
5     y = zeros(1, l);
6     for n = 1:l
7         for k = 1:N
8             hIndex = n - k + 1;
9             if hIndex > 0 && hIndex <= M
10                 y(n) = y(n) + x(k) * h(hIndex);
11             end
12         end
13     end
14 end
15
16 function checkDistributiveProperty(x, h)
17     fprintf("By Calculating customly: x*h ");
18     y1 = linearConv(x, h)
19     fprintf("By Calculating customly: h*x ");
20     y2 = linearConv(h, x)
21     if y1 == y2
22         fprintf("Commutative property is proved");
23     else
24         fprintf("Commutative is not proved");
25     end
26 end
27
28 fprintf("Enter the values of x:");
29 x = str2num(input(' ', 's'));
30 fprintf("Enter the values of h:");
31 h = str2num(input(' ', 's'));
32 checkDistributiveProperty(x, h)
```

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

4 MATLAB program to verify the commutative property of convolution for two sequences A and B.

```
1 function y = circularConvolution(x, h)
2     N = max(length(x), length(h));
3
4     x = [x, zeros(1, N-length(x))];
5     h = [h, zeros(1, N-length(h))];
6     y = zeros(1, N);
7
8     for n = 1:N
9         for k = 1:N
10             y(n) = y(n) + x(k) * h(mod(n-k, N) + 1);
11         end
12     end
13 end
14
15 fprintf("Enter the values of x:");
16 x = str2num(input(' ', 's'));
17
18 fprintf("Enter the values of h:");
19 h = str2num(input(' ', 's'));
20
21 fprintf("Circular convolution of x and h manually");
22 y=circularConvolution(x, h)
23
24 fprintf("Circular convolution of x and h (built in function)");
```

```
25 y=cconv(x,h, max(length(x), length(y)))
```

Program 4: MATLAB program to verify the commutative property of convolution for two sequences A and B

5 MATLAB program to verify the distributive property of convolution.

```
1 function checkDistributiveProperty(x, h1, h2)
2     mx = max(length(h1), length(h2))
3     h1(length(h1)+1:mx)=0
4     h2(length(h2)+1:mx)=0
5
6     fprintf("By Calculating customly: x*(h1+h2) ");
7     y1 = conv(x, h1+h2)
8     fprintf("By Calculating customly: x*h1+x*h2 ");
9     y21 = conv(x, h1);
10    y22 = conv(x, h2);
11    y2 = y21 + y22
12
13    if y1 == y2
14        fprintf("Distributive property is proved");
15    else
16        fprintf("Distributive is not proved");
17    end
18 end
19
20 fprintf("Enter the values of x:");
21 x = str2num(input(' ', 's'));
22
23 fprintf("Enter the values of h1:");
24 h1 = str2num(input(' ', 's'));
25
26 fprintf("Enter the values of h2:");
27 h2 = str2num(input(' ', 's'));
28
29 checkDistributiveProperty(x, h1, h2)
```

6 MATLAB program to verify the associative property of convolution.

```
1 function checkAssociativity(x, h1, h2)
2     fprintf("By Calculating: (x*h1)*h2 ");
3     y1 = conv(conv(x, h1), h2)
4     fprintf("By Calculating: (x)*(h1*h2) ");
5     y2 = conv(x, conv(h1, h2))
6
7     if y1 == y2
8         fprintf("Associativity is proved");
9     else
10        fprintf("Associativity is not proved");
11    end
12 end
13
14 fprintf("Enter the values of x:");
15 x = str2num(input(' ', 's'));
16
17 fprintf("Enter the values of h1:");
18 h1 = str2num(input(' ', 's'));
19
20 fprintf("Enter the values of h2:");
21 h2 = str2num(input(' ', 's'));
22
23 checkAssociativity(x, h1, h2)
```

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