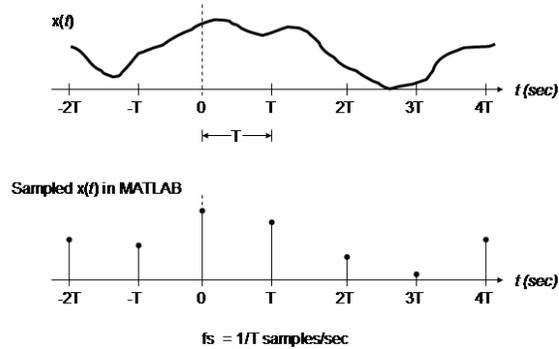


EE322 Lab 03: DTMF Signal Generation

Introduction

In this lab, you will write functions to generate dual-tone multi-frequency (DTMF) signals, the kind you hear using a touch-tone phone. These signals are sums of two sinusoids, but realize that a continuous signal (such as a sinusoid) cannot actually be created in MATLAB, as computers can only approximate them with discrete signals. These discrete signals are *sampled* versions of the continuous-time function, and the samples are T seconds apart. This means that the continuous signal is sampled at a sample frequency of $f_s = 1/T$ samples per second. The sampling of a CT signal is illustrated in Fig. 1 below.



There is a minimum rate at which an analog signal may be sampled (called the *Nyquist* rate), which is twice the highest frequency content of the analog signal—more on this later in the course. So any signal that you wish to create in MATLAB is subject to the Nyquist rate criteria.

A table that summarizes the tones associated with each button on a touch-tone phone is shown below. Note that the last column is not used on a typical touch-tone phone (digits A, B, C and D), but can be used for data transmission.

		High-Group Frequencies			
		1209Hz	1336Hz	1477Hz	1633Hz
Low-Group Frequencies	697Hz	1	ABC 2	DEF 3	A
	770Hz	GHI 4	JKL 5	MNO 6	B
	852Hz	PRS 7	TUV 8	WXY 9	C
	941Hz	*	OPER 0	#	D

Since DTMF tones are pairs of sinusoids, for each digit of a phone number you will need to create two separate sinusoids (use cosines), with the two frequencies as given in the table above, and add them together. We must be concerned that the sinusoids we create are subject to the Nyquist criteria. The minimum sample rate to be used with DTMF tones, given the table above is $2 \times 1633 \text{ Hz} = 3266 \text{ samples/sec}$ (or also referred to as 3266 Hz). This is because 1633 Hz is the highest-frequency sinusoid we will use. **For this lab, we will use a sample frequency higher than the Nyquist rate, $f_s = 12000 \text{ Hz}$.**

I. DTMF Functions

Use the sample function contained in the course policy statement to guide you in writing your functions.

A. *isvalidDTMF.m*

Write a function called *isvalidDTMF* that will test if an input character is a valid DTMF identifier. The valid DTMF identifiers are: `'0'`, `'1'`, `'2'`, `'3'`, `'4'`, `'5'`, `'6'`, `'7'`, `'8'`, `'9'`, `'*'`, `'#'`, `'A'`, `'B'`, `'C'`, and `'D'`.

Some examples of how it will be used:

```
>> y = isvalidDTMF('5');
```

or

```
>> a = '2';, v = isvalidDTMF(a);
```

Note: the input MUST be a one-character string (in single quotes)!!! The input MUST NOT be numbers!!!

This will return the variable *y*, which is a 1 if the input is a valid DTMF identifier, or 0 if not. If any one of the above 16 characters are input, this function outputs a 1. If not, the function should output a 0. To be a valid DTMF identifier, the input MUST be only one of the above characters...that is, an input like `'13'` or `'A1#'` or `'z'` is NOT valid.

B. *create_DTMF.m*

Write a DTMF function called *create_DTMF.m* that will create a DTMF tone for any of the 16 digits in the table above. There will be three inputs, the digit (a string one character long, entered in single quotes by the user), the time duration in seconds (also entered by the user), and the sample frequency. You can use *if/elseif* statements to create the proper DTMF tone.

This function must call your *isvalidDTMF* function to check if the input is a valid DTMF character or not.

An example of how it will be used:

```
>> y = create_DTMF('5',1.2,12000);
```

This will return the variable *y*, which is the DTMF tone for the digit '5' that lasts for 1.2 sec, with sample frequency 12 kHz. Note: for this lab, the sample frequency will ALWAYS be 12 kHz.

IMPORTANT: This function should NOT PLAY ANY SOUND! The *soundsc* function should NOT appear in it. This function ONLY creates the DTMF signal output.

C. *create_silence.m*

Write a function that will produce a period of silence (zero values), at an input sample frequency. The two inputs to this function will be the time duration in seconds and the sample frequency. This function will be called *create_silence*.

An example of how it will be used:

```
>> y = create_silence(1.5e-3,12000);
```

This will return the variable y , which is a period of silence (all zero values) that lasts for 1.5 milliseconds (with sample frequency 12 kHz). Note: the simplest way to do this is to create the proper time vector, then let the output be equal to zero times the time vector.

WARNING: the silence function should NOT attempt to play any sound (i.e., the *soundsc* function should NOT be called inside of this function).

II. Phone Numbers

A. In this section you will write a MATLAB program that will create and play a set of multiple DTMF tones.

1. Write a MATLAB program called *phone_number.m* that creates and plays the DTMF tones corresponding to a 10 digit phone number that a user inputs (use the *input* function with the 's' option so that the input is read as a string). The user inputs the phone number in the form below (14 characters long):

1-xxx-xxx-xxxx (e.g., 1-410-293-6152, which is Prof. Ives' phone number)

Note that the format ALWAYS starts with a "1", includes the area code, and contains 3 dashes in the correct locations. The other entries MUST each be a digit 0-9 (note: not all DTMF characters are digits!). If the format is not correct, or the user enters an incorrect character, then an error message is displayed.

Hint: you can use a "for" loop to cycle through characters in a character string.

Hint: the characters for digits '0' through '9' are consecutive characters, so you can use a > or < sign when testing if a character in a string is a digit.

Create the overall DTMF signal using your *create_DTMF* function and *create_silence* functions. Don't forget that you can combine vectors (signals) together using a command such as:

```
>> s=[d1 s d2 s d3]; % here, d1, d2 and d3 are three digit signals, and s is silence
```

In this program, each DTMF tone should last for 50 msec, and the silence in between each digit should last for 100 msec. After creating the signal composed of ALL the digits and ALL the silences in between, play it using the *soundsc* function at the correct sample frequency (12 kHz). Demonstrate that your program works correctly for the professor for any phone number he inputs, and handles errors correctly.

For your write-up, turn in your *invalidDTMF.m*, *create_DTMF.m*, *create_silence.m* and *phone_number.m* code. Your code should have COMMENTS to help me determine how you're going about doing things—you will be marked down if you lack sufficient comments!

Demonstrate your *phone_number* program for the professor. He will test your program using erroneous as well as correct phone numbers.

BONUS !! (turn this page over)

III. Bonus—Do not seek assistance from another mid.

1. To get an extra 2% on this lab, create a function for the dial tone, called *dial_tone.m*. The dial-tone is comprised of the sum of a 350 Hz cosine and a 440 Hz cosine. Your function should have two inputs: time duration and sample frequency. It should return the dial tone signal.
2. To get an extra 5% on this lab, write a function that creates the busy signal, called *busy_signal.m*. This signal is the sum of a 480 Hz cosine and a 620 Hz cosine, which is on for 0.5 sec and off for 0.5 sec. Your function should have two inputs: time duration and the sample frequency. It should return the busy signal.

Demonstrate your functions to the professor for the bonus credit.