ELECENG 3TR4 Lab 1: Fourier Analysis

Aaron Pinto
pintoa9

Raeed Hassan
hassam41

February 5, 2021

Design Problem

The output of a function generator is a square wave with 50% duty cycle and the period of the wave is 0.1 ms. You may assume that the amplitude of the square wave is 1 Volt. Design a second order Butterworth filter so that any harmonics in the output is at least 23 dB below the fundamental sinusoid at 10 kHz, and the loss of the fundamental sinusoid due to filtering should be less than 2 dB. The purpose of this experiment is to generate a high amplitude sinusoidal signal out of a square wave (to have a high quality sinusoid, the amplitudes of harmonics should be sufficiently low). You should choose the cutoff frequency of the filter (i.e. 3 dB bandwidth of the filter) such that the above design constraints are met.

Determining the Frequency Response of the Filter

The first step we took was to implement a second order Butterworth filter in MATLAB and examine its frequency response. To simulate the filter, the transfer function of a second order Butterworth filter was used. The transfer function that was used for the second order Butterworth filter was provided in the textbook.

$$H(s) = \frac{1}{s^2 + 1.414s + 1}$$

The MATLAB script used to simulate the filter, frequencyResponse.m, generated the frequency response of the filter with its transfer function using the freqs function. A portion of the MATLAB code used to generate and plot the frequency response of the filter is shown below in Listing 1.

Listing 1: Generating the Frequency Response of Second Order Butterworth Filter

```
23 range = ceil(third_harm_freq/fc) + 1;
24 w = linspace(-range,range,500);
25 h = freqs(b,a,w); % a for poles, b for zeros, w for normalized
      freq range from -5 to 5 rad/s
26 mag = 20*log10(abs(h)); %% convert magnitude to dB
27 %phase = angle(h);
28 %phasedeg = phase*180/pi;
29
30 %% plot frequency response
31 % normalized frequency response
32 figure(1)
33 subplot(2,1,1)
34 plt1 = plot(w,mag);
35 xlim([w(1) w(length(w))]);
36 grid on
37 title('Normalized Frequency Response (Magnitude)');
38 xlabel('Frequency (rad/s)')
39 ylabel('Magnitude (dB)')
40 xline(1,'k',{'Cutoff frequency'}); % 3dB frequency (at cutoff
      frequency)
```

```
41
42 % frequency response
43 subplot(2,1,2)
44 plt2 = plot(fc*w,mag);
45 xlim([fc*w(1) fc*w(length(w))]);
46 plt2ax = ancestor(plt2(1), 'axes'); plt2ax.XAxis.Exponent = 0;
47 grid on
48 title('Frequency Response (Magnitude)');
49 xlabel('Frequency (Hz)')
50 ylabel('Magnitude (dB)')
51 xline(fc,'b',{'Cutoff frequency'}); % 3dB frequency (at cutoff
      frequency)
52 xline(fund_freq,'r',{'Fundamental frequency'}); % first
      harmonic
53 xline(third_harm_freq,'m',{'3rd harmonic'}); % third harmonic
```

The MATLAB script was used to visualize the normalized frequency response of the filter, and to determine the attenuation provided by the filter at frequencies of interest (the fundamental frequency, and the frequency of the third harmonic). An example of plots generated by script is shown in Figure 1.

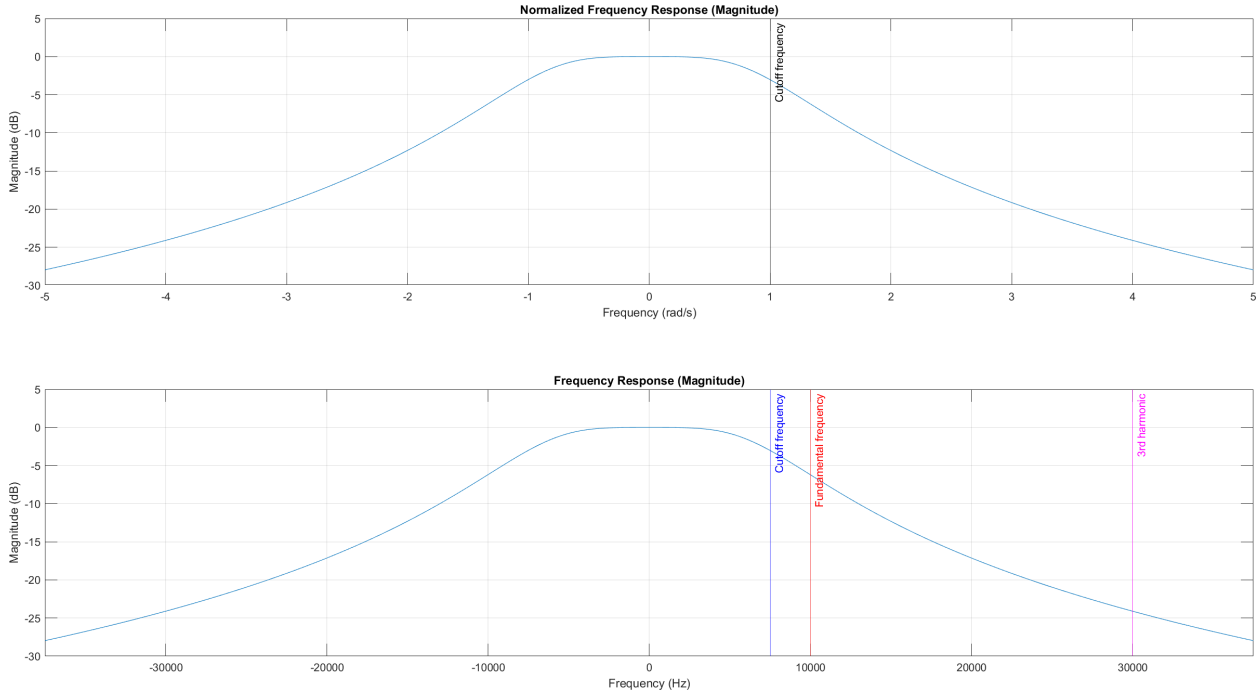

Figure 1: The frequency response plots with the cutoff frequency at 7.5 kHz

Determining the Cutoff Frequency

The MATLAB script was used to determine the appropriate cutoff frequency to meet the design constraints. The first design constraint that we investigated required the loss of the fundamental sinusoid due to filtering to be less than 2 dB. We determined that this occurred when the cutoff frequency was greater than 11430 Hz. We selected a cutoff frequency of 11500 Hz, where the attenuation at the fundamental frequency was approximately 1.96 dB, as our starting point to continue and explore the other design constraints.

The other design constraint that had to be verified was the attenuation of harmonics other than at the fundamental frequency. The design problem requires any harmonics in the output to be at least 23 dB below the fundamental sinusoid. The square wave has natural attenuation that results the third harmonic being 9.54 dB below the first harmonic as calculated: $20\log_{10}\left(\frac{|C_3|}{|C_1|}\right) = 20\log_{10}\left(\frac{0.1061}{0.3183}\right) = -9.54$ dB. Therefore, the filter should provided an additional $23 - 9.54 = 13.46$ dB attenuation. When the cutoff frequency is 11.5 kHz, the attenuation provided by the filter is found to be approximately 14.79 dB, which is greater than the required 13.46 dB. Figure 2 shows the frequency response of the filter at 11.5 kHz. Therefore, we can conclude the minimum value of the cutoff frequency required to meet all design constraints is around 11.5 kHz.

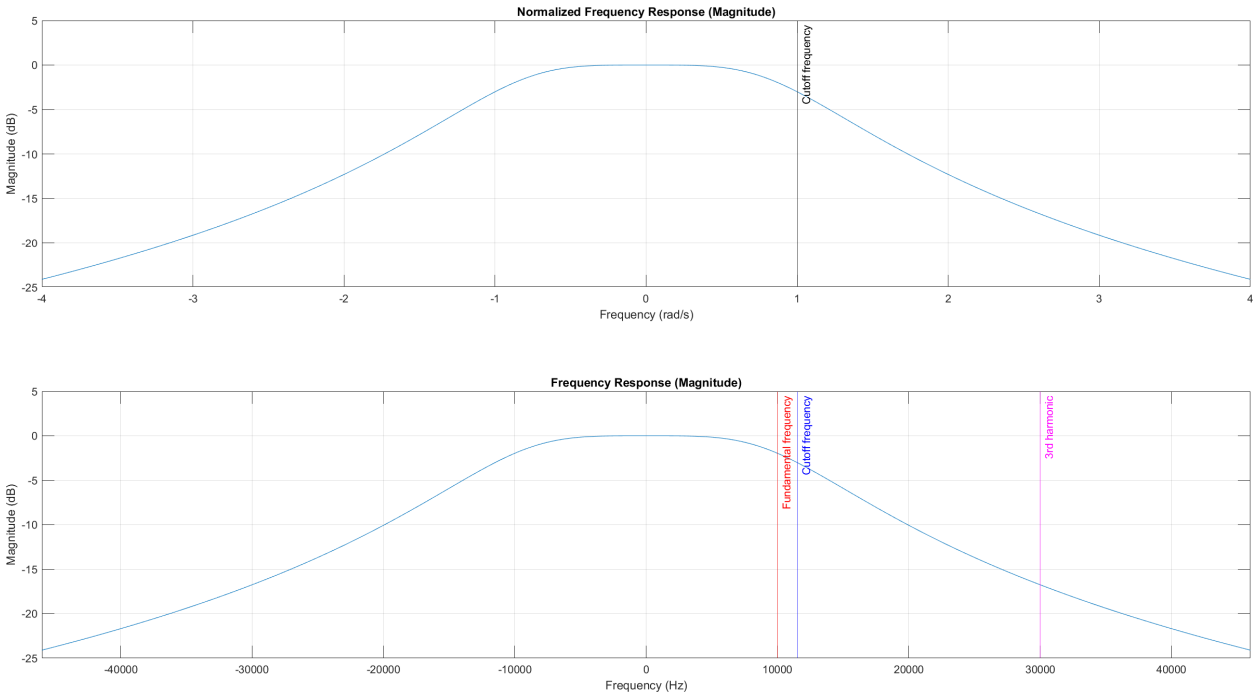

Figure 2: The frequency response plots with the cutoff frequency at 11.5 kHz

Discussing the Impact of Changing Cutoff Frequency

Determining the Range of Acceptable Cutoff Frequencies

We have determined through simulation that the minimum cutoff frequency required to satisfy the design constraints is around 11.5 kHz. We utilize the same approach explored earlier to determine what is the maximum cutoff frequency that the filter can use while still satisfying the design constraints, and determined that the maximum cutoff frequency is around 12.9 kHz. When the cutoff frequency is set to be greater than 12.9 kHz, the filter no longer provides the required attenuation between the fundamental sinusoid and the third harmonic. Figure 3 shows the frequency response of the filter at 12.9 kHz.

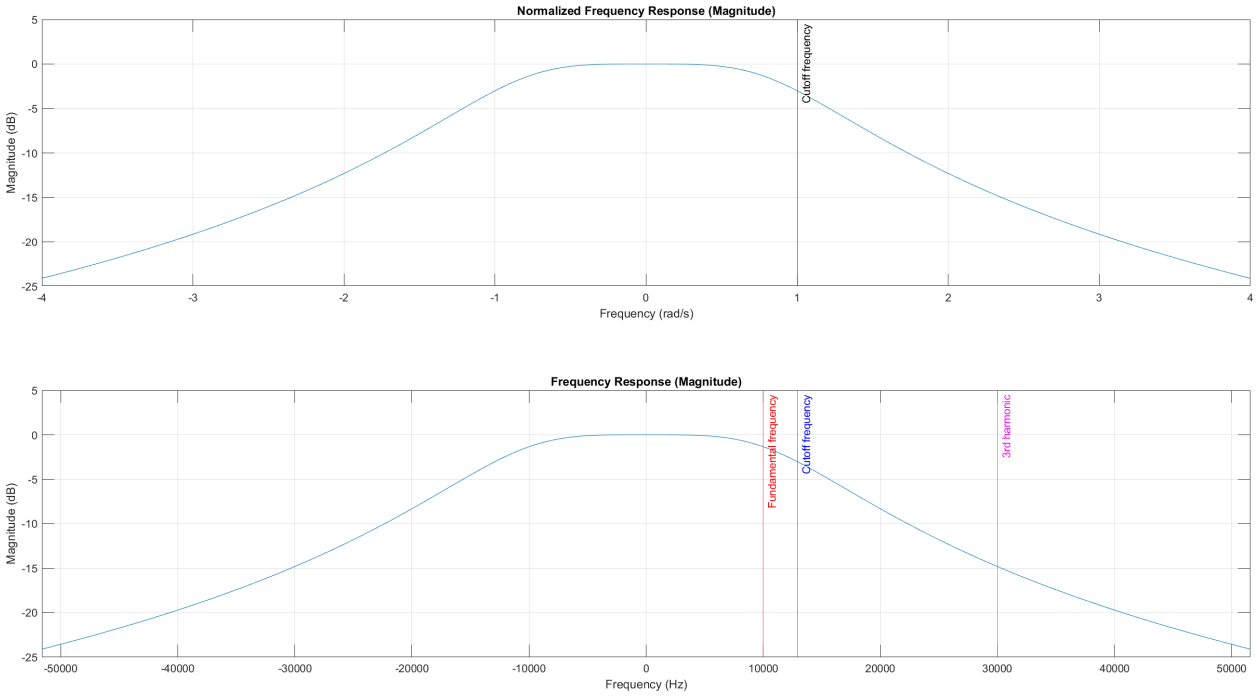

Figure 3: The frequency response plots with the cutoff frequency at 12.9 kHz

We have determined that the acceptable range of cutoff frequencies that satisfy the design constraints of the problem are between 11.5 kHz and 12.9 kHz.

We can confirm these ranges are acceptable analytically through the transfer function of the second order Butterworth filter.

$$H(s) = \frac{1}{s^2 + 1.414s + 1}$$

$$\begin{aligned} 20\log_{10}\left|H\left(\frac{30\text{kHz}}{11.5\text{kHz}}\right)\right| - 20\log_{10}\left|H\left(\frac{10\text{kHz}}{11.5\text{kHz}}\right)\right| &\approx -16.74933 - (-1.96258) \\ &\approx -14.79 \text{ dB} \end{aligned} \tag{1}$$

$$20\log_{10}\left|H\left(\frac{30\text{kHz}}{12.9\text{kHz}}\right)\right| - 20\log_{10}\left|H\left(\frac{10\text{kHz}}{12.9\text{kHz}}\right)\right| \approx -14.80678 - (-1.33777) \approx -13.47 \text{ dB} \quad (2)$$

We can see through Equation 1 and Equation 2 that both cutoff frequencies provide the necessary attenuation from the filter (> 13.46 dB) and also satisfies the condition of not having more than 2 dB of attenuation at the fundamental frequency.

We were able to meet the design constraints with a 2nd order filter, thus if we had to build this filter in hardware, it would be cheaper, smaller and use less resources compared a higher order filter.

Simulation

The simulation of the filter was approached the same way as the triangular_filtering.m MATLAB script provided with the lab. The MATLAB script used to simulate our filter is named lab1.m.

Simulating the Input

The input of the filter is a square wave with with 50% duty cycle, a period of 0.1 ms and an amplitude of 1 V. To generate the square wave in MATLAB, we used the square function from MATLAB's Signal Processing Toolbox. A discrete time square wave (period of 0.0001 s, amplitude of 1 V) was generated which spanned an interval of -0.3 ms and 0.3 ms. The MATLAB code used to generate the square wave is shown in Listing 2.

Listing 2: Generating the Square Wave

```
10 f0 = 10000; % fundamental freq of input square wave
11 T0 = 1/f0;  % period
12 tstep = 0.001*T0;
13 no_sample = 6*T0/tstep + 1;
14 tt = -3*T0:tstep:3*T0;
15
16 input = square(tt*2*pi*f0,50); % input square wave
```

The magnitude and phase spectrums of the input signal was generated in the same manner demonstrated in the triangular filtering script. The Fourier series representation of the square wave was generated using the Fourier series coefficients of a square wave, which has been discussed previously in lecture. The magnitude spectrum plot illustrated the absolute value of the Fourier series representation, while the phase spectrum plot illustrated the angles of the Fourier series representation. The MATLAB code used to generate the magnitude and phase spectrums of the input signal is shown in Listing 3.

Listing 3: Generating the Magnitude and Phase Spectrums of the Input

```
29 N = 100; % number of harmonics
30 nvec = -N:N;
31 c_in = zeros(size(nvec));
32 for n = nvec
33    m = n + N + 1;
34    if (mod(n,2))
35        c_in(m) = sinc(n/2);
36    else
37        c_in(m) = 0.0;
38    end
39 end
40 f = nvec*f0;
41 mag = abs(c_in);
```

```
42 phase = angle(c_in);
```

Figure 4 shows the time and frequency domain plots of the input signal. The magnitude spectrum illustrates the magnitude of the frequency response, with an impulse at every odd harmonic (for each sinusoid that makes up the Fourier series representation of the square wave). The fundamental sinusoid (at frequency 10 kHz) contributes the most to the signal, with each subsequent sinusoid contributing a decreasing amount. These harmonics occur at every other multiple of the fundamental frequency (10 kHz, 30 kHz, 50 kHz, etc.). The phase spectrum illustrates the phase of the frequency response.

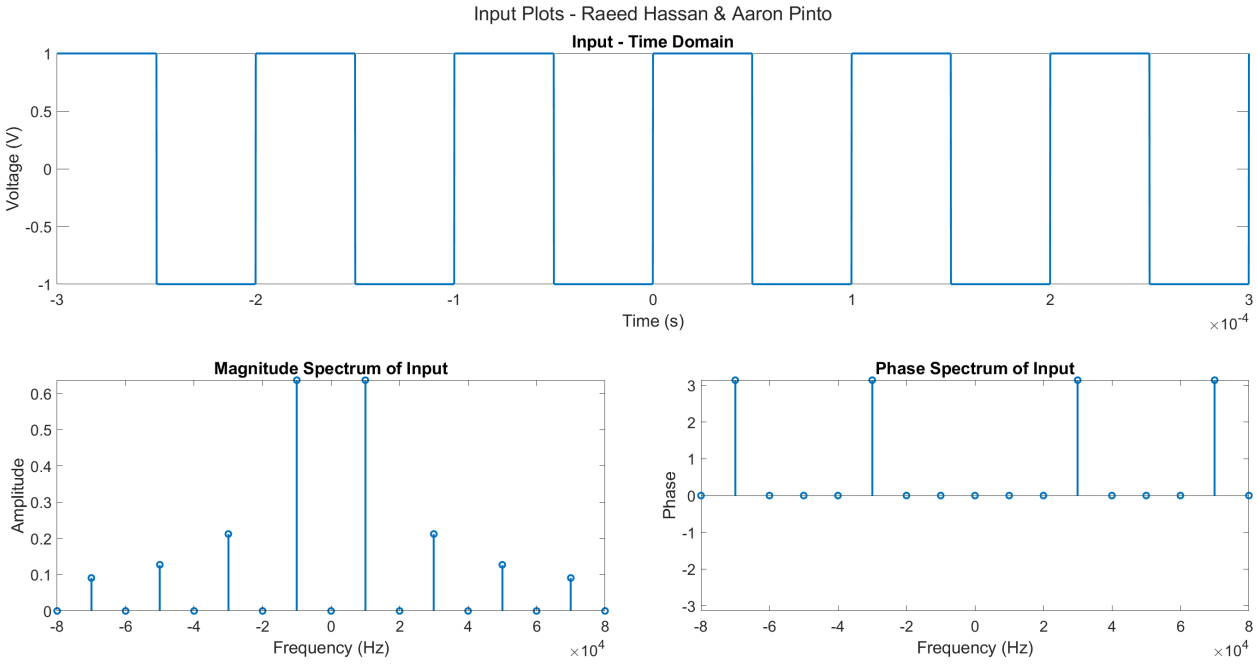

Figure 4: The time and frequency domain plots of the input signal

Simulating the Filter and the Output

To simulate the filter and the output, we used the Fourier series representation method described in the lab document, which was also used in the provided triangular filtering file.

The Fourier series representation of the input signal was multiplied by the transfer function evaluated at the corresponding frequencies. The cutoff frequency used when evaluating the transfer function was the minimum acceptable cutoff frequency that was previously calculated, 11.5 kHz. The magnitude and phase spectrums of the output signal was generated using this new Fourier series representation in the same manner described for the input signal. The weighted sum of the the Fourier series is calculated to generate the output signal in the time domain. The MATLAB code used to simulate the filter and the filtered output is shown in Listing 4

Listing 4: Simulating the filter

```
%% Designing the 2nd order Butterworth filter parameters
fc = 11500; % set your cutoff frequency

Hf = 1 ./ (1 + 1.414*(1i*f/fc) + (1i*f/fc).^2); % transfer
   function of filter
c_out = c_in .* Hf; % Fourier coefficients of the filter output

%% Construct the output signal from the Cout Fourier
   coefficients
A = zeros(2*N+1,ceil(no_sample));
for n = nvec
    m=n+N+1;
    A(m,:) = c_out(m) .* exp(1i*2*pi*n*f0*tt);
end
output = sum(A);
```

The time and frequency domain plots of the output, compared with the corresponding plots of the input, are shown in Figure 5.

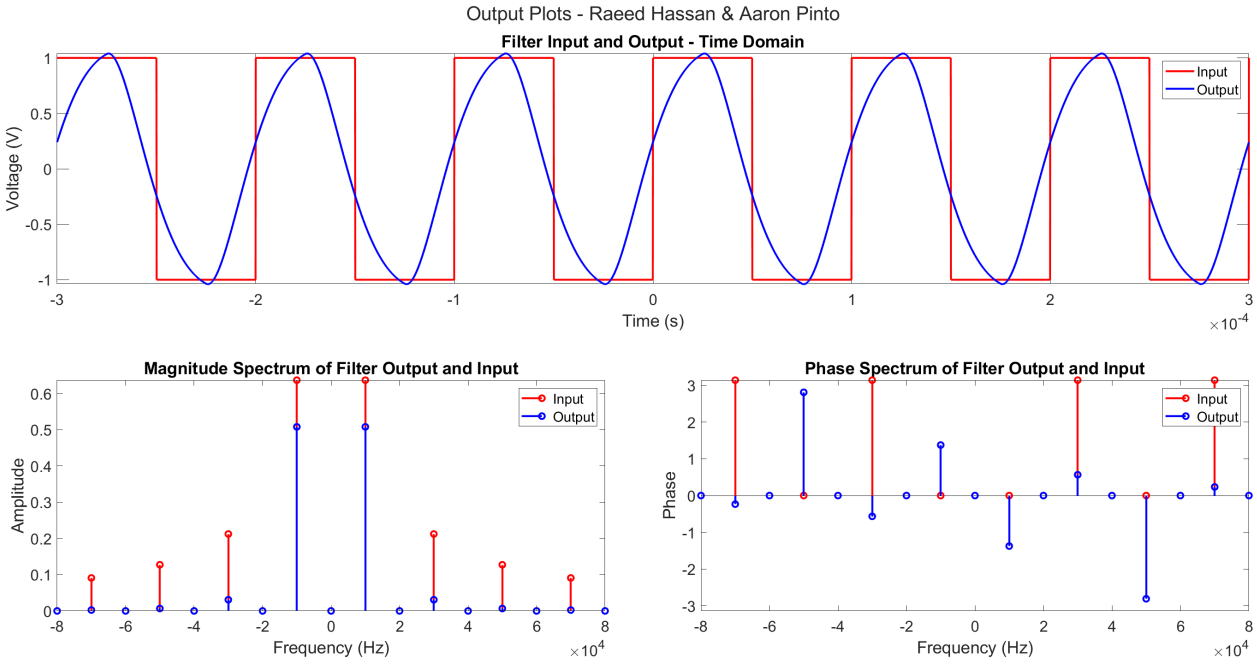

Figure 5: The time and frequency domain plots of the output signal

The output of the filter in the time domain closely resembles a pure sine wave with a period of 0.1 ms, which is what we were expecting at the output. All harmonics other than at the fundamental frequency have been attenuated (as can bee seen in the magnitude spectrum plot), leaving the majority of the Fourier series contribution to the sinusoid at the fundamental frequency (10 kHz). An evaluation of the attenuation in the magnitude

spectrum reveals that the filter satisfied the design constraints, with 1.96 dB attenuation at the fundamental frequency and 16.75 dB attenuation at the third harmonic. These values were also nearly identical to the values determined during the design process.

Conclusion

In conclusion, we have determined that a second order Butterworth filter with a cutoff frequency between 11.5 kHz and 12.9 kHz will satisfy the design constraints detailed in the design problem. This was verified analytically through calculations using the filter's transfer function, and experimentally through MATLAB simulation. The MATLAB script to determine the frequency response is attached as frequencyResponse.m, and the MATLAB script used to simulate the input, filter, and output is attached as lab1.m.