Understanding Continuous Time Signals with MATLAB

Aidan Sharpe & Elise Heim

***Abstract*—In addition to detailing some important properties of continuous time signals, this lab also highlights some of the strengths of using MATLAB as a visualization and computation tool for understanding these signals. Some core topics include different fundamental continuous time signals, how signals with different properties interact when added, and the relationship between signals expressed with code and with equations.**

I. Objectives

Through lecture, several fundamental continuous-time signals were studied. The relationship between these signals and their geometry and calculus was analyzed. Now, it is time to understand this relationship by modeling such signals so that they can be visually understood. This opens the door to simulating both fundamental and complex signals, as well as their interactions. The primary objective of this lab was to gain this pragmatic understanding, and to be able to apply it in conjunction with prior knowledge of signals.

II. Introduction

Continuous-time signals are as their name suggests: a signal that is defined for all time. Discrete-time signals only have integer values of time. This makes these signals much easier to process. The primary method used in this lab for handling continuous-time signals consists of a process called sampling. This can be done by modeling them as discrete-time signals. Modeling these signals in this way can allow for a more concrete visual understanding of continuous time signals.

III. Background

A few basic continuous-time signals utilized in this lab include the unit step and ramp functions. The unit step is defined as a function wherein the value is equal to 0 when time is less than 0, and 1 when time is greater than 0. It looks similar to a step, as its name suggests. The appearance of the ramp function also coincides with its nature. This function is 0 when time is less than 0, and it linearly increases with a user-defined slope when time is greater than 0. These fundamental continuous-time signals can be advanced or delayed, as well as compressed or stretched. They can also be easily modeled by setting specific values of time as integers. By having a great understanding of basic signals, similar principles can be applied to more complex signals.

IV. Results & Discussion

First, the unit step signal was copied and modified into the ramp signal. This modification is seen in listing 1. Specifically, the y value was redefined from 1 to a being based on the time, t, advance, ad, and slope, m.

```
if t(i) >= -ad,
    y(i) = (t(i) + ad)*m;
end
```

Listing 1. Unit Step to Ramp Function

After adding the ramp signal, several elementary operations were applied to scale and shift both the ramp and unit step signals. In doing so, their behaviors under these operations could be visualized without the need for rigorous manual computation. Simple examples of this can be seen in figures 1 and 2. At a glance, the shape and position of the signal can be seen and interpreted.

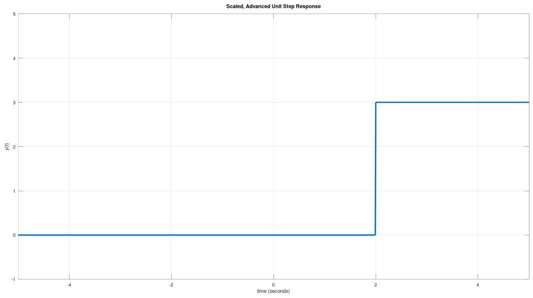

Fig. 1. Scaled, advanced unit step response

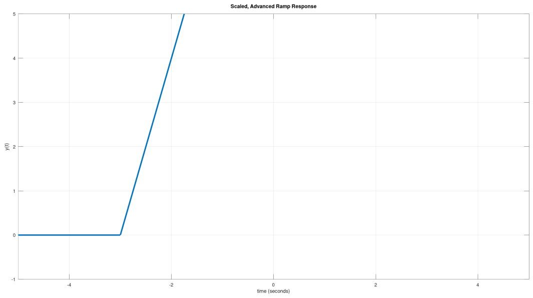

Fig. 2. Scaled, advanced ramp response

After evaluating quite simple operations on the ramp and unit step, a combination of them was analyzed. The signal defining this combination is seen in equation 1. The signal, $y(t)$ is comprised of 3 differently scaled and advanced ramp signals and a scaled and advanced unit step. As seen in figure 3, the result is much more complex than any of the previously analyzed signals.

$$y(t) = 3r(t+3) - 6r(t+1) + 3r(t) - 3u(t-3) \qquad (1)$$

Looking back at figure 3, it is apparent that the signal described by $y(t)$ is of finite duration from $t = -3$ to $t = 3$. Therefore, it should be noted that it is possible to combine infinite support signals to yield a finite support one.

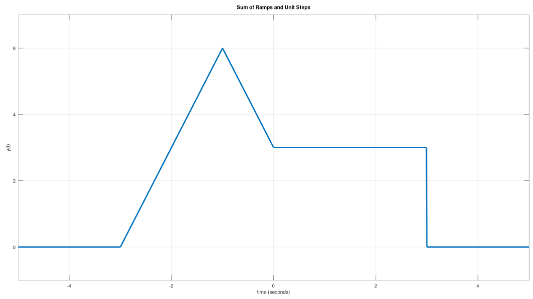

Fig. 3. The sum of several ramps and unit steps

With this basic understanding, doing the reverse—interpreting code as a combination of signals—was attempted. Given the script shown in listing 2. It is known that any call to the `ramp` function will translate to some $ar(t-\alpha)$, and any call to the `ustep` function will result in some $au(t-\alpha)$. Knowing this, the script from listing 2 was translated to the signal seen in equation 2.

```
clear all;
clf;
t = -5:0.01:5;
y1 = ramp(t,2,2.5);
y2 = ramp(t,-5,0);
y3 = ramp(t,3,-2);
y4 = ustep(t,-4);
y = y1 + y2 + y3 + y4;
plot(t,y,’k’);
axis([-10 10 -2 6]);
grid
```

Listing 2. Provided MATLAB script

$$x(t) = 2r(t+2.5) - 5r(t) + 3r(t-2) + u(t-4) \qquad (2)$$

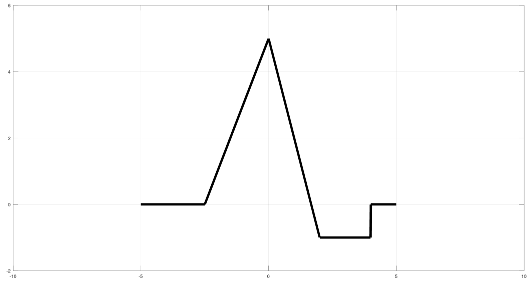

Fig. 4. Plot from provided MATLAB script

After piecing together the continuous time signal, $x(t)$, for the plot shown in figure 4, the even and odd components were found. While this can be done arithmetically, we employed the use of a function, `evenodd`, which returns both the even and odd components of a continuous time signal. Seen in figure 5, the even component is shown in blue, while the odd component is shown in orange.

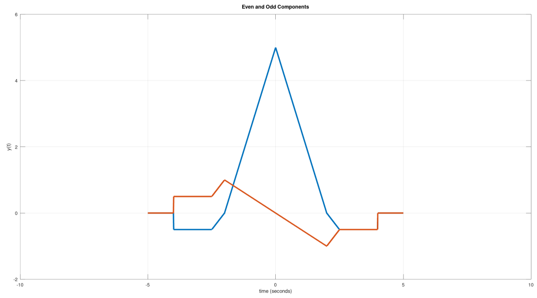

Fig. 5. The even (blue) and odd (orange) components of $x(t)$

Sometimes, in real world scenarios, signals must be altered from their original form to suit specific circumstances. For example, in order for an acoustic signal that is 3.3 minutes long to be played on a radio station for a 3-minute-long segment, the signal cannot be used exactly as it is. Now, the radio station could simply cut off the extra 0.3 seconds from the end of the signal. However, if this were to be a song, it may cut off in the middle of a word. For this reason, it may be better to speed up the signal by compressing the data. By scaling the signal by a factor of 10/11 (0.90909...), the signal can be shortened in order to fit in the allotted time. This will shorten the signal's period.

The period of some signals can be easily shown by modeling them. The function of $x(t)$ as shown in 3 is comprised of two cosine functions summed together. This function is in fact periodic, as it repeats itself at a regular time interval of every 6 seconds.

$$x(t) = \cos(\pi t) + \cos(2\pi t/3)) \qquad (3)$$

From the equation alone, it may not be intuitive to discern its period. Without already knowing how a cosine works, it may not be obvious whether the signal was periodic or not in the first place. However, visually modeling the equation yields an almost obvious period, as demonstrated in figure 6.

The cosine is a periodic function. It is probably the most important periodic function, and arguably *the* periodic function, since it is the basic building block of all periodic functions. Even when a cosine function is stretched and summed with another stretched cosine function, the result is importantly still periodic. For this reason, with a little bit of calculus trickery, every periodic function can be expressed by the sum of cosines of differing frequency and amplitude.

V. Conclusion

This lab has shown how modeling signals can be beneficial to understanding them. Basic continuous-time signals such as

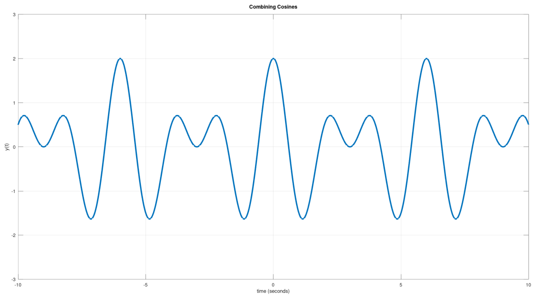

Fig. 6. The sum of two cosines

ramps and unit steps can be stretched, compressed, advanced, delayed, and summed or subtracted from each other. These alterations to the signal can be better understood by modeling them.

One thing that was very nice about having visual representations of the signals was how easy it was to find the period of a periodic signal. Finding the period was as easy as taking the difference between the time of two peaks.

Overall, using MATLAB has several advantages, since it reduces the amount of manual computation required and also makes signals more intuitive with a visual model. Having the skills to create these models quickly will prove invaluable in the long run.