Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
Department of Mechanical Engineering

6.050J/2.110J **Information and Entropy** **Spring 2005**

Issued: January 31, 2005 **Due: February 4, 2005**

Problem Set 1-2

Laboratory Assignment

In 6.050J/2.110J, MATLAB will be used frequently in problem sets and demonstrations. MATLAB is a mathematics software package that is efficient at matrix operations as well as a good tool for data visualization. Access MATLAB on any Athena workstation by typing `add matlab` at your `athena%` prompt. Then type `matlab &` to enter the program. A separate window will appear, in which you can enter MATLAB commands.

Whenever a problem requires MATLAB, write a text file that holds the commands or functions that you would normally type into the MATLAB command line. This text file, also known as an M-file, should have a `.m` extension (e.g., `filename.m`). This file can be executed in the MATLAB window by typing the filename without the `.m` extension. This takes away the drudgery of retyping commands over and over again. Programs like vi, textedit, emacs, and pico can be used to create and edit the M-file on Athena. Moreover, we also request that you type `diary` (in MATLAB) which keeps track of all your commands and outputs from MATLAB. The results are placed in a file called diary. Please create a separate M-file for each problem and edit the diary for readability.

Go through the MATLAB Tutorial (http://www-mtl.mit.edu/Courses/6.050/2005/notes/matlab.pdf) given out in class to familiarize yourself with the syntax and environment. Remember that at any time, you can type `help` at the MATLAB prompt (before typing this, type `more on` to turn on page-by-page viewing). It is our intention that MATLAB be used as a helpful tool in this course, and that it not be a barrier to anyone's success in the course. We also hope that you'll develop skill with MATLAB which may be useful to you in other classes. To that end, we're here to help you if you run into problems, so feel free to ask us for assistance. If you have any questions, please email 6.050-staff@mit.edu.

Problem 1: No-*NAND* Workaround

You are finishing up your robot control system late one night and run out of *NAND* gates. The stockroom is closed for the evening, and none of your friends has any *NAND* gates. You only have OR and NOT gates available.

The 6.050 TA suggests that you implement *NAND* with *OR* and *NOT* gates using a circuit like the one on figure 1–1.

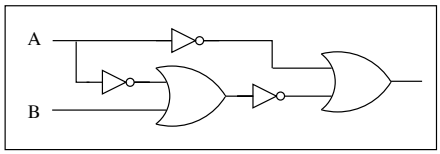

Figure 1–1: Logic circuit

You wonder if this circuit is actually equivalent to the *NAND* function, so you decide to find out.

a. Using MATLAB, test this expression for all possible values of A and B, and see whether it is what you want. Hint: MATLAB has functions *OR*, *AND*, and *NOT*. These functions can be applied using |, &, and ∼, respectively.

b. The Professor walks in and informs you that the project is almost out of money. She notes that your circuit uses two *OR* gates, which are expensive, and asks if there is another circuit that you can use instead. Either find a less costly circuit, and give its Boolean expression, or write a paragraph justifying the need for two *OR* gates.

Problem 2: Morse Encoder

In this problem you will program a Morse Encoder in MATLAB that will both **display** and **play** the Morse code. For your convenience, we have outlined a way of doing so below. To make the program shorter, you may limit the Morse encoder to work only with the following eight characters and the space:

$$EISHTMOC.$$

Note that space is not considered a character, so you should convert it to a word separation or seven spaces. Write a MATLAB function[1] that receives text as input and returns the corresponding Morse code in written form, that is, returns a string composed of dots ('.'), dashes ('-'), and spaces according to the specifications in Table 1–1.

Code	Representation
Dot	'.' (period)
Dash	'-' (hyphen)
Separation between the symbols (dots and dashes) within the code of one character	1 space
Separation between characters	3 spaces
Separation between words	7 spaces
End of Code	3 spaces

Table 1–1: Specifications for the output of morse.m

As an example, consider the following encoding

OH MOOSE ⟶ $\underbrace{-\square-\square-}_{O}\square\square\square\underbrace{.\square.\square.\square.}_{H}\square\square\square\square\square\square\square\underbrace{-\square-}_{M}\square\square\square\underbrace{-\square-\square-}_{O}\square\square\square\underbrace{-\square-\square-}_{O}\square\square\square\underbrace{.\square.\square.}_{S}\square\square\square\underbrace{.}_{E}\square\square\square$

where we have used □ to represent the space. Note the three spaces between characters and the seven spaces between words.

We ask you to turn in the morse.m function and its execution on the following sentences as part of the diary for this problem:

'Choose EECS', 'Choose MECHE', 'SOS'

The following 4-step procedure is a suggestion to program the Morse encoder

[1] A MATLAB function is simply an m-file (you wrote such files in the Laboratory Assignment above) that starts with the following syntax:

`function [output]=FunctionName(inputs)`.

For the Morse encoder create a file called morse.m and have its first line read: `function [coded]=morse(tocode)`, so that the input is stored in the variable `tocode`, and your result is to be stored in the variable `coded`. The function can be invoked from the MATLAB command line with `output=morse('some text')`.

- Create a function with the text to be encoded as input that outputs the Morse encoding. Both input and output shall be strings. Name this function morse.m.
- The function must start defining two vectors. One with the eight characters above, another with their corresponding Morse code. The Morse code matrix, should be a matrix of strings; with each row representing the Morse code for one character.

  Note: You can find the table for Morse code and the conventions for separation of characters and words in table 2.8 of the lecture notes. Strings in MATLAB are embedded within single quotes, The MATLAB function char is useful to construct tables of strings. Note that padding will occur in constructing a matrix out of strings of different length, i.e., spaces will be inserted at the end so that all strings end up with the same length.
- Replace each character by the corresponding code. You may use `find` to locate each character in the input text and `strrep` (to replace the character by its Morse code), and `deblank` (to remove the padding blanks at the end of strings.)

  Note: When choosing the order in which to replace characters by their morse code, you should pay attention to spaces in the original text. According to table 1–1, each space in the original text should be replaced by 7 spaces. However, because all the replacements involve adding spaces (for symbol, and character separation), if you replace spaces at the end you will inadvertently replace also those spaces that were added by previous replacements.
- The final step is adding sound to your encoder. You can do so using the sound function of MATLAB. Or you can use the function unitsound.m[2] that we provide below:

  Beginning of file unitsound.m

```
function []=unitsound(n,f)
%usage unitsound(n,f)
% unitsound will play a sound at frequency f (in Hz) for n
% times 0.15 seconds
% n - number of unit length intervals ( unit duration is .15 seconds)
% f - frequency of the sound.  440Hz is the A above middle C.
% We recommend 1300, for beeps and 0 for silences,
% and strongly discourage trying higher frequencies,
% (any frequency higher than ∼ 4000Hz will not be
% played correctly, do so at your own risk).
fs=8192;
sound (sin(f*2*pi*(0:(1/fs):(n*.1))),fs)
% MATLAB does not stop execution when playing a sound,
% therefore, to prevent all characters from being played
% at once we must force MATLAB to pause for the duration of
% the sound played
pause((n*.15));
```

  End of file unitsound.m

  Note: You may use the MATLAB function `sound` *to play sounds instead of* `unitsound`*. Indeed, MATLAB can play any waveform if it is provided as a vector of samples. A simple harmonic of*

[2] in MATLAB functions, the commented lines that follow the function declaration are often referred to as help comments. They are displayed if you type help followed by the name of the function at the MATLAB prompt. Try it! After copying `unitsound.m`, execute `help unitsound` at the MATLAB prompt, and see what you get.

the form $sin(\omega t)$ will be enough for this example. For instance: playing the A above middle C note during 5 seconds (often taken to be $f = 440Hz$, $\omega = 2\pi f$) can be achieved with the command: `sound(sin(440*2*pi*(0:(1/fs):5)),fs)`[3]*, where* `fs` *is the sampling rate, (a good choice being 8192Hz.)*

Turning in Your Solutions

You may have up to three files (two M-files and one diary) in addition to the files for the functions you are asked to write. Name the M-files `ps1p1.m` and `ps1p2.m` and name the diary `ps1diary`. You can rename your files at your `athena%` prompt using the following command.

```
mv oldfilename newfilename
```

Turn in this problem set by e-mailing your M-files and diary along with your answers to any problem(s) not done using MATLAB, to 6.050-submit@mit.edu. You may do this either by attaching them to the e-mail as text files, or by pasting their content directly into the body of the e-mail (if you do this, please somehow indicate where each file begins and ends). If you decide to turn in your set electronically, be sure to follow the 6.050 Electronic Submission Guidelines. Alternatively, you may turn in your solutions on paper in Room 38-344. The deadline for submission is the same no matter which option you choose.

Your solutions are due 5:00 PM on Friday, February 4, 2005. Later that day solutions will be posted on the Web.

[3] when a, b, and c are numeric, MATLAB will react to the command `a:b:c` by generating a vector of b-spaced elements from a to c. Thus `0:(1/fs):5` will produce a vector of numbers from 0 to 5 spaced by exactly `1/fs`