MISINTERPRETATIONS WHEN USING MATLAB IN DSP EDUCATION

Jukka Parviainen and Olli Simula

Laboratory of Computer and Information Science, Helsinki University of Technology,
P.O.Box 5400, FIN-02015 HUT, FINLAND, [parvi,Olli.Simula]@hut.fi

ABSTRACT

In the DSP education practical examples and demonstrations reveal theory better than one thousand words. Matlab (by The MathWorks, Inc.) is widely used as a tool for teachers and students. However, it is possible to make some mistakes, which can lead to erroneous results or conclusions even if the syntax of the code is correct. In this paper several such cases from the assignments of DSP basic course during 1999-2003 are introduced and discussed.

1. INTRODUCTION

Matlab is a widely used scientific software in engineering. There are plenty of toolboxes for a variety of fields. Matlab and its Signal Processing Toolbox [1] are very popular in DSP books nowadays [2] [3] [4]. The campus licenses are available for students, who can evaluate code and create new functions of their own.

Matlab interprets commands line after line. The interpreter gives an error message if a variable or a function name does not exist, the command syntax is not correct, or the vector and matrices are not of the right size. In these situations the user has to find the reason for an error. Matlab can normally give a hint in error line, what was wrong. For example, a correct function for computing cosine is `cos()`, not `cosine()`

```
>> cosine(3.14)
??? Undefined function or variable 'cosine'
```

This paper focuses on cases where no errors happen but the results are misunderstood. Each time the user receives the result from the computer, she has to evaluate the sensibility of the result. If the user does not fully understand what the theory in background is, what could be the range of results, or how the algorithm works, there is always a risk for mistakes. This type of mistake in Matlab can be described as a situation, where all needed variables and functions are found and typed syntactically correct and Matlab provides also results in a form of a number or a figure. Unfortunately, the result is nonsense.

At Helsinki University of Technology (HUT) there is a basic course for digital signal processing, T-61.246[1], consisting of analysis and synthesis of linear and time-invariant (LTI) digital filters. A practical assignment using Matlab is a compulsory part of the course. It is a set of small tasks using the help of Signal Processing Toolbox.

[1] `http://www.cis.hut.fi/Opinnot/T-61.246/`

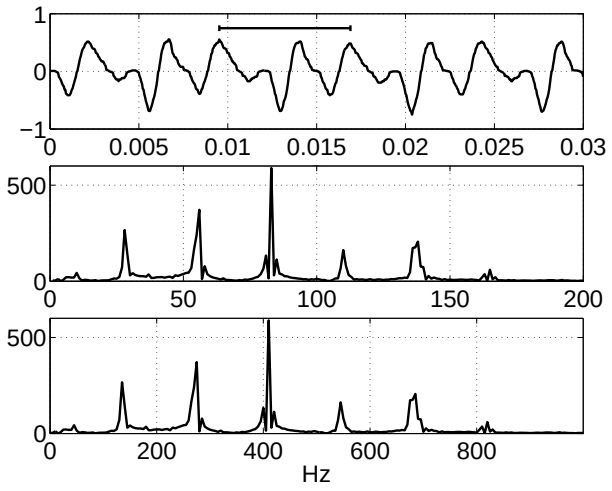

Figure 1. Waveform of a vowel "o" in range $0 \ldots 0.03$ s, the (appr.) fundamental period is 7.4 ms and marked with a segment of line (top). The spectrum with index numbers in horizontal axis using `plot(yF)` (middle), and the correct spectrum using `plot(w, yF)` (bottom). The fundamental frequency is 134 Hz.

The assignment is executed in small groups of one to three persons. Normally, the students do the work after the end of all lectures, when they know the theory in background. During 1999 and 2003 there has been approximately 1,000 assignments.

The cases are divided into two groups. In Section 2 there are examples where the user does not fully understand how to use Matlab and makes a mistake therefore. Examples in Section 3 reveal the limitations of numerical methods with floating point arithmetic. The examples are adopted from student assignments during the last years. They are executed using Matlab 6.1 (R12), Signal Processing Toolbox 5.1, on the Linux platform (GLNX86).

2. EXAMPLES OF FAULTY USE OF MATLAB

2.1. Fundamental frequency of a periodic signal

Human voice is a nice and understandable signal to analyse. Students were asked to record a vowel and to analyse it both in time and frequency domain. Students plotted a 30 ms long piece of waveform (as in top of Figure 1) and measured the fundamental period (signal was considered to be periodic enough). After that it was asked to use Fast

Fourier transform with `fft` and draw the spectrum of the signal. The pedagogical point was to show that the fundamental frequency is the inverse of the fundamental period $f = 1/T$, and discuss on the meaning of harmonics.

A very typical mistake occured when plotting the spectrum. An example is shown in Figure 1. In this particular case, the period was correctly received as 7.4 ms (top), and from that the frequency was computed to be 134 Hz. Now if the spectrum is plotted using `plot(yF)` (middle) instead of `plot(w, yF)` (bottom), the horizontal axis is erroneous with respect to the frequency.

Some students, who got the spectrum like in the middle figure, blindly believed what Matlab printed out. Their answer for the frequency was 27 Hz, or then they tried to explain results in various ways, e.g. that the fundamental frequency is the highest peak in the spectrum or it is a mixture of all peaks. The misinterpretations were due to the faulty use of `plot`. If `plot` is used with one input argument only then the x-axis contains index numbers of the vector `yF`, not any Hertz values as could be thought.

```
% Code for Figure 1
[y, fs, nbits] = wavread('vowel_o.wav');
t = linspace(0, length(y)/fs, length(y));
plot(t,y);   % plot the waveform
yF = fft(y);
plot(yF);    % plot the spectrum (WRONG)
w = linspace(0, fs, length(yF));
plot(w, yF); % plot the spectrum (CORRECT)
```

2.2. Computing DFT of length N

Another very typical pitfall in the previous task was also found when students were asked to compute Fourier transform of a (voice) sequence. Some students had learnt that the second argument in `fft` computes DFT in `N` points. It is common to think that the command computes some kind of average spectrum using, e.g. 512 points. In fact, it computes transform from the first 512 samples of `y`.

There is an example in Figure 2. A voice signal containing a vowel `/a/` is recorded and plotted in the top figure. Unfortunately, there is a short silence in the beginning from which the transform is actually computed (middle).

```
% Code for Figure 2
[y, fs, nbits] = wavread('vowel_a.wav');
t = linspace(0, length(y)/fs, length(y));
plot(t,y);     % plot the waveform
yF1 = 10*log10(abs(fft(y,512)));
w1 = linspace(0, fs, length(yF1));
plot(w1, yF1); % plot the spectrum
               % from first 512 samples
yF2 = 10*log10(abs(fft(y)));
w2 = linspace(0, fs, length(yF2));
plot(w2, yF2); % plot the spectrum
               % from the whole signal
```

2.3. Pole-zero plot of a Hamming window function

Students were asked to analyse a causal Hamming window function $w[n]$ of length $N + 1$ both in time and frequency domain

$$w[n] = 0.54 - 0.46\cos(2\pi n/N), \quad 0 \le n \le N$$

Again the Matlab code `zplane(hamming(20),1)` was syntactically correct, but if one compares it to the amplitude response, there is a mistake. If the first argument in `zplane` is a row vector, then the argument should contain the coefficents of the transfer function polynomial, if it is a column vector, then it treats the items as locations of zeros. Hence, correct is `zplane(hamming(20)',1)` where the vector is transposed to a row vector. The figures are given in Figure 3.

```
% Code for Figure 3
w = hamming(20); % column vector!!!
stem([0:19], w); % plot w[n]
[W, omega] = freqz(w, 1);
plot(omega/pi, 20*log10(abs(W)));
          % plot the amplitude spectrum
zplane(w, 1);  % a faulty pole-zero plot
zplane(w', 1); % a correct pole-zero plot
```

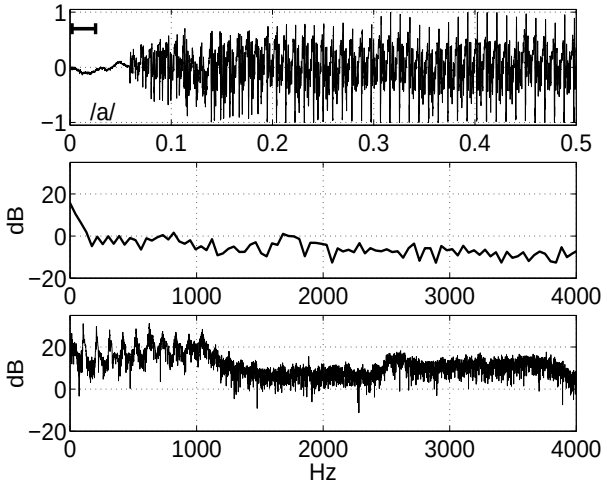

Figure 2. Waveform of a voice sequence in range $0 \ldots 0.5$ s (top), the segment of line in top left corner represents the first 512 values of the sequence from which the spectrum in the middle figure is computed using `fft(y, 512)`. The bottom figure is FFT of the whole sequence.

2.4. The order of bandpass and bandstop filters

Students were asked to design lowpass and bandpass filters with given specifications and write down the order of the filter. The command `buttord` estimates the required order and returns variables `N` (order) and `Wn`, which are fed into `butter` computing coefficients. However, in case of a bandstop (bandpass) filter `N` is only half of the order of the filter. Some students relied on the variable `N` given by the computer even if they printed out the correct transfer function.

```
[N,Wn] = buttord([.2 .8], [.3 .7], 1, 20);
[B,A] = butter(N, Wn);  % N equals 5
filt(B, A)   % H(z)= ...z^-10 / ...z^-10
```

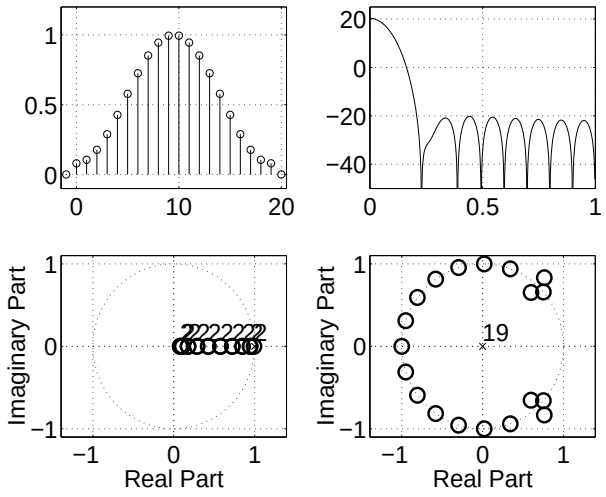

Figure 3. Hamming window function: coefficients $w[n]$ using `stem(0:19,w)` (top left), $|W(e^{j\omega})|$ (top right), a pole-zero plot of a Hamming window function using `zplane(w,1)` (bottom left), a correct plot using `zplane(w',1)` (bottom right).

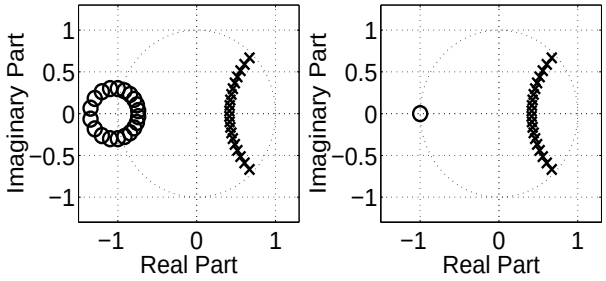

Figure 4. Zeros and poles of a digital Butterworth lowpass filter of order 20 and cut-off frequency $\omega = 0.25\pi$. A plot given by Matlab (left) and a corrected one with all zeros at $z = -1$ (right).

3. EXAMPLES OF NUMERICAL NATURE OF MATLAB

All zeros of a digital Butterworth lowpass filter (through bilinear transform) are located at $z = -1$. A pole-zero plot of a 20th order lowpass filter with 3dB cut-off frequency at $\omega = 0.25\pi$ is given with

```
% Code for Figure 4a
[B,A] = butter(20, 0.25);
zplane(B,A);    % pole-zero diagram
```

However, the pole-zero plot of the filter by Matlab does not have all zeros at the same point as seen in Figure 4. This is caused by universal numerical problems when taking roots of a high order polynomial [5]. An example of a 20th order polynomial is given by

$$\begin{aligned} f(x) &= (x-1)^{20} \\ &= x^{20} + 20x^{19} + 190x^{18} + \ldots + 20x + 1 \end{aligned}$$

The values of $f(x)$ should be monotonically decreasing up to -1 and increasing from -1, $f(-1.3) = 3.49 \cdot 10^{-11}$ and $f(-1.2) = 1.05 \cdot 10^{-14}$, etc. As can be seen in Figure 5 (left), where $x \in -1.3 \ldots -0.7$, the result computed through the 20th order polynomial is erroneous due to numerical methods with floating point arithmetic.

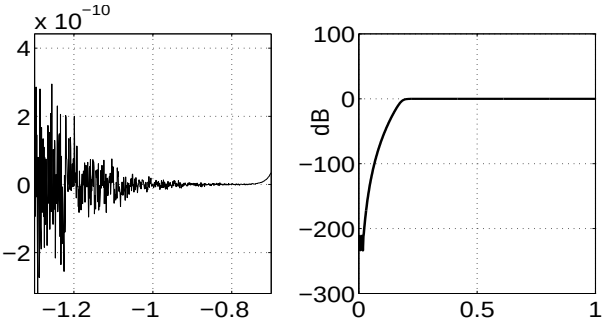

Figure 5. Two examples of numerical nature of Matlab. A polynomial $(x-1)^{20} = x^{20} + 20x^{19} + \ldots + 20x + 1$ when $x = -1.3 \ldots -0.7$ (left). Amplitude response of a digital Butterworth highpass filter (right). The range of precision ends at -240 desibels.

```
% Code for Figure 5a
x = [-1.3 : 0.001 : -0.7];
f = poly(-ones(1,20));
y = polyval(f, x);
plot(x, y);
```

4. DISCUSSION

4.1. Background of misinterpretations

There can be several reasons behind all faulty use cases. While there are no error messages the user has to find the explanation for the mistake by herself. This can be difficult if she does not know the theory in background, if the range of possible outputs is unknown, or, literally, if she is in a hurry in finishing the project before the deadline.

Some misinterpretations shown in this paper are quite common, some rare. In the autumn semester 2002 it was reported that from 207 students 70 (34 %) made a mistake in the scaling of the horizontal axis in the case of Section 2.1. It was not found any significant difference between the groups with respect to the results of the exam, see Table 1.

Group	# students	mean	std
Faulty use	70	25.2 / 48	9.72
Correct use	137	26.6 / 48	12.2

Table 1. Students having correct and faulty use of `plot` compared to their exam points.

In this particular case, it was probably an incorrect example in the first round of Matlab exercises, which led students to use `plot` with only one argument. It was not fully explained how `plot` command works. In order

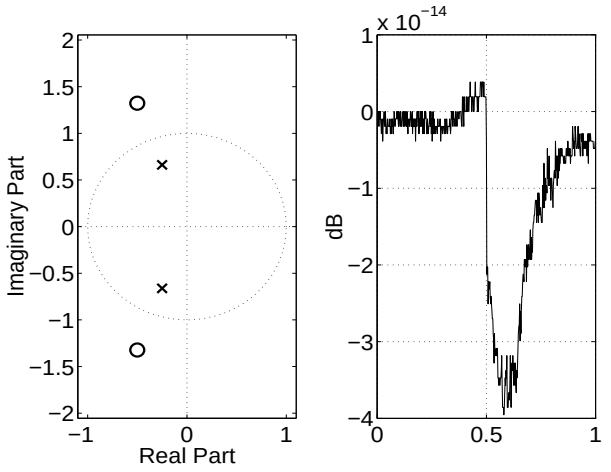

Figure 6. Second-order allpass filter, whose amplitude response looks like a bandstop if one does not notice the scale 10^{-14}.

to be suspicious to results given by the computer a student has to have an idea of possible fundamental frequencies of a human being (the range was from 60 to 400 Hz, mostly 90-140 Hz). One way to avoid mistakes is to give some small questions (checkpoints) to students: What are normal values for fundamental frequencies, how does the spectrum of a cosine look like, etc.

The other mistakes in Sections 2.2, 2.3, and 2.4 were also possible to avoid, at least, if one reads manual pages of Matlab. The command `fft(y, 512)` works nicely if `y(1:512)` contains the same frequency content as the whole signal `y`. Students may have also supposed that `fft(y, 512)` computes some kind of 512-point spectrum estimation of the whole signal as functions `psd`, `pwelch`, or `pmusic`.

Section 2.3 introduced an example of an annoying way of Matlab to overload functions respect to number or size of the arguments. The commands `freqz(w, 1)` (column vector `w`) and `freqz(w', 1)` (row vector `w'`) are identical and plot the same amplitude response. However, `zplane(w, 1)` and `zplane(w', 1)` convey different plots. Some functions return different outputs depending on how many output arguments are given. The user has to be careful in order to use functions right.

Reading the help pages of Matlab is not always enough. In the help of `buttord` it is stated explicitely that N is the order of filter. Later in the help of `butter` there is notation that the order is 2N for a bandpass or a bandstop filter.

The mistakes related to numerical computation are impossible to avoid. It can be also a lesson of finite wordlength. The user has to pay attention to the scale of the figure, for example, in Figure 6 (right), one should notice that the values are very close to 0 dB.

4.2. How to teach with Matlab

Some questions arise from the pitfalls. Should the education concentrate more on being critical to computer programs? Is it useful to give black-box programs to students if they do not know how the program works? The questions do not have right answers; the way of teaching should depend on knowledge level of students and it is also dependent on the amount of students. In a course with hundreds of students there are always those who want to have strict guidance and those who will find out the way by their own. Some common and practical advice can be given.

Demos are important part of teaching. When working with Matlab and data projector, the font size can be increased so that students in the last row are able to see. The actual learning happens by doing, so there have to be some relevant (and correct!) examples and small tests available for students. Students should be encouraged to read Matlab helps. They are also asked to write down their own code into script files, which they can rerun and edit.

There are two ways to introduce Matlab to students. Matlab can be seen as a difficult language containing vector index tricks and no `for` loops. Matlab can be also shown as a simple but powerful numerical computing software. It is practical to demonstrate Matlab as a normal calculator. One can construct a script file for computing convolution or amplitude response just using summations and multiplications - there is no need to use mysterious `freqz`. A simple averaging $y[n] = x[n] + x[n-1] + x[n-2]$ is a procedure which is understandable without any background in signal processing. It can be applied to voice signals to demonstrate time and frequency domain analysis, convolution and analysis of transfer function.

5. CONCLUSION

Matlab is a useful tool in DSP education. Students are able to learn the basics of Matlab quickly and use the functions in Signal Processing Toolbox. It is nice to see that theory works also in practice. It should be emphasized that it is not only DSP theory but also the basic tools that have to be taught.

6. ACKNOWLEDGMENTS

We thank our student, Janne Kallioniemi, for giving his voice into scientific purposes (Section 2.1).

7. REFERENCES

[1] The MathWorks, Inc, *Signal Processing Toolbox*.

[2] S. K. Mitra, *Digital Signal Processing*, McGraw-Hill, New York, 2001.

[3] S. K. Mitra, *Digital Signal Processing Laboratory Using Matlab*, McGraw-Hill, USA, 1999.

[4] J. H. McClellan, R. W. Schafer, and M. A. Yoder, *Signal Processing First*, Pearson Prentice Hall, USA, 2003.

[5] W. Cheney and D. Kincaid, *Numerical mathematics and computing*, Pacific Grove, CA, USA, 1999.