CPE 3500

Embedded Digital Signal Processing

Lab 3: Discrete-Time Systems & Convolution

Anindita Deb

09/24/2024

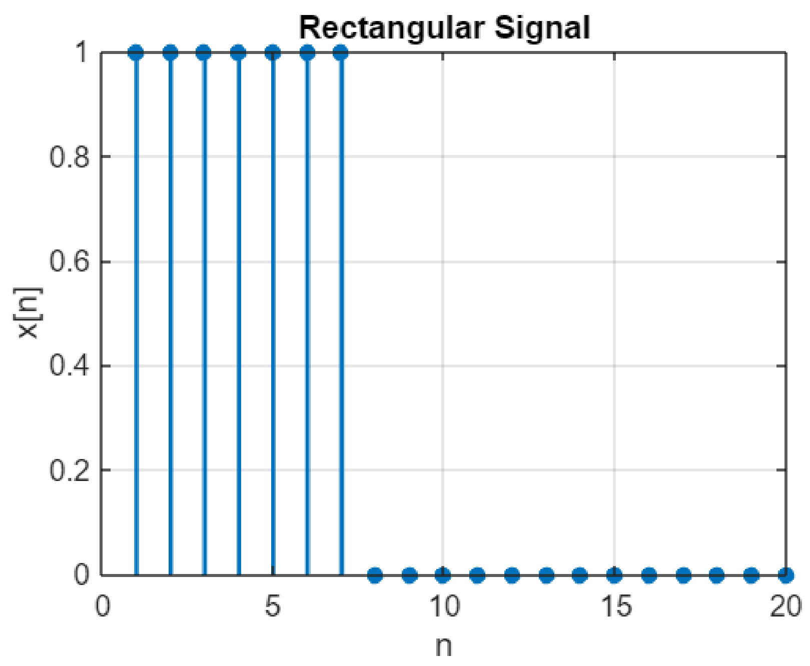
Task-1:

Place the system1 function in Private user code section inside the main.c file.

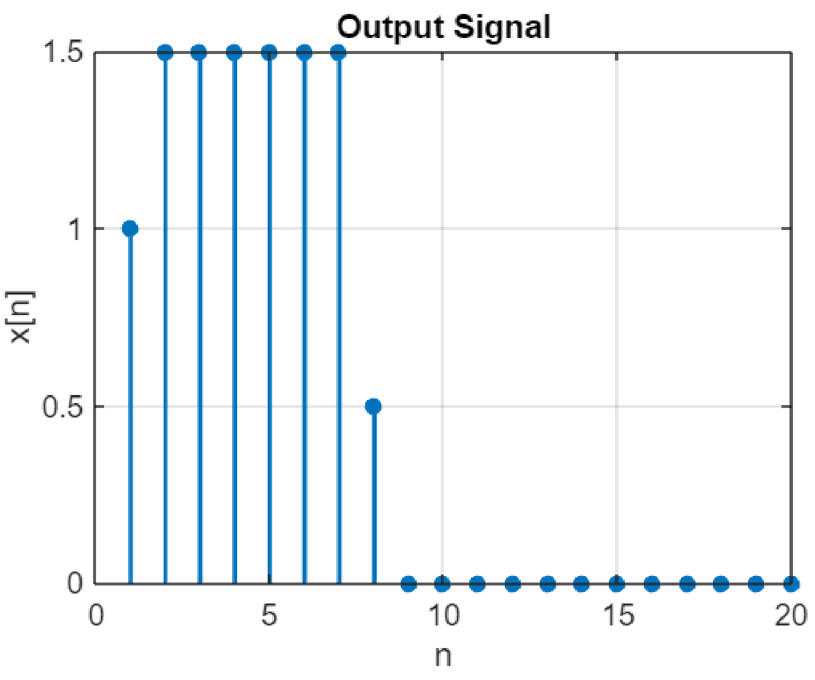
/* Private user code ---*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */

In the main function, define a DT rectangular signal with first 7 elements to be 1. Call system1 with rectangular signal as input and calculate the output signal while b=0.5. Export both input and output signals of the system and plot them in Matlab.

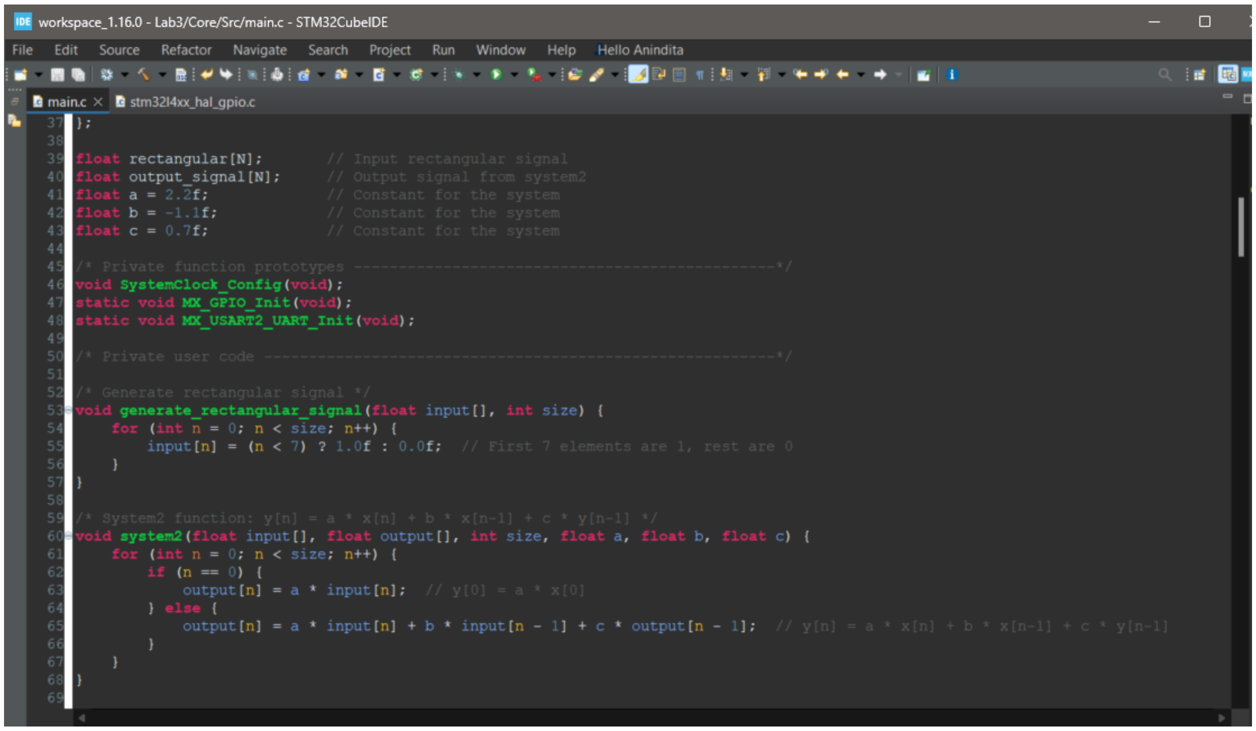
```matlab
fid = fopen('lab3rect1.bin'); % file should be in the working directory
data = fread(fid,inf,'float'); % put 'float' instead of int if your data is float type.
fclose(fid);

figure;
stem(data,'filled','LineWidth',2);
set(gca,'FontSize',14)
grid on;
xlabel('n','FontSize',16);
ylabel('x[n]','FontSize',16);
title("Rectangular Signal");  % This is the title of the plot (modify for each different plot)
```



```
fid = fopen('lab3rect2.bin'); % file should be in the working directory
data = fread(fid,inf,'float'); % put 'float' instead of int if your data is float type.
fclose(fid);

figure;
stem(data,'filled','LineWidth',2);
set(gca,'FontSize',14)
grid on;
xlabel('n','FontSize',16);
ylabel('x[n]','FontSize',16);
title("Output Signal");  % This is the title of the plot (modify for each different plot)
```

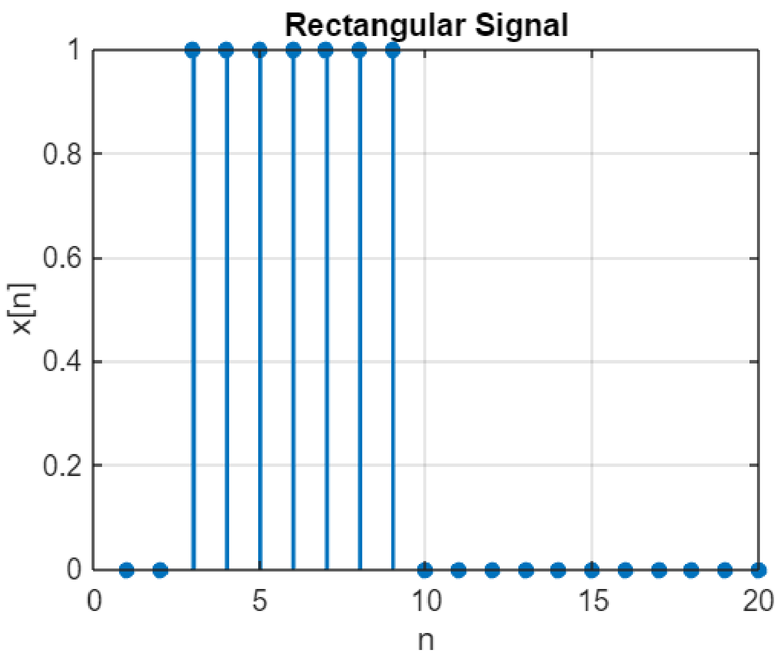


Task-2:
Based on system1, create another function which calculates the system;
y[n]=a*x[n]+b*x[n-1]+c*y[n-1] where a=2.2, b=-1.1 and c=0.7.
Call this function and calculate the output for the same DT signal as input. Export the output signal and plot in Matlab.

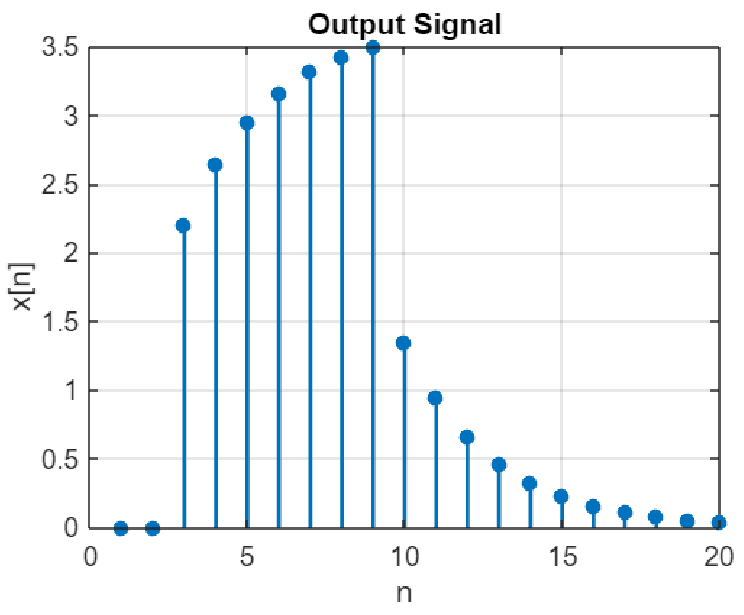

workspace_1.16.0 - Lab3/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help Hello Anindita
main.c
stm32l4xx_hal_gpio.c

```c
};

float rectangular[N];       // Input rectangular signal
float output_signal[N];     // Output signal from system2
float a = 2.2f;             // Constant for the system
float b = -1.1f;            // Constant for the system
float c = 0.7f;             // Constant for the system

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);

/* Private user code ---------------------------------------------------------*/

/* Generate rectangular signal */
void generate_rectangular_signal(float input[], int size) {
    for (int n = 0; n < size; n++) {
        input[n] = (n < 7) ? 1.0f : 0.0f;  // First 7 elements are 1, rest are 0
    }
}

/* System2 function: y[n] = a * x[n] + b * x[n-1] + c * y[n-1] */
void system2(float input[], float output[], int size, float a, float b, float c) {
    for (int n = 0; n < size; n++) {
        if (n == 0) {
            output[n] = a * input[n];  // y[0] = a * x[0]
        } else {
            output[n] = a * input[n] + b * input[n - 1] + c * output[n - 1];  // y[n] = a * x[n] + b * x[n-1] + c * y[n-1]
        }
    }
}

```

```
fid = fopen('lab3rect3.bin'); % file should be in the working directory
data = fread(fid,inf,'float'); % put 'float' instead of int if your data is float type.
fclose(fid);

figure;
stem(data,'filled','LineWidth',2);
set(gca,'FontSize',14)
grid on;
xlabel('n','FontSize',16);
ylabel('x[n]','FontSize',16);
title("Rectangular Signal");  % This is the title of the plot (modify for each different plo
```

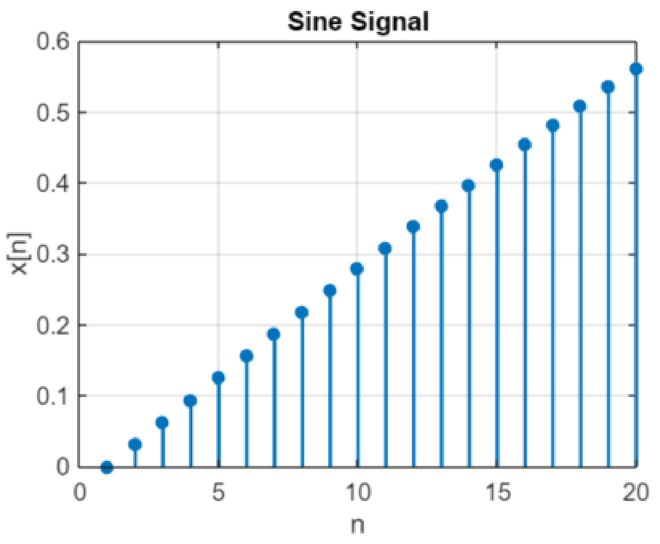


```
fid = fopen('lab3rect4.bin'); % file should be in the working directory
data = fread(fid,inf,'float'); % put 'float' instead of int if your data is float type.
fclose(fid);

figure;
stem(data,'filled','LineWidth',2);
set(gca,'FontSize',14)
grid on;
xlabel('n','FontSize',16);
ylabel('x[n]','FontSize',16);
title("Output Signal");  % This is the title of the plot (modify for each different plot)
```

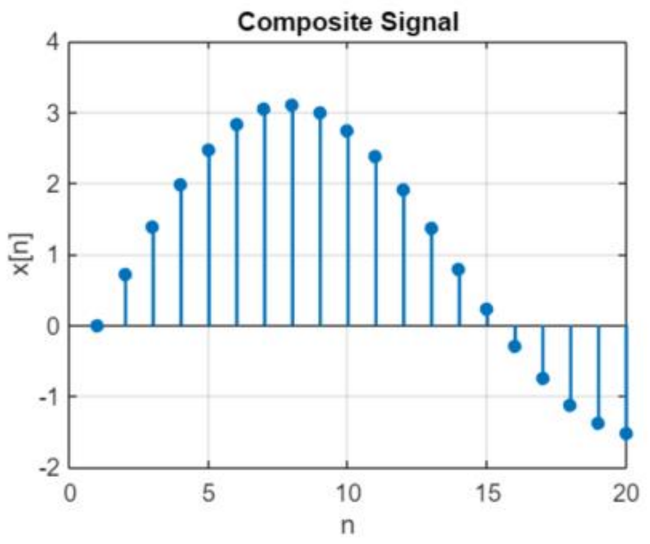


Task-3:
Use the sampleMain.c code provided on D2L to generate one single tone sine signal at 5 Hz and one composite sinusoidal signal combining 5 Hz sine with 3 more different frequency sine signals. For this, you can pick any frequency above 25 Hz and below 50 Hz.
Place the sections of the code into appropriate places in the main.c file of your project. A DT low pass filter is defined as a system in the code which you can calculate the convolution sum as the output signal (sineFiltered).
After running the code in debug mode, export single tone sine, composite sinusoidal signal and filtered signals. Plot these signals in a single figure in Matlab. Compare and comment on the results.
Note: Filtered signal may generate slightly higher amplitudes compared the single tone sine. It will be better to normalize this signal by dividing the array elements to the maximum value of the array in Matlab (i.e. sineFiltered = sineFiltered/max(sineFiltered)).

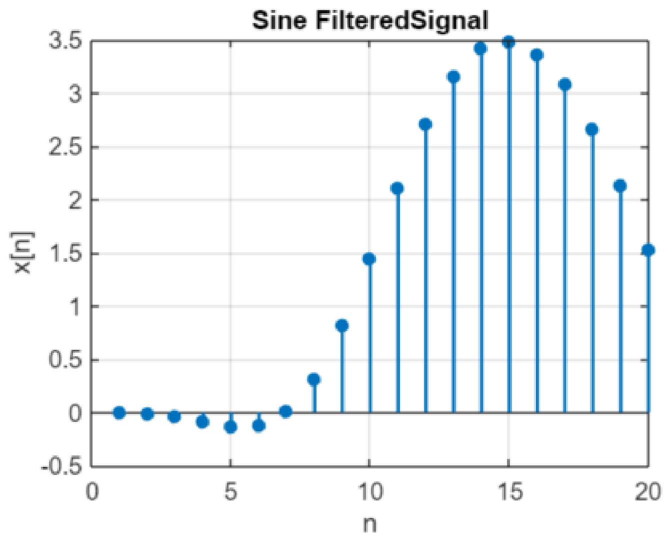
```
fid = fopen('sineSignal.bin'); % file should be in the working directory
data = fread(fid,inf,'float'); % put 'float' instead of int if your data is float type.
fclose(fid);

figure;
stem(data,'filled','LineWidth',2);
set(gca,'FontSize',14)
grid on;
xlabel('n','FontSize',16);
ylabel('x[n]','FontSize',16);
title("Sine Signal");  % This is the title of the plot (modify for each different plot)
```

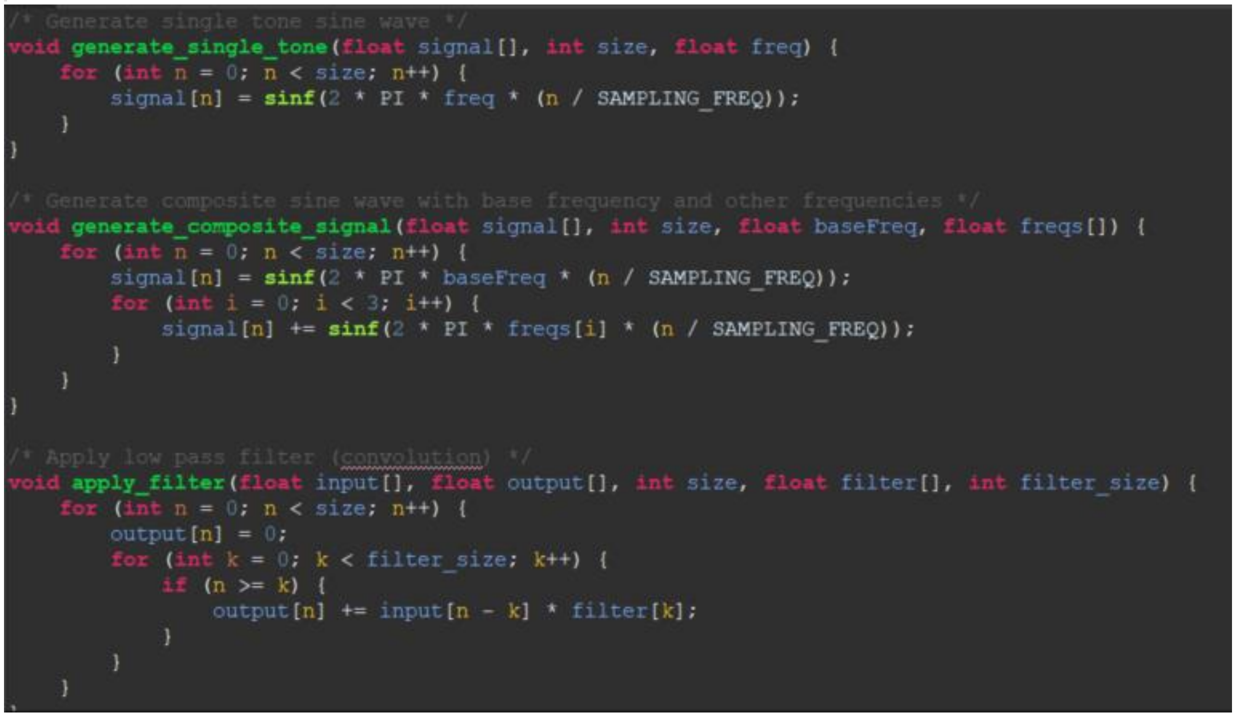


```
fid = fopen('compositeSignal.bin'); % file should be in the working directory
data = fread(fid,inf,'float'); % put 'float' instead of int if your data is float type.
fclose(fid);

figure;
stem(data,'filled','LineWidth',2);
set(gca,'FontSize',14)
grid on;
xlabel('n','FontSize',16);
ylabel('x[n]','FontSize',16);
title("Composite Signal");  % This is the title of the plot (modify for each different plot)
```



```
fid = fopen('sineFiltered.bin'); % file should be in the working directory
data = fread(fid,inf,'float'); % put 'float' instead of int if your data is float type.
fclose(fid);

figure;
stem(data,'filled','LineWidth',2);
set(gca,'FontSize',14)
grid on;
xlabel('n','FontSize',16);
ylabel('x[n]','FontSize',16);
title("Sine FilteredSignal");  % This is the title of the plot (modify for each different plot)
```





```
/* Generate single tone sine wave */
void generate_single_tone(float signal[], int size, float freq) {
    for (int n = 0; n < size; n++) {
        signal[n] = sinf(2 * PI * freq * (n / SAMPLING_FREQ));
    }
}

/* Generate composite sine wave with base frequency and other frequencies */
void generate_composite_signal(float signal[], int size, float baseFreq, float freqs[]) {
    for (int n = 0; n < size; n++) {
        signal[n] = sinf(2 * PI * baseFreq * (n / SAMPLING_FREQ));
        for (int i = 0; i < 3; i++) {
            signal[n] += sinf(2 * PI * freqs[i] * (n / SAMPLING_FREQ));
        }
    }
}

/* Apply low pass filter (convolution) */
void apply_filter(float input[], float output[], int size, float filter[], int filter_size) {
    for (int n = 0; n < size; n++) {
        output[n] = 0;
        for (int k = 0; k < filter_size; k++) {
            if (n >= k) {
                output[n] += input[n - k] * filter[k];
            }
        }
    }
```

Taking the comparison of the three individual plots you can see the filtered sine plot is a combination of the previous two plots. It’s essentially the essence of convolution as one low value cancels out the other while large values make even larger ones. The skew in the plot displays that very well.