Dalius NAVAKAUSKAS
Artūras SERACKIS

DIGITAL SIGNAL PROCESSING TOOLS
Dalius NAVAKAUSKAS  
Artūras SERACKIS

DIGITAL SIGNAL PROCESSING TOOLS

Manual of Laboratory Works
The laboratory manual on “Digital Signal Processing Tools” aims to stimulate acquirement of knowledge about modern means of digital signal processing, their operating principles and possibilities of application using MATLAB™. Problems and solutions of digital speech signal processing, modelling and synthesis as well as image segmentation and data classification are analysed.

The publication has been recommended by the Study Committee of VGTU Electronics Faculty.

Reviewed by:
Prof Dr Šarūnas Paulikas, VGTU Department of Telecommunication Engineering
Assoc. Prof Dr Dalius Matuzevičius, VGTU Department of Electronic Systems

This publication has been produced with the financial assistance of Europe Social Fund and VGTU (Project No VP1-2.2-ŠMM-07-K-01-047). The book is a part of the project “The Essential Renewal of Undergraduates Study Programs of VGTU Electronics Faculty”.

This is an educational methodology book, No 1338-S, issued by VGTU Press TECHNIKA http://leidykla.vgtu.lt

Language editor Dalia Blažinskaitė
Typesetter Rasa Labutienė

doi:10.3846/1338-S

© Dalius Navakauskas, 2012
© Artūras Serackis, 2012
© Vilnius Gediminas Technical University, 2012
## CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Why?</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Who?</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>What?</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Acknowledgements</td>
<td>9</td>
</tr>
<tr>
<td>Notation</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td><strong>Introduction with MATLAB™</strong></td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>1.1 The Aim</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>1.2 Important Material</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>1.2.1 MATLAB™ Main Function</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>1.2.2 MATLAB™ Local Functions</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>1.2.3 MATLAB™ Private Functions</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>1.3 Procedure</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>1.3.1 Software Tools</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>1.3.2 Work Order</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>1.4 Tasks for Laboratory Work</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>1.5 Questionnaire</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td><strong>Design of Finite Impulse Response Filters</strong></td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>2.1 The Aim</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>2.2 Important Material</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>2.2.1 FIR Filter Impulse Response and Frequency Response</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>2.2.2 FIR Filter Design</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>2.2.3 FIR Filter Design using MATLAB</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>2.3 Procedure</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>2.3.1 Software Tools and Signal Sources</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>2.3.2 Work Order</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>2.4 Tasks for Laboratory Work</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>2.5 Questionnaire</td>
<td>33</td>
</tr>
</tbody>
</table>
3 Design of Infinite Impulse Response Filters

3.1 Aim

3.2 Important Material

3.2.1 Description of the Analog Filter

3.2.2 IIR Filter Design by Impulse Invariance

3.2.3 IIR Filter Design by Bilinear Transform

3.2.4 IIR Filter Design using MATLAB

3.3 Procedure

3.3.1 Software Tools and Signal Sources

3.3.2 Work Order

3.4 Tasks for Laboratory Work

3.5 Questionnaire

Bibliography

4 Structures of Infinite Impulse Response Filters

4.1 The Aim

4.2 Important Material

4.2.1 Realisation of the Digital Filters

4.2.2 Direct Forms of the Digital Filters

4.2.3 Cascade Forms of the Digital Filters

4.2.4 Parallel Forms of the Digital Filters

4.2.5 Realisation of the Digital Filters using MATLAB

4.3 Procedure

4.3.1 Software Tools and Signal Sources

4.3.2 Work Order

4.4 Tasks for Laboratory Work

4.5 Questionnaire

Bibliography

5 Filtering of Audio Signals

5.1 The Aim

5.2 Important Material

5.2.1 Changing the Audio Signal Sampling Rate

Bibliography
## 8 Speech Signal Processing and Analysis

8.1 The Aim ............................................. 87

8.2 Important Material ................................. 87
  8.2.1 Speech Signal Segmentation ................... 87
  8.2.2 Speech Preemphasis ............................. 89
  8.2.3 Short-Time Fourier Analysis ................... 90
  8.2.4 Formant Tracking .............................. 93

8.3 Procedure ........................................... 95
  8.3.1 Software Tools and Signal Sources ............ 95
  8.3.2 Work Order .................................. 95

8.4 Tasks for Laboratory Work ......................... 95

8.5 Questionnaire ....................................... 97

Bibliography .......................................... 98

## 9 Speech Signal Modelling

9.1 The Aim ............................................. 99

9.2 Important Material ................................. 99
  9.2.1 Glottal Pulse Modelling ......................... 99
  9.2.2 Vocal Tract Modelling .......................... 100
  9.2.3 Vowel Modelling ................................ 104

9.3 Procedure ........................................... 104
  9.3.1 Software Tools and Signal Sources ............ 104
  9.3.2 Work Order .................................. 104

9.4 Tasks for Laboratory Work ......................... 105

9.5 Questionnaire ....................................... 109

Bibliography .......................................... 109

## Appendices

A List of MATLAB® Functions .......................... 111

B Subject Index ......................................... 117
PREFACE

We preface our laboratory work manual with W\textsuperscript{3} – trying to answer important to the reader questions: Why, What and Who?

Why?

Everyone who studies needs to know why is it worth to spend hours in laboratory fullfilling prescribed manuals, that is, what is an aim and what will be the result – what knowledge, cognitions, skills and abilities will be acquired.

Aim of laboratory work manual is to acquire knowledge about development and improvement of the modern Digital Signal Processing (DSP) tools, acquire cognitions about their operating principles and application possibilities and abilities to choose a reasoned solution, working individually or in group.

Knowledge to be acquired:

- development methods and means of digital signal processing tools, operation methods of the linear and non-linear digital signal processing tools and fundamentals of the quality control;
- fundamentals of the development of digital filters, artificial neural networks, analysis and modeling of the speech signals using \textsc{Matlab}\textsuperscript{TM};
- development and analysis of the new digital signal processing algorithms.

Cognitions to be acquired:

- identify and analyse the new problems in DSP and plan the solutions strategies;
- understand theoretically the new linear and non-linear DSP technologies;
- combine the elements of theory and practice; perform laboratory and experimental works required for the engineering practice.
Special skills to be acquired:

- select and apply the mathematical methods, software and hardware tools to solve the DSP problems;
- use the specialised for DSP MATLAB™ software tools, competent use of computers for problem data acquisition and processing, automatic design of the DSP tools, visualization of the results using MATLAB™ visualisation tools;
- apply computer-based methods to solve specific problems of DSP;
- perform an experimental analysis of the digital signals, analyse and read the received data;
- design the DSP tools and systems according to the customer specifications.

General abilities to be acquired:

- present the digital signal processing tools design, modelling and analysis results, conclusions in correct written and spoken words in state language, lucid (understandable) for various types of audience;
- independently perform individual modelling tasks, responsibly and accurately organize individual work;
- working in team for common task, effectively communicate with colleagues and professionals from the neighbouring technology fields with skills, necessary to solve given task;
- use the innovative technologies for design, modelling and analysis of the DSP tools, by performing an individual or common task.

Who?

Second usually raised important question – who ought or need to learn according to the given laboratory work manual?

The manual is devoted first of all to Bachelor degree students
of VGTU Electronics Faculty that study under Electronics Engineering and Information Systems Engineering study programmes and need to exercise in depth digital signal processing and DSP tools. Students from other universities can also find here important and valuable material that can support their self-studies and preparation for various DSP related projects.

**What?**

Third important question that commonly is raised – what are the rules of laboratory work evaluation?

Evaluation formula of the laboratory work is as follows:

\[
L = 0.6 \times L_m + 0.4 \times L_c, \tag{0.1}
\]

here: \(L_m\) – mandatory tasks evaluation mark; \(L_c\) – complementary task evaluation mark.

There are several mandatory tasks in each laboratory work thus \(L_m\) is calculated as an arithmetic mean of complementary tasks’ marks.

**Acknowledgements**

The manual was prepared under pursue of the project “VGTU Elektronikos fakulteto I pakopos studijų programų esminis atnaujinimas”. Prof. D. Navakauskas prepared layout and design of the manual, wrote Chapters 1, 8 and 9; Assoc. Prof. A. Serackis wrote Chapters 2–7.


Authors express sincere thanks to the reviewers of the manual Prof. Dr. Š. Paulikas and Assoc. Prof. Dr. D. Matuzevičius for their valuable and objective comments. We also are grateful to the Editor D. Blažinskaitė for the careful and critical editorial work.
Finally this laboratory work manual would not be possible to be produced without the patience, support and love of our family members. Cosmic thanks to Rūta and Aistė from Dalius, and to Greta and Ugnė from Artūras!
Visual Aids

Each section devoted for presentation of important material starts with the list of related MATLAB™ commands and functions:

<table>
<thead>
<tr>
<th>Related MATLAB™ commands and functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>help, load, edit, save.</td>
</tr>
</tbody>
</table>

MATLAB™ functions and any fragment of computer script are outlined by special font, e.g., `help`.

Windows, drop-down menus, buttons, fields and selections used in software are also outlined by special font, e.g., Favorites.

Examples in the text have unique numbers in each laboratory work, are entitled and outlined by horizontal lines:

**Example 0.0 Speech Signal Waveform Presentation**

Let’s develop function (*ssw.m*) that reads speech signal data from a file and presents it as a waveform in a graphic window.

MATLAB™ script that consists of several lines of code is presented as a separate part of the text with each line numbered:

```matlab
function a = first(b, c);
a = b + c;
```

The second line of shown script is referenced as follows L2.

Algorithms in the text have unique numbers in each laboratory work, are entitled and outlined by horizontal lines:

**Algorithm 0.0 Short-Time Fourier Analysis**

1. Select L signal samples at n-th time instance …
5. Repeat Steps 1–4 for each n-th value.
Tasks have unique numbers in each laboratory work, are outlined by horizontal lines with numbering shown in margins:

<table>
<thead>
<tr>
<th>Task 0.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using <code>helpwin</code> or <code>helpdesk</code> commands please find the function that evaluates Bessel function of the second type.</td>
</tr>
</tbody>
</table>

Remarks or hints have unique numbers in each laboratory work, are outlined by horizontal lines with numbering shown in margins:

<table>
<thead>
<tr>
<th>Hint 0.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>During <code>MATLAB™</code> code composition do not forget to:</td>
</tr>
<tr>
<td>✓ partition script into separate functions;</td>
</tr>
<tr>
<td>✓ write down comments nearby important processing steps.</td>
</tr>
</tbody>
</table>

### Abbreviations

- **AM**: Amplitude Modulation
- **DFT**: Discrete-Time Fourier Transform
- **DSP**: Digital Signal Processing
- **FFT**: Fast Fourier Transform
- **FIR**: Finite Impulse Response
- **FM**: Frequency Modulation
- **GUI**: Graphical User Interface
- **IIR**: Infinite Impulse Response
- **MLP**: Multi-Layer Perceptron
- **PM**: Phase Modulation
- **PPM**: Pulse-Position Modulation
- **PWM**: Pulse-Width Modulation
- **QAM**: Qadrature Amplitude Modulation
- **RBF**: Radial Basis Function
- **SFT**: Short-Time Fourier Transform
INTRODUCTION WITH MATLAB™

1.1 The Aim

Knowledge about basic commands and functions of program MATLAB™ was acquired during the course Script Programming (given at VGTU Faculty of Electronics) or equivalent other courses and self-study at home.

This laboratory work aim to enforce the knowledge about programming in MATLAB™ and extend it with main, local and private functions development through out execution of practical tasks.

It stipulates and develops programming, data analysis and processing skills, abilities to independently apply computerized techniques in order to solve specific practical problems.

1.2 Important Material

### Related MATLAB™ commands and functions

- else, end, fft, function, help, if, lookfor, nargin, nargout, sqrt, while.

1.2.1 MATLAB™ Main Function

Program MATLAB™ functions are sets of MATLAB™ command lines that are saved in text files having extension .m. They differ from MATLAB™ scripts in a way that keyword function is used (see code below). That action declares function and lets to input or/and output data through out declared input or/and output arguments of the function (Knight 2000: Chapter 38).

```matlab
function [s, d] = myfirst (a, b);
%MYFIRST Calculation of sum and difference
% [s, d] = myfirst (a, b)
% a, b - input arguments,
% s - result of summation,
% d - result of subtraction.
% Copyright by DN
```
In the first code line (L1) function is declared. In case there is no need of output arguments they can be omitted together with equality sign. Function name must be present and it must be the same as the file name. In case there is no need of input argument they can be omitted together with opening and closing brackets.

Code at L2-6, L8 and L9 starts with comment symbol thus it stands for function comment. Keep noted for L2 – this line is called H1 (H from help). If after the function declaration there will be no empty or command lines then program MATLAB™ will use given comment lines to present them as a help for the user. Just typing in the command window `help myfirst` will produce following output:

```
» help myfirst
MYFIRST Calculation of sum and difference
    [s, d] = myfirst (a, b)
    a, b - input arguments,
    s - result of summation,
    d - result of subtraction.
```

While the `lookfor myfirst` will just output content of L2:

```
» lookfor myfirst
MYFIRST Calculation of sum and difference
```

Thus it is common at H1 line to present function’s title and purpose. Afterwards (e.g., L3) complete function declaration is repeated in a case user forgot input/output arguments. Finally (as in L4-6) description of function arguments is given.

Frequently information about who and when prepared the function is given as comment, too. As this information not always is necessary to present for common user see included empty line – L7 – it prevents following lines (L8 and L9) to be shown in a command window.

It is important to acknowledge the fact that such invocation of function `[s, d] = myfirst(3)`, i.e., with only one input ar-
argument, will generate an error. In order to avoid such MATLAB™ function behaviour there are used two special purpose variables: nargin (number of arguments to input) and nargout (number of arguments to output). At the moment of function execution these variables gets their values and lets a programmer to prevent undesirable errors. One way of the approach is just to stop execution, e.g.:

```matlab
if nargin ~= 2,
    return;
end;
```

Another way – can be the assignment of default value to missing input argument:

```matlab
if nargin == 1,
    b = 10;
else;
    return;
end;
```

### 1.2.2 MATLAB™ Local Functions

During programming usually a huge amount of files (functions) are created. Not all of them are equally important and widely used. Some functions are very specialized thus are executed only by specific “parent” functions. In order to reduce the number of files and in some sense “hide” very specialized (or alternatively executed) functions the local function type is introduced in MATLAB™.

*Local function* – a function that accompanies main function and is written in the main function’s file. Nevertheless that local function is named differently from the main function, still it can not be invoked outside of the main function body – simply because local function is not written as a separate file. Local functions very well support program modularization and are very handy when repetitive calculation steps in an algorithm needs to be programmed, e.g.:

```matlab
function c = second(a)
% Example of the local function

second.m
```
b = 0;
while a < 1000,
    a = spec(a);
    b = b + pi/21;
end;
c = spec(b)/a;

function y = spec(x)
    % Local function
    y = sqrt(x^3) - pi/21;

It is evident that the local function (L11–13) follows after the code of the main function. It is enough to write function second time (or third, etc. if there are several local functions) and from that moment all following code will be assigned to the declared local function.

The order in which several local functions are presented in a file is not important. Probably it is natural to order them basing on general logic, e.g., from most important or from most frequently executed.

It is important to note that local function variables are also local, thus variable y is not known to function mysecond. As usually data are imputed or/and outputed through out input or/and output arguments of the function.

1.2.3 MATLAB™ Private Functions

One of local function use is to overwrite the common MATLAB™ function behavior preserving its name, e.g., calculate Fast Fourier Transform in a more specific way but invoke it with usual to MATLAB™ name – fft.

Private function – a function that can be invoked only by the owner. This behaviour is ensured “physically” when the function is written in a private subdirectory. From that moment only the user that has rights and is executing function or script in the parent directory is in a position to invoke also private functions.

Private functions as name states are for the owners purpose and give another opportunity to overwrite the common MATLAB™
or user developed and free to execute for others functions. Thought purpose of private function use is to “hide” user specific and/or probably yet undeveloped functions from the accidental execution by other program MATLAB™ users.

1.3 Procedure

1.3.1 Software Tools

A subset of MATLAB™ software package functions (The MathWorks, Inc. 2011) in the following groups – basic functions, basic plots and graphs, elementary matrices and arrays, array manipulation, polynomials, filtering and convolution, control flow – need to be used (Hunt et al. 2006; Navakauskas, Serackis 2008).

1.3.2 Work Order

The laboratory work consists of mandatory and complementary parts. Mandatory part (A and B groups) tasks need to be done first. Only if time will permit the task from complementary part (C group) should be approached. Exact task numbers (from A and B groups) will be given individually in the class by laboratory work supervisor.

1.4 Tasks for Laboratory Work

Group A Task

---

Task: *Calculations with hexadecimal numbers*

Create a function that lets to sum or subtract hexadecimal numbers:

- Input arguments: two numbers in hexadecimal form; indicator about necessary operation – summation or subtraction – to perform.
- Function does not have output arguments.
- Ensure that function will operate only with the first two input argument, too.
- Created function must present results of operation in hexadecimal as well as decimal forms.
**Task: Calculations with octal numbers**

Create a similar (for details see description of the previous task) function that lets to sum or subtract octal numbers.

**Group B Tasks**

**Task: Sorting integers**

Create a function that sorts integers in ascending order:

- **Input arguments**: array of integers to be sorted, indicator about odd or even results are needed to be outputed.
- **Output argument** – resulting sorted array of integers.
- **Ensure that function will operate only with the first input argument, too.**
- **Created function additionally must present in a graphical window given data array with sorted entries outlined by color.**

Keep in a mind that array to be sorted as individual elements can have all possible MATLAB™ values, i.e., negative and positive integers, infinities, not numbers.

**Task: Signal decimation**

Decimation of a signal (reduction of its sampling frequency) mathematically can be expressed by:

\[ y(n) = x(nM), \]

where \( M \) – a decimation constant, e.g., if \( M = 2 \), then signal is decimated taking out each second sample (Ingle, Proakis 1997).

- **Create function that performs signal decimation:**

```matlab
function y = downsample(x, m)
%DOWNSAMPLE - signal decimation.
% function y = downsample(x, m)
% x - input signal,
% m - decimation constant,
% y - decimation rezult.

% Generate signal \( x(n) = \sin(0, 125\pi n) \), when \(-50 \leq x \leq 50\).
Using created **downsample** decimate signal with $M = 4$.

Present graphically both signals in a one window.

**Group C Task**

**Task: Golden ratio**

$\phi$ – Golden ratio. It is not less intriguing number then $\pi$ or $e$ (Moler 2004). The value of $\phi$ frequently is used in technical science and can be calculated by:

$$\phi = \frac{1 + \sqrt{5}}{2}.$$  
(1.2)

One way of approximate value of $\phi$ calculation is the use of this infinite rule:

$$\phi = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \cdots}}}.$$  
(1.3)

Create a function for a Golden ratio value calculation and representation:

- Main function have to handle invocation of local functions and interaction with the user. User needs to input number of fractions to be used for calculation, $n$, and code that determines what action is needed to be performed;
- The first local function for a given value $n$ have to prepare textual representation of (1.3) and output it in request of the user;
- The second local function have to calculate Golden cut value for all $[1, n]$ cases;
- The third local function have to present graphically the Golden cut error for all $[1, n]$ cases.

**1.5 Questionnaire**

Q1. How correctly format user defined function in program MATLAB™?
Q2. What will be effect if someone will introduce empty line in a block of comments that follows H1 line?

Q3. For what purposes some functions are changed to local ones?

Q4. Is it a difference for the whole calculation under which order local functions are arranged?

Q5. What is a main difference between local and private functions?

Q6. If there are – local function, private function and MATLAB™ standard function – all with the same name, which one will be executed if invoked in main function?

Q7. If there are – local function, private function and MATLAB™ standard function – all with the same name, which one will be executed if invoked in a command window?

Bibliography


DESIGN OF FINITE IMPULSE RESPONSE FILTERS

2.1 The Aim

The aim of this laboratory is to learn the basis of a digital Finite Impulse Response (FIR) filter design accordingly to a given filter specification.

2.2 Important Material

Related MATLAB™ commands and functions

abs, fdatool, fft, fir1, fir2, fircls1, freqz, fvtool, imag, impz, log, stem, unwrap, window.

In this laboratory a linear filter digital FIR filter, having no feedback from past outputs and linear phase is analysed. The digital filters are used to modify the digital signal amplitude (the gain of the signal frequency components) accordingly to a desired frequency response. The linear phase response of the FIR filter means that all of the signal frequency components in the input are delayed by the same amount.

2.2.1 FIR Filter Impulse Response and Frequency Response

The impulse response† of the digital FIR filter is of finite duration and can be expressed by equation (Oppenheim, Schafer 2009: Chapter 7)

\[
y(n) = h_0 x(n) + h_1 x(n - 1) + \ldots + h_{M-1} x(n - M + 1) = b_0 x(n) + b_1 x(n - 1) + \ldots + b_{M-1} x(n - M + 1);
\]

(2.1)

here \(x[n]\) is the input signal sample at time \(n\), \(M\) is the number of filter coefficients \(b\). By means of the z-transform, the FIR filter can be expressed by the system function (Oppenheim, Schafer...
There are four main types of frequency selective filters: low-pass, high-pass, band-pass and band-stop (Fig. 2.1).

2.2.2 FIR Filter Design

The design of the FIR filter usually is performed by selecting the specification of the filter (the desired impulse response accordingly to the desired frequency response).

The desired frequency response is related to the desired impulse response by the Fourier transform relation (Proakis, Manolakis 2006: Chapter 7)

\[ H_d(\omega) = \sum_{n=0}^{\infty} h_d(n)e^{-j\omega n}. \]
Accordingly to the (2.3), the desired impulse response \( h_d(n) \) can be determined by evaluating the integral

\[
h_d(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(\omega)e^{j\omega n}d\omega, \text{ where } -\infty \leq n \leq \infty. \tag{2.4}
\]

The desired impulse response, defined by (2.4) is infinite in duration and must be truncated and shifted by \( M - 1/2 \) samples to design a real valued and causal FIR filter. Truncation can be performed by multiplying the desired impulse response \( h_d(n) \) by a “window” function. The “rectangular window” is defined by (Oppenheim, Schafer 2009: Chapter 7)

\[
w(n) = \begin{cases} 
1, \text{ for, } n = 0, 1, \ldots, M - 1, \\
0, \text{ otherwise.} 
\end{cases} \tag{2.5}
\]

Accordingly to (2.5), the application of “rectangular window” leaves \( M \) samples of desired impulse response and removes the rest. The effect of application of “rectangular window” to the desired frequency response of the filter can be evaluated by convolution of \( H_d(\omega) \) with \( W(\omega) \)

\[
H(\omega) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(\nu)W(\omega - \nu)d\nu. \tag{2.6}
\]

The Fourier transform \( W(\omega) \) of the “rectangular window” \( w(n) \) is defined by

\[
W(\omega) = \sum_{n=0}^{M-1} e^{-j\omega n} = e^{-j\omega(M-1)/2} \frac{\sin(\omega M/2)}{\sin(\omega/2)}. \tag{2.7}
\]

The convolution of the \( H_d(\omega) \) and \( W(\omega) \) has a smoothing effect and can be reduced by increasing the filter length \( M \). The sidelobes of “rectangular window” frequency response \( W(\omega) \) has undesirable effects on the frequency response of FIR filter. To reduce these undesirable effects, “windows” of different shapes were introduced: Bartlet (triangular), Hamming, Hanning, Lanc-
Another simple way to design the FIR filter is called Frequency Sampling Method. Using this filter design method, a set of equally spaced frequencies $\omega_k$ for desired frequency response $H_d(\omega)$ is specified. The impulse response of designed filter is

$$h(n) = \frac{1}{M} \sum_{k=0}^{M-1} H(k) e^{j2\pi kn/M}, \quad n = 0, 1, \ldots, M - 1; \quad (2.8)$$

Here $H(k) = H(\omega_k) = H\left(\frac{2\pi k}{M}\right)$; are the samples of FIR filter desired frequency response. For real $h(n)$, the frequency response is symmetric. Thus the frequency specification can be reduced to $(M + 1)/2$ for $M$ odd and to $M/2$ for $M$ even.

### 2.2.3 FIR Filter Design using MATLAB

Two MATLAB™ toolboxes support design of the digital FIR filters: Signal Processing Toolbox (The MathWorks, Inc. 2010b) and DSP system Toolbox (The MathWorks, Inc. 2010a). The classical method of windowed linear-phase FIR digital filter design is implemented in function `fir1`, found in Signal Processing Toolbox (The MathWorks, Inc. 2010b). Function returns the $b$ coefficients (see eq. (2.1) and (2.2)) for the Hamming windows based FIR filter of an order $n$ with normalised† cutoff† frequency $W_n$:

```matlab
N = 21;
f = 120; % cutoff frequency equal to 120 Hz
fs = 500; % sampling frequency equal to 500 Hz
Wn = f/(0.5*fs);
b = fir1(N, Wn);
```

The type of the window can be set by adding the additional parameter (window, specified by a column vector `window_vect`) to `fir1` function:

```matlab
N = 51;
window_vect = window(@gausswin,N,2.5); % 2.5 is the spread
b = fir1(N, Wn, window_vect);
```

The function `window` can be found in Signal Processing Toolbox.
(The MathWorks, Inc. 2010b) and is used to specify various windows (@bartlett, @hann, @barthannwin, @blackman, @bohmanwin, @chebwin, @kaiser, @flattopwin, @taylorwin, @gausswin, @nuttallwin, @hamming, @parzenwin, @rectwin, @tukeywin, @triang, @blackmanharris) by a column vector.

By default, the fir1 function designs the low-pass filter if \( W_n \) variable has one value or band-pass filter if \( W_n \) variable is a row vector with two values (\( W_n = [W_n_{low} \ W_n_{up}] \)), defining the lower (\( W_{low} \)) an upper (\( W_{up} \)) cutoff frequencies. The other types of the digital filter can be set by additional parameter ‘high’ for the high-pass filter and ‘stop’ for the band-stop filter:

```matlab
n = 50;
Wn_low = 0.2;
Wn_up = 0.4;
Wn = [Wn_low Wn_up];
b = fir1(n, Wn, ’stop’);
```

For the design of the symmetric (such as band-stop) filters the even filter order is always used. For odd filter order the frequency response at the Nyquist frequency is 0. The length of the window, used for the FIR filter should be equal to the length of the filter. The filter length is the number of impulse response samples. The order of the FIR filter is the highest power in a z-transform FIR filter representation (see eq. (2.2)).

The Frequency Sampling Method is implemented in MATLAB® function fir2. Function requires a vector of desired normalised frequency points used for approximation and vector of desired magnitude response at these frequency points.

Three main characteristics of the FIR filter (impulse response, frequency response and phase response) can be plotted by the use of impz and freqz functions respectively. MATLAB® Signal Processing Toolbox (The MathWorks, Inc. 2010b) also offers a Filter Visualization Tool (fvtool) which enables easy and quick view of various filter characteristics. For FIR filters these characteristics can be easily extracted without these specialised functions.

As it is seen in (2.1), the impulse response coefficients of the
designed FIR filter are equal to the filter $b$ coefficients. The visualisation of the impulse response can be performed by the `stem` function. Visualisation of the magnitude response and phase response requires the Fourier transform (`fft`) of the designed FIR filter impulse response. For the magnitude response, the complex modulus (magnitude)\(^\dagger\) of the Fourier transform is taken. The extraction of the designed FIR filter phase response needs the imaginary part (`imag`) of the Fourier transform logarithm (`log`). To produce the smoother phase plot, the additional `unwrap` function is usually used.

### 2.3 Procedure

#### 2.3.1 Software Tools and Signal Sources

MATLAB\textsuperscript{TM} software package (Navakauskas, Serackis 2008); DSP System Toolbox (The MathWorks, Inc. 2010a) and Signal Processing Toolbox (The MathWorks, Inc. 2010b) need to be used.

#### 2.3.2 Work Order

The laboratory work consists of mandatory and complementary parts. Mandatory part (A and B group) tasks need to be done first (sequentialy, in declared order). Only if time will permit the tasks from complementary part (C group) should be approach.

As some work group tasks have alternative work orders, please make sure that you follow the correct one work description.

Exact task numbers (from A, B and C groups) will be given individually in the class by laboratory work supervisor.

To start the laboratory, main requirements for the digital FIR filter should be set: filter type (low-pass, high-pass, band-pass, band-stop), cutoff frequencies, desired attenuation at the stop-band, type of the window to use (for FIR filter design using windowing techniques).

**Group A – FIR Filter Design using FDATool – Mandatory Part**

This mandatory part of the laboratory is dedicated to learn how to use the innovative MATLAB\textsuperscript{TM} tools for FIR filter design.
The first group of tasks (Group A) helps to get familiar with the graphical user interface (GUI) for digital filter design found in DSP System Toolbox (The MathWorks, Inc. 2010a).

1. Start the Filter Design and Analysis Tool
2. Select the filter response type accordingly to the task requirements.
3. Select the appropriate FIR filter design method (Window).
4. Select the type of the window for the truncation of desired filter impulse response.
5. Set the appropriate sampling ($F_s$) and cutoff ($F_c$) frequencies.
6. Change the order of the FIR filter to satisfy the given stop-band attenuation. If Kaiser window is used for FIR filter design, the minimum order of the filter should be evaluated automatically.

**Group B – FIR Filter Design without GUI – Mandatory Part**

This mandatory part focuses on MATLAB™ Signal Processing Toolbox (The MathWorks, Inc. 2010b) functions for FIR filter design using classical methods, such as windowing (fir1) and frequency sampling method (fir2).

**Alternative 1 – Windowing Technique**

1. Create an *.m file and name it (e.g., fir_lab.m)†.
2. Set the order of the designed FIR filter.
3. Prepare the vector of normalised cutoff frequencies.
4. Create the appropriate window vector.
5. Calculate FIR filter $b$ coefficients using fir1.
Alternative 2 – Frequency Sampling based Technique

1. Create an *.m file and name it (e.g., fir_lab.m)
2. Set the order of the designed FIR filter.
3. Prepare the vector of desired normalised frequency points for the approximation.
4. Prepare the vector of desired magnitude response at desired normalised frequency points.
5. Calculate FIR filter $b$ coefficients using fir2.

Group C – Complementary Part

The complementary tasks in Group C focus on understanding of the digital systems theory and optimal FIR filter design.

Alternative 1 – FIR Filter Phase Response

1. Create an *.m file and name it (e.g., fir_lab_compl.m). Don’t start file name with number!
2. Extract and plot the impulse response of the FIR filter, designed in mandatory part.
3. Calculate the FFT† (see eq. (0.1)) of the FIR filter impulse response $h(n)$.
4. Calculate the natural logarithm† for each coefficient.
5. Extract the imaginary part† of the coefficients.
6. Correct phase angles to produce smoother phase plots using unwrap.
7. Convert radians into degrees.
8. Plot the phase response of the designed FIR filter.
9. Compare results with the output from impz and freqz functions.
2.4 Tasks for Laboratory Work

Alternative 2 – FIR Filter Impulse Response

1. Create an *.m file and name it (e.g., fir_lab_compl.m). Don’t start file name with number!
2. Extract and plot the impulse response of the FIR filter, designed in mandatory part.
3. Calculate the FFT† (see eq. (0.1)) of the FIR filter impulse response \( h(n) \).
4. Extract the real part† of the coefficients.
5. Convert magnitude into decibels (dB).
6. Plot the magnitude response of the designed FIR filter.
7. Compare results with the output from impz and freqz functions.

Alternative 3 – Constrained Least Square FIR Filter

1. Create an *.m file and name it (e.g., fir_lab_compl.m). Don’t start file name with number!
2. Prepare the normalised cutoff, passband edge and stopband edge frequencies.
3. Set the maximum passband deviation from 1 and maximum stopband deviation from 0.
4. Set the filter order \( n \).
5. Calculate filter coefficients using fircls1.
6. Plot the frequency response, phase response and impulse response of the designed FIR filter.

2.4 Tasks for Laboratory Work

Group A – FIR Filter Design using FDATool – Mandatory Part

Task: Design of the Low-Pass FIR Filter using FDATool

Using Filter Design and Analysis Tool (fdatool), calculate two low-pass FIR filters of the lowest filter order to meet the requirements for stopband attenuation of 60 dB. Frequency at the edge of the
passband: 1500 Hz. Frequency at the beginning of the stopband: 1600 Hz. For the filter design please use Gaussian and Kaizer windows.

### Task: Design of the Low-Pass FIR Filter using FDATool

Using Filter Design and Analysis Tool (fdatool), calculate two low-pass FIR filters of the lowest filter order to meet the requirements for stopband attenuation of 50 dB. Frequency at the edge of the passband: 120 Hz. Frequency at the beginning of the stopband: 180 Hz. For the filter design please use Hamming and Kaizer windows.

### Task: Design of the High-Pass FIR Filter using FDATool

Using Filter Design and Analysis Tool (fdatool), calculate two high-pass FIR filters of the lowest filter order to meet the requirements for stopband attenuation of 30 dB. Frequency at the edge of the passband: 650 Hz. Frequency at the beginning of the stopband: 520 Hz. For the filter design please use Bartlett and Kaizer windows.

### Task: Design of the High-Pass FIR Filter using FDATool

Using Filter Design and Analysis Tool (fdatool), calculate two high-pass FIR filters of the lowest filter order to meet the requirements for stopband attenuation of 40 dB. Frequency at the edge of the passband: 410 Hz. Frequency at the beginning of the stopband: 320 Hz. For the filter design please use Blackmann-Harris and Kaizer windows.

### Task: Design of the Band-Pass FIR Filter using FDATool

Using Filter Design and Analysis Tool (fdatool), calculate two band-pass FIR filters of the lowest filter order to meet the requirements for stopband attenuation of 60 dB at the lower frequencies and 50 dB at the upper frequencies. Frequencies at the edge of the passband: 1200 Hz and 1600 Hz. Frequencies at the edge of the stopband: 1070 Hz and 1700 Hz. For the filter design please use
2.4 Tasks for Laboratory Work

Gaussian and Kaizer windows.

**Task: Design of the Band-Pass FIR Filter using FDATool**

Using Filter Design and Analysis Tool (fdatool), calculate two band-pass FIR filters of the lowest filter order to meet the requirements for stopband attenuation of 40 dB at the lower frequencies and 50 dB at the upper frequencies. Frequencies at the edge of the passband: 320 Hz and 460 Hz. Frequencies at the edge of the stopband: 270 Hz and 510 Hz. For the filter design please use Hamming and Kaizer windows.

**Task: Design of the Band-Stop FIR Filter using FDATool**

Using Filter Design and Analysis Tool (fdatool), calculate two band-stop FIR filters of the lowest filter order to meet the requirements for stopband attenuation of 45 dB. Frequencies at the edge of the passband: 610 Hz and 1050 Hz. Frequencies at the edge of the stopband: 680 Hz and 940 Hz. For the filter design please use Bohman and Kaizer windows.

**Task: Design of the Band-Stop FIR Filter using FDATool**

Using Filter Design and Analysis Tool (fdatool), calculate two band-stop FIR filters of the lowest filter order to meet the requirements for stopband attenuation of 40 dB. Frequencies at the edge of the passband: 370 Hz and 790 Hz. Frequencies at the edge of the stopband: 500 Hz and 700 Hz. For the filter design please use Triangular and Kaizer windows.

**Group B – FIR Filter Design without GUI – Mandatory Part**

**Alternative 1 – Windowing Technique**

**Task: Design of the Band-Stop FIR Filter using fir1**

Create the band-stop FIR filter of the lowest filter order to meet the requirements for stopband attenuation of 45 dB. Frequencies at the edge of the passband: 610 Hz and 1050 Hz. Fre-
frequencies at the edge of the stopband: 680 Hz and 940 Hz. The filter should be created by using \texttt{fir1} function. Visualise the received results using Filter Visualisation Tool \texttt{fvtool}. For the filter design please use Bohman and Kaizer windows.

\section*{Task: Design of the Band-Pass FIR Filter \texttt{fir1}}

Create the band-pass FIR filter of the lowest order to meet the requirements for stopband attenuation of 60 dB at the lower frequencies and 50 dB at the upper frequencies. Frequencies at the edge of the passband: 1200 Hz and 1600 Hz. Frequencies at the edge of the stopband: 1070 Hz and 1700 Hz. The filter should be created by using \texttt{fir1} function. Visualise the received results using Filter Visualisation Tool \texttt{fvtool}. For the filter design please use Gaussian and Kaizer windows.

\section*{Alternative 2 – Frequency Sampling based Technique}

\section*{Task: Design of the Band-Stop FIR Filter using \texttt{fir2}}

Create the band-stop FIR filter of the lowest filter order to meet the requirements for stopband attenuation of 40 dB. Frequencies at the edge of the passband: 370 Hz and 790 Hz. Frequencies at the edge of the stopband: 500 Hz and 700 Hz. The filter should be created by using \texttt{fir2} function. Visualise the received results using Filter Visualisation Tool \texttt{fvtool}. For the filter design please use Triangular and Kaizer windows.

\section*{Task: Design of the Band-Pass FIR Filter using \texttt{fir2}}

Create the band-pass FIR filter of the lowest order to meet the requirements for stopband attenuation of 40 dB. Frequencies at the edge of the passband: 320 Hz and 460 Hz. Frequencies at the edge of the stopband: 270 Hz and 510 Hz. The filter should be created by using \texttt{fir2} function. Visualise the received results using Filter Visualisation Tool \texttt{fvtool}. For the filter design please use Hamming and Kaizer windows.
2.5 Questionnaire

**Group C – Complementary Part**

**Alternative 1 – FIR Filter Phase Response**

- **Task:** *Calculation of the FIR Filter Phase Response*

  Extract and visualise the impulse response and phase response of the FIR filter, designed in mandatory Part A using functions `fft`, `abs` and `imag`. Compare the visualisation results with the output from `impz` and `freqz` functions.

**Alternative 2 – FIR Filter Impulse Response**

- **Task:** *Calculation of the FIR Filter Magnitude Response*

  Extract and visualise the impulse response and magnitude response of the FIR filter, designed in mandatory Part A using functions `fft` and `abs`. Compare the visualisation results with the output from `impz` and `freqz` functions.

**Alternative 3 – Constrained Least Square FIR Filter**

- **Task:** *Design of the Constrained Least Square FIR Filter*

  Create the optimal FIR filter, with parameters given in mandatory Part A using function `fircls1`. Visualise the FIR filter design results with `impz` and `freqz` functions.

**2.5 Questionnaire**

Q1. What is the difference between FIR filter $b$ coefficients and impulse response, defined by $h$ coefficients?

Q2. What is the difference between band-pass and band-stop filters?

Q3. What is the desired frequency response of the system and how it is related to an impulse response?

Q4. What is the purpose of the window functions, used for FIR filter design?
Q5. There are various windows with different shape used for FIR filter design, why do not we use the simple rectangular window?

Q6. What is the difference between FIR filter design using windowing technique and frequency sampling technique?

Q7. What is the difference between FIR filter design using windowing techniques and optimal FIR filter design?

Q8. How the filter magnitude response is related to a frequency response of the filter?

**Bibliography**


DESIGN OF INFINITE IMPULSE RESPONSE FILTERS

3.1 Aim

The aim of this laboratory is to learn the basis of an Infinite Impulse Response (IIR) filter design accordingly to a given filter specification.

3.2 Important Material

In this laboratory a digital IIR filter, having infinite response to a unit sample is analysed. The design of IIR filter is made by converting a pre-designed continuous time analog filter into digital one.

3.2.1 Description of the Analog Filter

The analog filter can be specified by its system function (Proakis, Manolakis 2006: Chapter 10)

\[
G(s) = \frac{B(s)}{A(s)} = \frac{\sum_{k=0}^{M} b_{ak} s^k}{\sum_{k=0}^{N} a_{ak} s^k}; \quad (3.1)
\]

here \( b_{ak} \) and \( a_{ak} \) are the analog filter coefficients.

The impulse response \( g(t) \) of a continuous time analog filter is related to the frequency response \( G(s) \) of this filter by a Laplace
transform (Proakis, Manolakis 2006: Chapter 10)

\[ G(s) = \int_{-\infty}^{\infty} g(t) e^{-st} dt. \]  

(3.2)

### 3.2.2 IIR Filter Design by Impulse Invariance

If the analog filter is a band-limited continuous time system, such as low-pass or band-pass analog filter, the digital filter can be designed by sampling analog filter impulse response (Oppenheim, Schafer 2009: Chapter 7).

Frequency response \( H(z) \) of the digital IIR filter can be easily expressed if equation (3.2) can be divided into the partial-fraction form (Proakis, Manolakis 2006: Chapter 10)

\[ G(s) = \sum_{k=1}^{N} \frac{c_k}{s - p_k}; \]  

(3.3)

here \( p_k \) are the poles of analog filter, \( c_k \) are the coefficients, received performing the partial-fraction expansion from (3.2) and \( N \) is the number of analog filter poles. The partial-fraction form of the digital IIR filter frequency response is

\[ H(z) = \sum_{k=1}^{N} \frac{c_k}{1 - e^{p_k T} z^{-1}}. \]  

(3.4)

### 3.2.3 IIR Filter Design by Bilinear Transform

Bilinear transform of analog filter have less limitations and is not limited only to low-pass and band-pass IIR filter design. The mapping of the s-plane to z-plane is expressed by (Proakis, Manolakis 2006: Chapter 10)

\[ s = \frac{2}{T} \left( \frac{1 - z^{-1}}{1 + z^{-1}} \right). \]  

(3.5)

For the bilinear transform, the mapping of the analog fre-
Important Material

3.2.4 IIR Filter Design using MATLAB

Two MATLAB™ toolboxes support design of the digital IIR filters: Signal Processing Toolbox (The MathWorks, Inc. 2010b) and DSP system Toolbox (The MathWorks, Inc. 2010a).

Example 3.1 Digital Filter Design from Analog Filter Prototype

Create a 4th order analog Butterworth low-pass filter. The passband cutoff frequency should be 120 Hz. The stopband cutoff frequency should be 140 Hz. Transform the created analog Butterworth filter into digital using bilinear transformation. The sampling frequency of the digital filter should be equal to 1000 Hz.

```
1 % Design of analog filter prototype
2 [z, p, k] = buttap(4);
3 % Evaluation of the filter coefficients
4 [b_1, a_1] = zp2tf(z, p, k);
5 % Passband frequency (angular) evaluation
6 f_p = 120;
7 w_p = 2*pi*f_p;
8 % Evaluation of the filter coefficients w_p
9 [b_a, a_a] = lp2lp(b_1, a_1, w_p)
10 fs = 1000; % sampling frequency (Hz)
11 % Bilinear transformation of the analog filter
12 [b_dg, a_dg] = bilinear(b_a, a_a, fs, f_p)
```

The evaluated digital filter coefficients are stored in MATLAB™ Workspace and can be shown in Command Window:

```
b_a =
    3.2318e+011
a_a =
    1.0e+011 *
         0.0000    0.0000    0.0000    0.0112    3.2318
b_dg =
    0.0087    0.0357    0.0535    0.0357    0.0089
a_dg =
```
Evaluated filter coefficients can be written into the system functions $G(s)$ (analog filter) and $H(z)$ (digital filter):

$$G(s) = \frac{3.2318}{s^4 + 0.0112s + 3.2318} = \frac{3.2318}{0.0112s + 3.2318};$$

$$H(z) = \frac{0.0087 + 0.0357z^{-1} + 0.0535z^{-2} + 0.0357z^{-3} + 0.0087z^{-4}}{1 - 2.0484z^{-1} + 1.8418z^{-2} - 0.7824z^{-3} + 0.1317z^{-4}}.$$

The Signal Processing Toolbox also has several advanced functions for IIR filter design: `butter`, `cheby1`, `cheby2`, `ellip` (Navakauskas, Serackis 2008: Chapter 9).

**Example 3.2 Design of the Elliptic Band-Stop Filter**

Create an Elliptic band-stop filter with passband cutoff frequencies 220 Hz and 380 Hz. The desired stopband attenuation should not exceed 40 dB in the range [280; 320].

First, the normalised frequencies (1 is equal to Nyquist frequency) should be calculated:

```matlab
f_s = 1000; % Sampling frequency
% Passband cutoff frequencies
W_p = [2*220/f_s 2*380/f_s];
% Stopband cutoff frequencies
W_st = [2*280/f_s 2*320/f_s];
R_st = 40; % Stopband attenuation in dB
R_p = 3;  % Passband attenuation in dB
```

Second, the minimum Elliptic filter order should be evaluated accordingly to a given cutoff frequencies and desired attenuation:

```matlab
[n, W_n] = ellipord(W_p, W_st, R_p, R_st);
```

Next, the digital filter coefficients are evaluated and Magnitude response is drawn:

```matlab
[b_dg, a_dg] = ellip(n, R_p, R_st, W_n, 'stop');
freqz(b_dg, a_dg, 1024, f_s);
```
3.3 Procedure

3.3.1 Software Tools and Signal Sources

A subset of MATLAB™ software package functions in the following groups – elementary matrices and arrays, array manipulation, math constants, basic functions, Fourier transforms, filtering and convolution, control flow, handle graphics – as well as some functions from Signal Processing Toolbox (The MathWorks, Inc. 2010b), DSP System Toolbox (The MathWorks, Inc. 2010a) need to be used.

3.3.2 Work Order

The laboratory work consists of mandatory and complementary parts. Mandatory part (A and B group) tasks need to be done first (sequentially, in declared order). Only if time will permit the tasks from complementary part (C group) should be approach.

As some work group tasks have alternative work orders, please make sure that you follow the correct one work description.

Exact task numbers (from A, B and C groups) will be given individually in the class by laboratory work supervisor.

Group A – IIR Filter Design using FDATool – Mandatory Part

This mandatory part of the laboratory is dedicated to learn how to use the innovative MATLAB™ tools for IIR filter design. The first group of tasks (Group A) helps to get familiar with the graphical user interface (GUI) for digital IIR filter design found in DSP System Toolbox. The Group B focuses on MATLAB™ Signal Processing Toolbox functions for IIR filter design using classical methods, such as analog filter transformation to digital filter using Bilinear Transformation and Impulse Invariance method. The complementary tasks in Group C focus on deep understanding of the digital IIR filter design.

1. Start the Filter Design and Analysis Tool†.
2. Select the desired response type of the filter, accordingly to the task.
3. Select the appropriate IIR filter design method.

4. Choose Minimum order option in Filter Order section.

5. Set the appropriate sampling (Fs), passband edge (Fpass or Fpass1 and Fpass2) and stopband edge (Fstop and Fstop1 and Fstop2) frequencies.

6. Set the desired attenuation at the passband (Apass or Apass1 and Apass2) and stopband (Astop or Astop1 and Astop2) attenuation.

**Group B – IIR Filter Design without GUI – Mandatory Part**

This mandatory part of the laboratory is dedicated to learn how to use MATLAB™ functions `butter`, `cheby1`, `cheby2` and `ellip` to design the digital IIR filter with desired frequency specifications.

1. Create an * .m file and name it (e.g., `iir_lab.m`).

2. Prepare the vector of normalised cutoff frequencies.

3. Calculate the order for the designed IIR filter using specialised MATLAB™ function.

4. Calculate the transfer function for the designed digital filter.

5. Draw the impulse response, magnitude response and phase response of the filter using Filter Visualization Tool.

**Group C – Complementary Part**

**Alternative 1 – Design by Impulse Invariance**

1. Create an * .m file and name it (e.g., `iir_lab_compl.m`).

2. Create an analog low-pass filter prototype with a cutoff frequency of 1 rad/s.

3. Transform analog low-pass filter prototype to analog low-pass (or band-pass) filter with desired cutoff angular frequency.

4. Use impulse invariance method to transform analog low-
pass (or band-pass) filter to a digital one with desired frequency response.

5. Calculate the impulse response $h(n)$ of the designed IIR filter from its system function $H(z)$ and visualise it.

6. Calculate and visualise the magnitude and phase response of the designed low-pass (or band-pass) digital IIR filter.

**Alternative 2 – Least-Squares Fit Method**

1. Create an *.m file and name it (e.g., iir_lab_compl.m).
2. Prepare the vector of desired normalised frequency points for the approximation.
3. Prepare the vector of desired magnitude response at desired normalised frequency points.
4. Set the filter order $n$.
5. Calculate $b$ and $a$ IIR filter coefficients using *yulewalk* MATLAB™ function.
6. Plot the magnitude response, phase response and impulse response of the designed IIR filter.

**3.4 Tasks for Laboratory Work**

**Group A – IIR Filter Design using FDATool – Mandatory Part**

**Task: Design of the Low-Pass Butterworth IIR Filter**

Using Filter Design and Analysis Tool (fdatool), calculate a low-pass Butterworth IIR filter of the lowest filter order to meet the requirements for stopband attenuation of 60 dB. Frequency at the edge of the passband: 320 Hz. Frequency at the beginning of the stopband: 400 Hz.

**Task: Design of the Low-Pass Chebyshev Type 2 IIR Filter**

Using Filter Design and Analysis Tool (fdatool), calculate a low-pass Chebyshev Type 2 IIR filter of the lowest filter order to meet the requirements for stopband attenuation of 50 dB. Frequency
at the edge of the passband: 820 Hz. Frequency at the beginning of the stopband: 950 Hz.

**Task: Design of the High-Pass Elliptic IIR Filter**

Using Filter Design and Analysis Tool (fdatool), calculate a high-pass Elliptic IIR filter of the lowest filter order to meet the requirements for stopband attenuation of 30 dB. Frequency at the edge of the passband: 650 Hz. Frequency at the beginning of the stopband: 600 Hz.

**Task: Design of the High-Pass Chebyshev Type 2 IIR Filter**

Using Filter Design and Analysis Tool (fdatool), calculate a high-pass Chebyshev Type 2 IIR filter of the lowest filter order to meet the requirements for stopband attenuation of 40 dB. Frequency at the edge of the passband: 410 Hz. Frequency at the beginning of the stopband: 350 Hz.

**Task: Design of the Band-Pass Elliptic IIR Filter**

Using Filter Design and Analysis Tool (fdatool), calculate a band-pass Elliptic IIR filter of the lowest filter order to meet the requirements for stopband attenuation of 40 dB. Frequencies at the edge of the passband: 570 Hz and 880 Hz. Frequencies at the edge of the stopband: 500 Hz and 950 Hz.

**Task: Design of the Band-Pass Chebyshev Type 2 IIR Filter**

Using Filter Design and Analysis Tool (fdatool), calculate a band-pass Chebyshev Type 2 IIR filter of the lowest filter order to meet the requirements for stopband attenuation of 45 dB. Frequencies at the edge of the passband: 620 Hz and 860 Hz. Frequencies at the edge of the stopband: 570 Hz and 1010 Hz.

**Task: Design of the Band-Stop Chebyshev Type 1 IIR Filter**

Using Filter Design and Analysis Tool (fdatool), calculate a band-stop Chebyshev Type 1 IIR filter of the lowest filter order to meet
the requirements for stopband attenuation of 45 dB. Frequencies at the edge of the passband: 610 Hz and 1050 Hz. Frequencies at the edge of the stopband: 680 Hz and 940 Hz.

**Task: Design of the Band-Stop Elliptic IIR Filter**

Using Filter Design and Analysis Tool (fdatool), calculate a band-stop Elliptic IIR filter of the lowest filter order to meet the requirements for stopband attenuation of 60 dB. Frequencies at the edge of the passband: 470 Hz and 890 Hz. Frequencies at the edge of the stopband: 550 Hz and 710 Hz.

**Group B – IIR Filter Design without GUI – Mandatory Part**

**Task: Design of the Band-Stop Butterworth IIR Filter**

Create the band-stop Butterworth IIR filter of the lowest filter order to meet the requirements for stopband attenuation of 45 dB. Frequencies at the edge of the passband: 310 Hz and 580 Hz. Frequencies at the edge of the stopband: 230 Hz and 640 Hz. The filter should be created by using buttord, butter MATLAB™ functions. Visualise the received results using Filter Visualisation Tool fvtool.

**Task: Design of the Band-Pass Chebyshev Type 1 IIR Filter**

Create the band-pass Chebyshev Type 1 IIR filter of the lowest order to meet the requirements for stopband attenuation of 50 dB. Frequencies at the edge of the passband: 900 Hz and 1300 Hz. Frequencies at the edge of the stopband: 770 Hz and 1420 Hz. The filter should be created by using cheblord, cheby1 MATLAB™ functions. Visualise the received results using Filter Visualisation Tool fvtool.

**Task: Design of the Band-Pass Chebyshev Type 2 IIR Filter**

Create the band-pass Chebyshev Type 2 IIR filter of the lowest order to meet the requirements for stopband attenuation of 60 dB. Frequencies at the edge of the passband: 900 Hz and 1300 Hz.
Frequencies at the edge of the stopband: 770 Hz and 1420 Hz. The filter should be created by using `cheb2ord, cheby2` MATLAB™ functions. Visualise the received results using Filter Visualisation Tool `fvtool`.

---

### Task: Design of the Band-Stop Elliptic IIR Filter

Create the band-stop Elliptic IIR filter of the lowest filter order to meet the requirements for stopband attenuation of 40 dB. Frequencies at the edge of the passband: 370 Hz and 790 Hz. Frequencies at the edge of the stopband: 500 Hz and 700 Hz. The filter should be created by using `ellipord, ellip` MATLAB™ functions. Visualise the received results using Filter Visualisation Tool `fvtool`.

---

### Task: Design of the Band-Pass Butterworth IIR Filter

Create the band-pass Butterworth IIR filter of the lowest order to meet the requirements for stopband attenuation of 40 dB. Frequencies at the edge of the passband: 320 Hz and 460 Hz. Frequencies at the edge of the stopband: 270 Hz and 510 Hz. The filter should be created by using `buttord, butter` MATLAB™ functions. Visualise the received results using Filter Visualisation Tool `fvtool`.

---

### Group C – Complementary Part

**Alternative 1 – Design by Impulse Invariance**

### Task: Design of the Low-Pass IIR Filter

Design a low-pass Butterworth IIR filter, with parameters given in 3.1 Task, using impulse invariance method† of the analog filter prototype, created by `buttap` MATLAB™ function. Visualise the impulse response, magnitude and phase response of the designed IIR filter using `impz` and `freqz` MATLAB™ functions.
3.5 Questionnaire

Q1. What is the difference between FIR and IIR filter?

Q2. What is the difference between analog and digital Butterworth filter?

Q3. What is the difference between Butterworth, Chebyshev and Elliptic filters? Which one is the best?

Q4. What is the purpose of \texttt{lp2lp} function?

Q5. What is the difference between $G(s)$ and $H(z)$?

Q6. Why, for Bilinear transform method, the analog filter frequencies are related to the digital filter frequencies through a non-linear equation (3.6)?

Q7. Can the Impulse Invariance method be applied for bandstop digital filter design?

Q8. What is the relation between analog filter and digital filter frequencies used for impulse invariance method?
Bibliography


4.1 The Aim

The aim of this laboratory is to learn the basics of a digital IIR filter design and realisations using various structures and forms.

4.2 Important Material

**Related MATLAB™ commands and functions**

| butter, buttord, dfilt, dfilt.cascade, dfilt.df1, dfilt.parallel, fdatool, fvtool, residuez, tf2sos, tf2zp, zp2sos, zplane |

Different realisations of the digital filters are equivalent mathematically, but usually have different performance when they are implemented in practice (Sen M. Kuo, Tian 2006: Chapter 5).

4.2.1 Realisation of the Digital Filters

MATLAB™ stores all digital filter parameters in double precision floating point format. Most of the digital signal processing units used today use fixed point arithmetic and always have finite word length. Applying digital filter to an input, signal values are multiplied by filter coefficients. Result of multiplication is always of double precision. To store this value in fixed point memory unit, the rounding should be applied. It usually leads to a quantisation errors, which influence to the whole system depends on the structure of the digital filter.

4.2.2 Direct Forms of the Digital Filters

The Direct Form I realisation of the digital IIR filter is defined by an input/output equation (Sen M. Kuo, Tian 2006: Chapter 5)
Graphical representation of the digital filter realisations

\[
y(n) = \sum_{k=0}^{M-1} b_k x(n - k) + \sum_{l=1}^{N-1} a_l y(n - l); \quad (4.1)
\]

here \( M \) and \( N \) are the number of digital filter numerators and denominators respectively.

Graphical representation of the filter realised in Direct Form I is shown in Figure 4.1, a.

As it is seen in Figure 4.1, a, the Direct Form I realisation requires \((M + N)\) multiplications and \((M + N - 1)\) additions.

The Direct Form II realisation of the digital IIR filter is received by interpreting the filter transfer function \( H(z) \) as the cascade of two transfer functions (Sen M. Kuo, Tian 2006: Chapter 5)

\[
H(z) = H_1(z) H_2(z) = \frac{1}{1 - \sum_{l=1}^{N} a_l z^{-l}} \sum_{k=0}^{M-1} b_k z^{-k}. \quad (4.2)
\]

Graphical representation of the filter realised in Direct Form II is shown in Figure 4.1, b.

As it is seen in Figure 4.1, b, the Direct Form II realisation requires the same number of multiplications and additions but up to two times less memory locations.
4.2 Important Material

4.2.3 Cascade Forms of the Digital Filters

The influence of the quantisation effects to whole system performance can be reduced by using a cascade combination of several low order filter sections instead one high order filter section. Mathematically it can be expressed as (Sen M. Kuo, Tian 2006: Chapter 5), (Oppenheim, Schafer 2009: Chapter 6)

\[ H(z) = b_0 \prod_{k=1}^{K} H_k(z) = b_0 \frac{\prod_{k=1}^{M} (1 - q_k z^{-1})}{\prod_{l=1}^{N} (1 - p_l z^{-1})}. \]  (4.3)

The cascade combination of two (high-pass and low-pass) IIR filters should lead to a band-pass filter if the cutoff frequency of the low-pass filter is higher than the cutoff frequency of the high-pass filter.

4.2.4 Parallel Forms of the Digital Filters

A partial fraction expansion of the IIR filter leads to another – Parallel Form – of the digital filter. Mathematically it is expressed as (Sen M. Kuo, Tian 2006: Chapter 5), (Oppenheim, Schafer 2009: Chapter 6)

\[ H(z) = \sum_{k=1}^{N_p} C_k z^{-k} + \sum_{k=1}^{N_1} \frac{A_k}{1 - c_k z^{-1}} + \sum_{k=1}^{N_2} \frac{B_k (1 - e_k z^{-1})}{(1 - d_k z^{-1})(1 - d_k^* z^{-1})}. \]  (4.4)

The parallel combination of two (high-pass and low-pass) IIR filters should lead to a band-stop filter if the cutoff frequency of the low-pass filter is lower than the cutoff frequency of the high-pass filter.

4.2.5 Realisation of the Digital Filters using MATLAB

To implement designed IIR filter in Cascade Form, the roots of two polynomials (numerator and denominator) should be calculated. This can be easily done using MATLAB\textsuperscript{TM} roots function. As an alternative, the Signal Processing Toolbox function tf2zp (The
MathWorks, Inc. 2010b) can be used to zeros, poles and gain from IIR filter transfer function coefficients:

```matlab
[z, p, c] = tf2zp(b, a);
```

Calculated zeros and poles can be combined to form various low order filter sections. To design a Cascade Form filter using second order filter sections, MATLAB™ mComzp2sos or tf2sos function can be used:

```matlab
[sos, G] = zp2sos(z, p, c);
```

To implement designed IIR filter in Parallel Form, residuez function can be used:

```matlab
[r, p, c] = residuez(b, a);
```

residuez function returns residues \( r \) \( (A_k \) coefficients in (4.4) equation), poles \( p \) \( (c_k \) coefficients in (4.4) equation) and direct terms \( k \) \( (C_k \) coefficients in (4.4) equation) of the partial fraction expansion of filter transfer function.

Band-pass, band-stop and multiband digital filters can be designed by using a cascade or parallel combination of the low-pass and high-pass filters. MATLAB™ dfilt functions for discrete-time filter design can be used for the design of various filter structures.

### Example 4.1 Multiband filter design

In this example a multiband IIR filter with two band-pass sections will be designed by using MATLAB™ dfilt tools. Design a two band multiband IIR filter with passband from 300 Hz to 450 Hz and from 1200 Hz to 1350 Hz by using two low-pass and two band-pass Butterworth IIR filters.

Let us prepare the normalised frequencies for the low-pass and high-pass filters and select a 50 dB ripple at the stop-bands:

```matlab
Fs = 3000; % Sampling freq. is more than twice higher than 1350
% Normalisation is performed using Nyquist frequency = 0.5*Fs
Wp11 = 300/1500;
Wp12 = 450/1500;
Wp21 = 1200/1500;
Wp22 = 1350/1500;
```
4.2 Important Material

For IIR filter design, let us select the frequencies at the edges of the stop-band:

\[
\begin{align*}
W_{s11} &= \frac{250}{1500}; \\
W_{s12} &= \frac{500}{1500}; \\
W_{s21} &= \frac{1150}{1500}; \\
W_{s22} &= \frac{1400}{1500}; \\
\end{align*}
\]

It is important to select appropriate filter order to meet the requirements of the stopband attenuation:

\[
\begin{align*}
[n_{11}, W_{n11}] &= \text{buttord}(W_{s11}, W_{p11}, Rp, Rs); \quad \text{for high-pass filter} \\
[n_{12}, W_{n12}] &= \text{buttord}(W_{p12}, W_{s12}, Rp, Rs); \quad \text{for low-pass filter} \\
[n_{21}, W_{n21}] &= \text{buttord}(W_{s21}, W_{p21}, Rp, Rs); \quad \text{for high-pass filter} \\
[n_{22}, W_{n22}] &= \text{buttord}(W_{p22}, W_{s22}, Rp, Rs); \quad \text{for low-pass filter} \\
\end{align*}
\]

Now the Butterworth IIR filter can be designed and realised into Direct Form:

\[
\begin{align*}
[b_{11}, a_{11}] &= \text{butter}(n_{11}, W_{n11}, \text{'high'}); \quad \text{high-pass filter} \\
[b_{12}, a_{12}] &= \text{butter}(n_{12}, W_{n12}); \quad \text{low-pass filter} \\
[b_{21}, a_{21}] &= \text{butter}(n_{21}, W_{n21}, \text{'high'}); \quad \text{high-pass filter} \\
[b_{22}, a_{22}] &= \text{butter}(n_{22}, W_{n22}); \quad \text{low-pass filter} \\
\end{align*}
\]

A band-pass filter can be created by a cascade combination of high-pass and low-pass filters:

\[
\begin{align*}
H_{d11} &= \text{dfilt.d1}(b_{11}, a_{11}); \\
H_{d12} &= \text{dfilt.d1}(b_{12}, a_{12}); \\
H_{d21} &= \text{dfilt.d1}(b_{21}, a_{21}); \\
H_{d22} &= \text{dfilt.d1}(b_{22}, a_{22}); \\
\end{align*}
\]

A filter with two passband can be created by a parallel combination of two band-pass filters:

\[
\begin{align*}
H_{d_{bp1}} &= \text{dfilt.cascade}(H_{d11}, H_{d12}); \\
H_{d_{bp2}} &= \text{dfilt.cascade}(H_{d21}, H_{d22}); \\
\end{align*}
\]

The visualisation of the designed and implemented multiband IIR filter is performed using Filter Visualization Tool:

\[
\text{fvtool}(H_{d_{mb}})
\]
4.3 Procedure

4.3.1 Software Tools and Signal Sources

A subset of MATLAB™ software package functions in the following groups – array manipulation, math constants, basic functions, Fourier transforms, filtering and convolution, control flow, handle graphics – as well as some functions from Signal Processing Toolbox (The MathWorks, Inc. 2010b), DSP System Toolbox (The MathWorks, Inc. 2010a) need to be used.

4.3.2 Work Order

The laboratory work consists of mandatory and complementary parts. Mandatory part (A and B group) tasks need to be done first (sequentialy, in declared order). Only if time will permit the tasks from complementary part (C group) should be approach.

As some work group tasks have alternative work orders, please make sure that you follow the correct one work description.

Exact task numbers (from A, B and C groups) will be given individually in the class by laboratory work supervisor.

Group A – IIR Filter Realisation – Mandatory Part

1. Start the Filter Design and Analysis Tool†.
2. Select the appropriate response type of the filter.
3. Select the appropriate IIR filter design method.
4. Select the Minimum order option in Filter Order.
5. Set the appropriate sampling (Fs), passband (Fstop) and stopband (Fpass) frequencies.
6. Design filter and export† filter data into Workspace.
7. Change digital filter structure† into Single Section.
8. Export† the numerator and denominator† coefficients of the designed filter into MATLAB™ Workspace.
9. Create an *.m file and name it (e.g., struct_lab_a.m).
4.4 Tasks for Laboratory Work

Group B – Multiband IIR Filter Realisation – Mandatory Part

1. Create an *.m file and name it (e.g., struct_lab_b.m).
2. Normalise the passband and stopband frequencies.
3. Analyse the given task and design the low-pass and high-pass filters, appropriate for given multiband filter design.
4. Calculate numerator and denominator for minimum order low-pass and high-pass filters of type, given in the task.
5. Create a discrete-time filter of the form, given in the task and combine them to form a desired multiband filter.
6. Visualise the filter design result using Filter Visualization Tool.

Group C – Complementary Part

1. Create an *.m file and name it (e.g., struct_lab_c.m).
2. Carefully read the given task.
3. Read the additional literature, to find the relations between the task and digital filter design theory.
4. Solve given task using previously used and additional functions (if needed).

4.4 Tasks for Laboratory Work

Group A – IIR Filter Realisation – Mandatory Part

Task: Design of the Butterworth Band-Pass Filter

Using MATLAB™ Filter Design and Analysis Tool design a minimum order Butterworth band-pass IIR filter with passband from 660 Hz to 980 Hz. The attenuation at the stopband should be 50 dB for frequencies up to 630 Hz and less than 40 dB for frequencies higher than 1 kHz. Implement the designed filter using Second-Order Sections and compare the frequency response with the same filter, implemented in Single Section. Visualise the Pole/Zero plot for single-section filter realisation.
Task: **Design of the Chebyshev Type I Band-Pass Filter**

Using **MATLAB** Filter Design and Analysis Tool design a minimum order Butterworth band-pass IIR filter with passband from 16 kHz to 18 kHz. The attenuation at the stopband should be less than 50 dB for frequencies up to 15.8 kHz and less than 40 dB for frequencies higher than 18.3 kHz. Implement the designed filter using Second-Order Sections and compare the frequency response with the same filter, implemented in Single Section. Visualise the Pole/Zero plot for single-section filter realisation.

Task: **Design of the Elliptic Band-Stop Filter**

Using **MATLAB** Filter Design and Analysis Tool design a minimum order Elliptic band-stop IIR filter with stopband from 230 Hz to 510 Hz. The attenuation at the stopband should be 50 dB. The passband edge frequencies for designed band-stop filter should be equal to 200 Hz and 550 Hz. Implement the designed filter using Second-Order Sections and compare the frequency response with the same filter, implemented in Single Section. Visualise the Pole/Zero plot for single-section filter realisation.

Task: **Design of the Chebyshev Type II Band-Stop Filter**

Using **MATLAB** Filter Design and Analysis Tool design a minimum order Elliptic band-stop IIR filter with stopband from 48 MHz to 53 MHz. The attenuation at the stopband should be 50 dB. The passband edge frequencies for designed band-stop filter should be equal to 35 MHz and 59 MHz. Implement the designed filter using Second-Order Sections and compare the frequency response with the same filter, implemented in Single Section. Visualise the Pole/Zero plot for single-section filter realisation.

Group B – Design of the Multiband IIR Filter – Mandatory Part

Task: **Design of the Multiband IIR Filter with Two Passbands**

Using low-pass and high-pass Chebyshev Type I IIR filters, design a multiband filter with two passbands:
4.4 Tasks for Laboratory Work

✓ from 592 MHz to 604 MHz;
✓ from 776 MHz to 788 MHz.

The attenuation at the stopband should be 35 dB for frequencies less than 588 MHz, from 608 MHz to 772 MHz and higher than 792 MHz. Use the lower order filter that meets the given requirements. Implement the low-pass and high-pass filters using Direct Form Type II Second-Order Sections (dfilt.df2sos) and combine them into cascade and parallel form to design a multiband filter. Visualise the magnitude response of the designed multiband IIR filter.

**Task: Design of the Multiband IIR Filter with Two Stopbands**

Using low-pass and high-pass Elliptic IIR filters, design a multiband filter with two stopbands:
✓ from 674 MHz to 682 MHz;
✓ from 826 MHz to 832 MHz.

The attenuation at the both stopbands should be 30 dB. Passbands with attenuation of 3 dB and less should be at frequencies less than 670 MHz, from 686 MHz to 822 MHz and higher than 836 MHz. Use the lower order filter that meets the given requirements. Implement the low-pass and high-pass filters using Direct Form Type I Second-Order Sections (dfilt.df1sos) and combine them into cascade and parallel form to design a multiband filter. Visualise the magnitude response of the designed multiband IIR filter.

**Group C – Complementary Part**

**Task: Design of the IIR Filter using Different Order Sections**

For given partial-fraction expansion of the system function

\[
H(z) = \frac{0.001(0.2196 - 0.6588z^{-2} + 0.6588z^{-4} - 0.2196z^{-6})}{1 + 2.7488z^{-2} + 2.5282z^{-4} + 0.7776z^{-6}} + \frac{0.0009(0.2196 - 0.6588z^{-2} + 0.6588z^{-4} - 0.2196z^{-6})}{1 - 3.3858z^{-1} + 6.5733z^{-2} - 7.657z^{-3} + 6.0448z^{-4} - 2.863z^{-5} + 0.7776z^{-6}}
\]

(4.5)
calculate filter frequency response. Create the digital filter with
the same frequency characteristics using parallel combination of the second order filter and fourth order filter sections. Visualise and compare magnitude response of these two filter designs.

**Task: Design of the Multiband FIR Filter**

Using the same requirements for passband, stopband edge frequencies and stopband attenuation, given in Mandatory Part (in Group B), design a multiband FIR filter. Compare the frequency response of IIR and FIR filters.

### 4.5 Questionnaire

Q1. What is the difference between the Direct Form I and Direct Form II filter realisations?

Q2. What is the difference between the Direct Form and Cascade Form filter realisations? Which one is better? Why?

Q3. What is the difference between the Direct Form and Parallel Form filter realisations? Which one is better? Why?

Q4. In which situations we should use Cascade realisation of the digital filter and in which situation – Parallel?

Q5. What is the multiband digital filter?

Q6. What is the difference between the multiband FIR and multiband IIR digital filter?

### Bibliography


FILTERING OF AUDIO SIGNALS

5.1 The Aim

The aim of this laboratory is to learn how to design a digital filter for solving practical problems and acquire cognitions to perform audio signal analysis and processing tasks.

5.2 Important Material

<table>
<thead>
<tr>
<th>Related MATLAB™ commands and functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs, decimate, demod, design, dfilt.cascade, fdesign.bandstop, filter, fir1, firls, interp, modulate, plot, resample, stem, upfirdn, wavread.</td>
</tr>
</tbody>
</table>

Three DSP application areas are adverted in this laboratory: audio signal sampling rate conversion, time-invariant noise filtering and modulation/demodulation for the digital audio signal for communication systems.

5.2.1 Changing the Audio Signal Sampling Rate

The sampling rate of the audio signal should be adjusted accordingly to a digital signal processing system sampling rate. Depending on the application, the resampling of the digital signal is performed using interpolation or decimation procedure.

Interpolation techniques are used to increase the sampling rate of the digital signal. The harmonic content of the digital signal should be preserved during interpolation – the highest frequency component of the interpolated signal should not exceed the Nyquist frequency of the initial signal. The limitation of the frequency is usually made by using a low-pass FIR filter.

Decimation is used to decrease the sampling rate of the digital signal. If the sampling rate of the signal should be decreased by an integer number $N$ of times, we can just take every $N^{th}$ sample of the digital signal. In case, the sampling rate should be decreased by a non-integer number, the low-pass filtering (using
IIR or FIR filter) – smoothing of the signal – and new sampling using desired frequency is used. For example, the decimation by approximately 2.19 times is needed to convert an audio record, made using 96 kHz recorder sampling frequency, to an audio signal with sampling rate of 44.1 kHz, suitable for Audio-CD recording.

5.2.2 Filtering of the Time-Invariant Noise in Audio Signal

Time-invariant noise is the noise, which frequency characteristics do not change in time. The analysis of the digital signal in frequency domain helps to understand the properties of the filter, needed to remove a narrow bandwidth noise component.

Example 5.1 Noise Filtering in Audio Signal

An audio signal is affected by a 1.2 kHz noise. Remove this noise using digital filtering techniques.

The noise appears in the spectrum of the corrupted audio signal as additional frequency component at 1.2 kHz. To remove this noise component, it is needed to design a band-stop digital filter with a minimum attenuation at the frequencies around the 1.2 Hz.

5.2.3 Transmission and Reception of the Audio Signals

To transmit an audio signal over a communication system, the signal (or any message) should be modulated using one of the modulation techniques, that suits best for selected communication system and message type. There are various modulation techniques used for audio signal transmission: Amplitude Modulation (AM), Frequency Modulation (FM), Phase Modulation (PM), Pulse-Width Modulation (PWM), Pulse-Position Modulation (PPM), Quadrature Amplitude Modulation (QAM), etc.

The type of signal modulation usually depends on a transmission line properties and amount of data, that needs to be transmitted. For digital audio and video signals, various data compression techniques are applied before signal modulation.
5.2.4 Audio Signal Processing using MATLAB

There are several MATLAB™ functions for multirate signal processing. The interpolation can be performed using `interp` function. This function expands the original digital signal by inserting zeros between signal magnitude values and applies the specially designed FIR filter to minimise the mean square error between the interpolated points and their ideal values. The original data should pass through this filter unchanged.

The decimation of the digital signal can be performed using `decimate` function. By default it uses a Chebyshev Type I IIR low-pass filter with cutoff frequency equal to $0.8/r$. Here $r$ is a decimation factor by which the sampling rate of the signal should be reduced.

There is a possibility to use a low-pass FIR filter instead of IIR. This can be set by additional option `'fir'` in the input of `decimate` function. The low-pass FIR filter is designed using Windowing Techniques (`fir1` function).

The resampling of the digital signal by a rational factor $p/q$ also can be performed using MATLAB™ `resample` function. It uses polyphase filter implementations and applies a low-pass FIR filter for anti-aliasing. The FIR filter is designed using a linear-phase FIR filter (`firls` function) that minimises the weighted, integrated squared error between an ideal piecewise linear function and the magnitude response of the filter over a set of desired frequency bands (The MathWorks, Inc. 2010b). For resampling of the digital signal, `resample` use `firls` function with Kaiser window for truncation of the impulse response.

5.3 Procedure

5.3.1 Software Tools and Signal Sources

A subset of MATLAB™ software package functions (Navakauskas, Serackis 2008) in the following groups – elementary math, math constants, elementary matrices and arrays, basic functions, rounding and remainder, Fourier transforms, filtering and convolution, control flow, handle graphics, audio recording and play-
back – as well as some functions from Signal Processing Toolbox (The MathWorks, Inc. 2010b), Filter Design Toolbox or DSP Systems Toolbox (The MathWorks, Inc. 2010a) need to be used.

Signal data have to be downloaded from the course Web-site.

5.3.2 Work Order

The laboratory work consists of mandatory and complementary parts. Mandatory part (A and B group) tasks need to be done first (sequentialy, in declared order). Only if time will permit the tasks from complementary part (C group) should be approach.

As some work group tasks have alternative work orders, please make sure that you follow the correct one work description.

Exact task numbers (from A, B and C groups) will be given individually in the class by laboratory work supervisor.

Group A – Resampling of the Audio Signal – Mandatory Part

1. Create an *.m file and name it (e.g., fas_lab_a.m).
2. Read the samples of the audio file and it’s sampling frequency into MATLAB™ workspace using wavread.
3. For stereo audio file extract left and right audio channel into two vectors (e.g., audioL = audio_file(:,1)) and process them separately.
4. Select an appropriate MATLAB™ function and it’s input parameters to change the sampling rate of the audio file accordingly to the given task requirements.
5. Calculate the Fourier transform of the initial and resampled audio signal.
6. Plot the spectrum (abs(fft_result)) of the audio file† using stem.
7. Save the resampled audio signal into a stereo WAV file.

Group B – Time-Invariant Noise Filtering – Mandatory Part

1. Create an *.m file and name it (e.g., fas_lab_b.m).
2. Read the samples of the audio file, saved in Mandatory Part A and it’s sapling frequency into MATLAB™ workspace using `wavread`.

3. For stereo audio file extract left and right audio channel into two vectors (e.g., `audioL = audio_file(:,1)`) and process them separately.

4. Create two sine waves of the desired frequency† with the same sampling frequency as in audio signal.

5. Add the samples of the generated two sine waves to the samples of the audio signal.

6. Calculate the spectrum of the new signal and visualise it using `stem`.

7. Create a multiband digital filter with two stopbands to remove two added frequency components†:
   - Create a band-stop digital filter with the cutoff frequencies below \(F_{\text{stop1}}\) and above \(F_{\text{stop2}}\) the frequency of the first sine wave.
   - Create a band-stop filter for the second sine wave.
   - Connect these filters into cascade structure.

8. Apply the multiband filter using `filter` function (use function from Filter Design Toolbox).

9. Calculate and visualise the spectrum of the filtered signal.

**Group C – Complementary Part**

**Alternative 1 – Modulation of the Audio Signal**

1. Create an *.m file and name it (e.g., `fas_lab_c.m`).

2. Create a sine wave of the desired frequency.

3. Modulate the created signal with desired carrier frequency using `modulate`.

4. Plot 200 samples of the modulated signal.

5. Read the audio file using `wavread`. 
6. Take the desired number of samples from audio signal.
7. Modulate extracted audio signal samples.
8. Demodulate the modulated audio signal.
9. Plot the modulation and demodulation result using `plot`.

**Alternative 2 – Filtering of the Resampled Signal**

1. Create an *.m file and name it (e.g., `fas_lab_c.m`).
2. Read the audio signal, corrupted with two sine waves of frequencies, set in Mandatory Part B.
3. Create a multiband FIR filter to remove these two sine waves at sampling frequency of 44.1 kHz.
4. Calculate the upsampling factor to change the sampling frequency of the audio signal to 44.1 kHz.
5. Using `upfirdn` function apply the created multiband FIR filter to the audio signal.
6. Visualise the spectrum of initial and filtered audio signals.

**5.4 Tasks for Laboratory Work**

**Group A – Resampling of the Audio Signal – Mandatory Part**

**Alternative 1 – Resampling using FIR Filter**

**Task: Decimation of the Audio Record Sampling Frequency**

Using MATLAB™ `decimate` function, found in Signal Processing Toolbox, change the sampling rate of the audio record `audio1.wav` to 22.05 kHz. The decimation should be performed using FIR filter. Calculate and visualise the spectrum of the initial and decimated audio signal.

**Task: Interpolation of the Audio Record Sampling Frequency**

Using MATLAB™ `interp` function, found in Signal Processing Toolbox, change the sampling rate of the audio record `audio2.wav` to 44.1 kHz. Calculate and visualise the spectrum of the initial and interpolated audio signal.
**Task: Increasing the Audio Record Sampling Frequency**

Using MATLAB™ resample function, found in Signal Processing Toolbox, change the sampling rate of the audio record audio1.wav to 16 kHz. Calculate and visualise the spectrum of the initial and resampled audio signal.

**Task: Decreasing the Audio Record Sampling Frequency**

Using MATLAB™ resample function, found in Signal Processing Toolbox, change the sampling rate of the audio record audio1.wav to 48 kHz. Calculate and visualise the spectrum of the initial and resampled audio signal.

**Alternative 2 – Resampling using IIR Filter**

**Task: Decimation of the Audio Record Sampling Frequency**

Using MATLAB™ decimate function, found in Signal Processing Toolbox, change the sampling rate of the audio record audio1.wav to 16 kHz. The decimation should be performed using IIR filter. Calculate and visualise the spectrum of the initial and decimated audio signal.

**Group B – Time-Invariant Noise Filtering – Mandatory Part**

**Task: Audio Signal Filtering using Multiband FIR Filter**

Read the resampled audio file, saved in Mandatory Part Group A and add two sine waves with frequency of 1.5 kHz and 2.5 kHz. The magnitude of the sine waves should be not higher than 0.02. Filter these two sine waves from the signal using a multiband FIR filter.

Visualise the spectrum of the initial signal, signal with noise (with added sine waves) and filtered audio signal. For the filter design you may use fdesign.bandstop, dfilt.cascade and design functions.
Task: Audio Signal Filtering using Multiband IIR Filter

Read the resampled audio file, saved in Mandatory Part Group A and add two sine waves with frequency of 800 Hz and 1.4 kHz. The magnitude of the sine waves should be not higher than 0.018. Filter these two sine waves from the signal using a multiband IIR filter.

Visualise the spectrum of the initial and filtered audio signal. For the filter design you may use fdesign.bandstop, design and dfilt.cascade functions.

Group C – Complementary Part

Task: Modulation and Demodulation of the Signal using AM

Using MATLAB™ function modulate, modulate the sine wave of frequency 50 Hz with a carrier frequency of 2 kHz using Amplitude Modulation. Visualise the modulation result using plot function. Using Amplitude Modulation modulate 200 samples of the audio signal, used in Mandatory Part Group A.

Demodulate received signal using demod function. Visualise the initial, modulated and demodulated signal on the same graph.

Task: Modulation and Demodulation of the Signal using FM

Using MATLAB™ function modulate, modulate the sine wave of frequency 50 Hz on a 2 kHz sine wave using Frequency Modulation. Visualise the modulation result using plot function. Using Frequency Modulation modulate 200 samples of the audio signal, used in Mandatory Part Group A.

Demodulate received signal using demod function. Visualise the initial, modulated and demodulated signal on the same graph.

Task: Modulation and Demodulation of the Signal using QAM

Using MATLAB™ function modulate, modulate the sine wave of frequency 50 Hz on a 2 kHz sine wave using Quadrature Amplitude Modulation. Visualise the modulation result using plot function. Using Quadrature Amplitude Modulation modulate 200
samples of the audio signal, used in Mandatory Part Group A. 
Demodulate received signal using `demod` function. Visualise the initial, modulated and demodulated signal on the same graph.

5.5 Questionnaire

Q1. What are the main features of the time-invariant noise?

Q2. Why does `wavread` function give a two-column matrix instead a vector, representing the sound wave?

Q3. How (name the mathematical method) can we calculate the spectrum of an audio signal?

Q4. Is there a difference between spectrum calculation for mono and for stereo audio record?

Q5. Why the spectrum of the signal is symmetrical?

Q6. What is the relation between generated sine wave frequency and signal sampling frequency?

Q7. Does the magnitude of the signal increase when the sine wave is added to a signal?

Q8. Why does a single sine wave appear as several samples of the spectrum instead of one – at the frequency of the sine wave?

Q9. How should we select the frequencies for a bandstop filter if we want to remove a sine wave component from the signal?

Q10. How should we select the frequencies for a bandpass filter if we want to leave one sine wave component from the signal, where several sine waves with different magnitudes and frequencies are added?

Q11. Why should we connect two bandstop filters into a cascade structure instead parallel?
Bibliography


6.1 The Aim

The aim of this laboratory is to learn how to design a digital filter for processing of two-dimensional signal, such as a grayscale image.

6.2 Important Material

**Related MATLAB™ commands and functions**

abs, fft2, fftshift, fir1, fir2, for, ftrans2, ifft2, ifftshift, imhist, imread, imshow, rgb2gray, size, subplot, uint8, watershed.

6.2.1 Two-Dimensional Fourier Transform

The two-dimensional discrete Fourier transform (DFT) for image, described as a function \( I(m, n) \), where \( m \) and \( n \) are the indexes of the image pixels, is given by equation

\[
F(x, y) = \sum_{m=1}^{M} \sum_{n=1}^{N} I(m, n) e^{-j2\pi \left( \frac{x}{M+1} + \frac{y}{N+1} \right)}; \tag{6.1}
\]

here \( x = 1, 2, 3, \ldots, M \) and \( y = 1, 2, 3, \ldots, N \). The Fourier transform is completely reversible. The inverse DFT is given by equation

\[
I(m, n) = \frac{1}{(M+1)(N+1)} \sum_{x=1}^{M} \sum_{y=1}^{N} F(x, y) e^{j2\pi \left( \frac{x m}{M+1} + \frac{y n}{N+1} \right)}; \tag{6.2}
\]

here \( m = 1, 2, 3, \ldots, M \) and \( n = 1, 2, 3, \ldots, N \).

6.2.2 Image Filtering in Frequency Domain

The image can be treated as a two-dimensional signal. The high frequency elements in the image are the rapid changes of
Analysis and filtering of the image in frequency domain is performed in three steps:

1. Computation of the two-dimensional discrete Fourier transform \( F(x, y) \) for the image \( I(m, n) \).
2. Multiplication of \( F(x, y) \) by the two-dimensional filter function \( H(x, y) \).
3. Computation of the two-dimensional inverse Fourier transform \( I_{\text{filtered}}(m, n) \) of the multiplication result.

### 6.2.3 Image Processing using MATLAB

Images can be filtered in frequency domain using the two-dimensional FIR filters. A two-dimensional FIR filter can be designed by the transformation of the one-dimensional FIR filter using MATLAB\textsuperscript{TM} function `ftrans2`:

```matlab
h = ftrans2(b);
```

here: \( h \) is the kernel of the FIR filter; \( b \) are the coefficients of the one-dimensional FIR filter. `ftrans2` function performs a FIR filter transform in frequency domain preserving all main features of the one-dimensional filter.

### Example 6.1 Transformation of One-Dimensional into a Two-Dimensional Filter

Create a digital low-pass FIR filter with:

- filter order is equal to a rectangular the image width;
- normalised passband frequency 0.05.

First, the image should be read:

```matlab
1 % Reading of the RGB image
2 I = imread('gele.jpg');
```

Let’s calculate the image height and width:

```matlab
3 % Estimating image size
```
6.3 Procedure

6.3.1 Software Tools and Signal Sources

MATLAB™ software package (Navakauskas, Serackis 2008); Signal Processing Toolbox (The MathWorks, Inc. 2010b) and Image Processing Toolbox (The MathWorks, Inc. 2010a) need to be used. Sig-
nal data (images as a two-dimensional signals) have to be downloaded from the course web site. You may also use images taken from your digital camera.

6.3.2 Work Order

The laboratory work consists of mandatory and complementary parts. Mandatory part (A and B group) tasks need to be done first (sequentialy, in declared order). Only if time will permit the tasks from complementary part (C group) should be approach.

As some work group tasks have alternative work orders, please make sure that you follow the correct one work description.

Exact task numbers (from A, B and C groups) will be given individually in the class by laboratory work supervisor.

Group A – Mandatory Part

1. Create an *.m file and name it (e.g., if_lab_a.m).
2. Set the order for the designed image filter.
3. Read the image data into a three-dimensional array using `imread` function.
4. Convert RGB image (three-dimensional array) into grayscale—a two-dimensional matrix—if needed.
5. Calculate the width and the height of the image using \texttt{size} function.

6. Set the desired cutoff frequency given in the task.

7. Create a loop to calculate a filter function $H_d$ values for each image pixel, accordingly to a given equation (you may use \texttt{for} loop for this task).

8. Calculate the 2D DFT for the image using \texttt{fft2} function.

9. Shift the values of the 2D DFT using \texttt{fftshift}.

10. Apply created 2D filter to the image (multiply each element of image matrix with its corresponding element of filter matrix by using the \texttt{.*}).

11. Shift back the values of the filtered image matrix using \texttt{ifftshift}.

12. Calculate an inverse two-dimensional DFT of the filtered image using \texttt{ifft2}.

13. Calculate the complex modulus of the filtered image.

14. Change the filtered image variable type to \texttt{uint8}.

15. Visualise the result in one figure using functions \texttt{subplot} and \texttt{imshow}.

\textbf{Group B – Mandatory Part}

1. Create an \texttt{*.m} file and name it (e.g., \texttt{if_lab_b.m}).

2. Read the image using \texttt{imread} function.

3. Convert RGB image into grayscale if needed.

4. Calculate the width and the height of the image using \texttt{size} function.

5. Calculate the 2D DFT for the image using \texttt{fft2} function.

6. Shift the values of the 2D DFT using \texttt{fftshift}.

7. Choose the filter order equal to the image width (make sure that the image is rectangular) and set the desired cutoff frequency.
8. Design a one-dimensional FIR filter.
9. Transform designed filter using \texttt{ftrans2}.
10. Apply the two-dimensional FIR filter to a given image (multiply each element of image matrix with its corresponding element of filter matrix by using the \texttt{.*}).
11. Calculate an inverse DFT of the filtered image using \texttt{ifft2}.
12. Shift back the values of the filtered image matrix using \texttt{ifftshift}.
13. Change the filtered image variable type to \texttt{uint8}.
14. Visualise the result in one figure using functions \texttt{subplot} and \texttt{imshow}.

\textbf{Group C – Complementary Part}

1. Create an \texttt{*.m} file and name it (e.g., \texttt{vs\_lab\_c.m}).
2. Read the image using \texttt{imread} function.
3. Choose the filter order for the low-pass filter and the desired cutoff frequency.
4. Design a one-dimensional FIR filter.
5. Transform designed one-dimensional filter into two-dimensional filter using \texttt{ftrans2}.
6. Apply the two-dimensional FIR filter to a given image (multiply each element of image matrix with its corresponding element of filter matrix by using the \texttt{.*}).
7. Apply the image segmentation method, mentioned in the task. The more information can be found in literature (Navakauskas, Serackis 2008; Semmlow 2004: Chapter 15).
8. Visualise the result in one figure using functions \texttt{subplot} and \texttt{imshow}.
9. If the segmentation results are not acceptable, change the cutoff frequency of the designed FIR filter, repeat the filtering of initial image and filtered image segmentation.
6.4 Tasks for Laboratory Work

Group A – Mandatory Part

---

**Task: Design of the Low-Pass Butterworth Image Filter**

Filter the image in frequency domain using 15th order two-dimensional Butterworth low-pass filter, defined by

\[
H_d(h, w) = \frac{1}{1 + (\sqrt{2} - 1) (D(h, w)/F_c)^{2N}}; \\
(6.3)
\]

here \(h\) and \(w\) are the indexes of the image FFT matrix; \(D(h, w)\) is the distance matrix given by

\[
D(h, w) = \sqrt{(W/2 - (w - 1))^2 + (H/2 - (h - 1))^2}; \\
(6.4)
\]

here \(W\) is the width of the image, \(H\) is the height of the image; \(F_c\) is the cutoff frequency multiplied by the image width; \(N\) is the order of the filter.

The cutoff frequency of the filter should be equal to 0.06. Visualise the initial image and filtering result in one figure.

---

**Task: Design of the High-Pass Butterworth Image Filter**

Filter the image in frequency domain using 17th order two-dimensional Butterworth high-pass filter, defined by

\[
H_d(h, w) = \frac{1}{1 + (\sqrt{2} - 1) (F_c/D(h, w))^{2N}}; \\
(6.5)
\]

here \(h\) and \(w\) are the indexes of the image FFT matrix; \(D(h, w)\) is the distance matrix given by

\[
D(h, w) = \sqrt{(W/2 - (w - 1))^2 + (H/2 - (h - 1))^2}; \\
(6.6)
\]

here \(W\) is the width of the image, \(H\) is the height of the image; \(F_c\) is the cutoff frequency multiplied by the image width; \(N\) is the order of the filter.

The cutoff frequency of the filter should be equal to 0.017. Visualise the initial image and filtering result in one figure.
6.3 Task: Design of the Band-Pass Butterworth Image Filter

Filter the image in frequency domain using 20th order two-dimensional Butterworth band-pass filter, defined by

\[
H_d(h, w) = 1 - \frac{1}{1 + (\sqrt{2} - 1) \left( \frac{D(h, w) \cdot W_p}{D^2(h, w) - F_c^2} \right)^2N}; \tag{6.7}
\]

here \(h\) and \(w\) are the indexes of the image FFT matrix; \(D(h, w)\) is the distance matrix given by

\[
D(h, w) = \sqrt{(W/2 - (w - 1))^2 + (H/2 - (h - 1))^2}; \tag{6.8}
\]

here \(W\) is the width of the image, \(H\) is the height of the image; \(F_c\) is the cutoff frequency at the centre of the passband multiplied by the image width; \(W_p\) is the width of the passband; \(N\) is the order of the filter. The cutoff frequency of the filter should be equal to 0.05 and the width of the passband should be equal to 0.15 of the rectangular image width. Visualise the initial image and filtering result in one figure.

6.4 Task: Design of the Band-Stop Butterworth Image Filter

Filter the image in frequency domain using 25th order two-dimensional Butterworth band-stop filter, defined by

\[
H_d(h, w) = \frac{1}{1 + (\sqrt{2} - 1) \left( \frac{D(h, w) \cdot W_p}{D^2(h, w) - F_c^2} \right)^2N}; \tag{6.9}
\]

here \(h\) and \(w\) are the indexes of the image FFT matrix; \(D(h, w)\) is the distance matrix given by

\[
D(h, w) = \sqrt{(H/2 - (h - 1))^2 + (W/2 - (w - 1))^2}; \tag{6.10}
\]

here \(W\) is the width of the image, \(H\) is the height of the image; \(F_c\) is the cutoff frequency at the centre of the stopband multiplied by the image width; \(W_p\) is the width of the stopband; \(N\) is the order of the filter. The cutoff frequency of the filter should be
equal to 0.05 and the width of the stopband should be equal to 0.1 of the rectangular image width. Visualise the initial image and filtering result in one figure.

**Task: Design of the Low-Pass Gaussian Image Filter**

Filter the image in frequency domain using two-dimensional Gaussian low-pass filter, defined by

\[
H_d(h, w) = e^{-\frac{D^2(h, w)}{2\sigma^2}};
\]  

(6.11)

here \( h \) and \( w \) are the indexes of the image FFT matrix; \( D(h, w) \) is the distance matrix given by

\[
D(h, w) = \sqrt{(W/2 - (w - 1))^2 + (H/2 - (h - 1))^2};
\]  

(6.12)

here \( W \) is the width of the image, \( H \) is the height of the image; \( \sigma \) can be treated as the cutoff frequency multiplied by the image width. The cutoff frequency of the filter should be equal to 0.04 of the rectangular image width. Visualise the initial image and filtering result in one figure.

**Task: Design of the High-Pass Gaussian Image Filter**

Filter the image in frequency domain using two-dimensional Gaussian high-pass filter, defined by

\[
H_d(h, w) = 1 - e^{-\frac{D^2(h, w)}{2\sigma^2}};
\]  

(6.13)

here \( h \) and \( w \) are the indexes of the image FFT matrix; \( D(h, w) \) is the distance matrix given by

\[
D(h, w) = \sqrt{(W/2 - (w - 1))^2 + (H/2 - (h - 1))^2};
\]  

(6.14)

here \( W \) is the width of the image, \( H \) is the height of the image; \( \sigma \) can be treated as the cutoff frequency multiplied by the image width. The cutoff frequency of the filter should be equal to 0.08 of the rectangular image width. Visualise the initial image and filtering result in one figure.
Task: **Design of the Band-Pass Gaussian Image Filter**

Filter the image in frequency domain using two-dimensional Gaussian band-pass filter, defined by

\[ H_d(h, w) = e^{-\left( \frac{D^2(h, w) - \sigma^2}{D(h, w) \cdot W_p} \right)^2}; \]  

(6.15)

here \( h \) and \( w \) are the indexes of the image FFT matrix; \( D(h, w) \) is the distance matrix given by

\[ D(h, w) = \sqrt{(W/2 - (w - 1))^2 + (H/2 - (h - 1))^2}; \]  

(6.16)

here \( W \) is the width of the image, \( H \) is the height of the image; \( \sigma \) can be treated as the cutoff frequency at the centre of the passband multiplied by the image width; \( W_p \) is the width of the passband. The cutoff frequency of the filter should be equal to 0.06 and the width of the passband should be equal to 0.15 of the rectangular image width.

Visualise the initial image and filtering result in one figure.

---

Task: **Design of the Band-Stop Gaussian Image Filter**

Filter the image in frequency domain using two-dimensional Gaussian band-stop filter, defined by

\[ H_d(h, w) = 1 - e^{-\left( \frac{D^2(h, w) - \sigma^2}{D(h, w) \cdot W_p} \right)^2}; \]  

(6.17)

here \( h \) and \( w \) are the indexes of the image FFT matrix; \( D(h, w) \) is the distance matrix given by

\[ D(h, w) = \sqrt{(W/2 - (w - 1))^2 + (H/2 - (h - 1))^2}; \]  

(6.18)

here \( W \) is the width of the image, \( H \) is the height of the image; \( \sigma \) can be treated as the cutoff frequency at the centre of the stopband multiplied by the image width; \( W_p \) is the width of the stopband. The cutoff frequency of the filter should be equal to 0.03 and the width of the stopband should be equal to 0.1 of the rectangular image width.

Visualise the initial image and filtering result in one figure.
Group B – Mandatory Part

**Task: Image Filtering using 1D High-Pass Filter**

Using `fir1` function, create a high-pass FIR filter and apply it to the given image. The normalised cutoff frequency of the FIR filter should be equal to 0.16.

**Task: Image Filtering using 1D Low-Pass Filter**

Using `fir1` function, create a low-pass FIR filter and apply it to the given image. The normalised cutoff frequency of the FIR filter should be equal to 0.12.

**Task: Image Filtering using 1D Band-Pass Filter**

Using `fir2` function, create a band-pass FIR filter and apply it to the given image. The normalised cutoff frequencies of the FIR filter passband should be equal to 0.15 and 0.16.

**Task: Image Filtering using 1D Band-Stop Filter**

Using `fir1` function, create a band-stop FIR filter and apply it to the given image. The normalised cutoff frequencies of the FIR filter stopband should be equal to 0.13 and 0.14.

Group C – Complementary Part

**Task: Improvement of the Protein Spot Segmentation**

Create a low-pass filter to improve the segmentation of the image `gel.png`. The segmentation should be performed using Watershed Transformation based algorithm, implemented in `watershed` function.

**Task: Improvement of the Texture Segmentation**

Create a low-pass filter to separate two different textures in image `textures.png`. The segmentation of the image should be performed by the analysis of image histogram†.
6.5 Questionnaire

Q1. What is the influence of the high-pass filter, applied to an image?
Q2. Is it possible to design a two-dimensional band-pass or band-stop filter?
Q3. Is it possible to perform an image filtering not in a frequency domain?
Q4. Why the shift of the FFT is used for image filtering?
Q5. One period of the sine wave, plotted in the image (and takes the full image width), will be the high frequency element or the low frequency element of the image?
Q6. What is the advantage of the image filtering in frequency domain instead using the spatial filtering techniques?
Q7. How image analysis in frequency domain could be used in image compression algorithms?

Bibliography


7.1 The Aim

The aim of this laboratory is to learn how to design an artificial neural network to solve simple classification or function approximation task.

7.2 Important Material

<table>
<thead>
<tr>
<th>Related MATLAB™ commands and functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>feedforwardnet, imread, init, newff, newp, newrb, perceptron, plot, sim, train.</td>
</tr>
</tbody>
</table>

7.2.1 Single-Layer Perceptron

A mathematical model of the single perceptron is defined by (Haykin 2009: Chapter 3)

\[ y = \phi (w^T x + b); \]  
(7.1)

here \( \phi \) is the neuron activation function; \( w \) is the weight vector of the neuron inputs \( x \); \( b \) is the bias value of the neuron.

The most common activation function \( \phi \) used for perceptron is the signum function (Haykin 2009: Chapter 3):

\[ \text{sgn} (w^T x + b) = \begin{cases} 
+1, & \text{if } (w^T x + b) > 0, \\
-1, & \text{if } (w^T x + b) < 0. 
\end{cases} \]  
(7.2)

The single perceptron can be used to classify the set of \( n \), linearly separable, inputs \( x = [x_1, x_2, x_3, \ldots, x_n] \) into two classes, \( C_1 \) or \( C_2 \).

7.2.2 Training of the Artificial Neural Network

The classification result for the inputs \( x \) depends on the weight vector \( w = [w_1, w_2, w_3, \ldots, w_n] \) and bias value \( b \). To find the ap-
appropriate values of $w$ and $b$, the training of the perceptron is performed. Training of the perceptron is performed in 5 steps (Haykin 2009: Chapter 2, 3):

1. Initialisation. Initial values for $w$ and $b$ are set.
2. Activation. The perceptron is activated using inputs $x^\dagger$.
3. Computation of actual response. The signum function is applied for the weighted sum of the inputs:
   \[ y(n) = \text{sgn}(w^T x) \tag{7.3} \]
4. Adaptation of the weight vector. The weight vectors are updated accordingly to a desired response $d(n)$:
   \[ w(n + 1) = w(n) + \eta(d(n) - y(n)) x; \tag{7.4} \]
   here
   \[ d(n) = \begin{cases} +1, \text{if } x(n) \text{ belongs to class } C_1, \\ -1, \text{if } x(n) \text{ belongs to class } C_2. \end{cases} \]
5. Continuation. The time step $n$ is incremented and training is continued from the activation step.

### 7.2.3 Multi-Layer Perceptron

The non-linear classification, function approximation tasks can be solved using a multi-layer perceptron (MLP). In comparison to a single-layer perceptron the neuron activation functions, used for MLP, are non-linear. A common activation function for MLP is sigmoidal (logistic) function

\[ y = \frac{1}{1 + e^{(w^T x + b)}}. \tag{7.5} \]

The use of several layers (output layer and one or more hidden layers) of neurons in MLP makes it possible to solve various complex classification, approximation tasks. On the other hand, the use of additional layers leads to complexity in the update of the neuron input weights – training of the MLP. A computationally efficient algorithm for the training of MLP is the back-propagation algorithm (Rumelhart et al. 1986). The algorithm is
divided into two phases: forward pass and backward pass. In the forward pass, the input signal passes through the all layers of MLP and the weights of the neurons are not changed. In the backward pass, the error signal is passed through the neurons backward, starting from the last layer, and the error gradient is computed for each neuron in MLP. The input weights of the neuron are updated using delta rule (Haykin 2009: Chapter 4)

$$\Delta \omega(n) = \eta \delta(n) x(n); \quad (7.6)$$

here $\Delta \omega(n)$ is the value, added to a neuron input weight at the $n$-th iteration of the training algorithm; $\eta$ is the learning rate; $\delta(n)$ is the local error gradient for the neuron and $x(n)$ is the input of the neuron.

### 7.2.4 ANN Modeling using MATLAB

A single-layer perceptron can be easily created and trained using MATLAB™ Neural Network Toolbox functions (The MathWorks, Inc. 2010b) perceptron (newp in previous toolbox versions) and train.

**Example 7.1 A Single-Layer Perceptron for Classification**

Solve the classification task using a single-layer perceptron. Create a single-layer perceptron network to classify a set of examples into three classes accordingly to a four features. To classify the set of examples into 3 classes, we need to use two perceptrons in a layer with signum activation functions (output of the perceptron can be +1 or −1).

```
% Preparation of the class 1 examples
x_cl_1 = [0.21 -1.21 0.6 32; ...
           0.15 -1.05 0.7 26; ...
           0.12 -1.22 0.5 36];

% Preparation of the class 2 examples
x_cl_2 = [0.82 -1.31 0.1 18; ...
           0.79 -1.28 0.2 12; ...
           0.91 -1.29 0.1 13];

% Preparation of the class 3 examples
x_cl_3 = [-0.51 1.12 0.3 23; ...]
```
The weights and bias values of the created single-layer perceptron is obtained by network training, using known examples.

To test created and trained single-layer perceptron we can use the `sim` function.

7.3 Procedure

7.3.1 Software Tools and Signal Sources

MATLAB™ software package (Navakauskas, Serackis 2008); Neural Network Toolbox (The MathWorks, Inc. 2010b) and Image Processing Toolbox (The MathWorks, Inc. 2010a) need to be used. Signal data have to be downloaded from the course web site.
7.3 Procedure

7.3.2 Work Order

The laboratory work consists of mandatory and complementary parts. Mandatory part (A group) task needs to be done first (sequentialy, in declared order). Only if time will permit the task from complementary part (B group) should be approach.

As some work group tasks have alternative work orders, please make sure that you follow the correct one work description.

Exact task numbers (from A and B groups) will be given individually in the class by laboratory work supervisor.

**Group A – Classification using Perceptron – Mandatory Part**

1. Create an *.m file and name it (e.g., ann_lab_a.m).
2. Read all the images with apples and pears into MATLAB workspace using `imread`.
3. Using two prepared functions `spalva_color` and `apvalumas_roundness`, calculate for each image two features: colour and roundness.
4. Select the features, calculated for three apples and two pears and write into a matrix of size $2 \times 5$.
5. Set the vector (size $1 \times 5$) of the desired outputs† for the Single Perceptron.
6. Train the Single Perceptron using `train` function.
7. Simulate created Perceptron using `sim` function and input matrix, used for training.
8. Test the created Perceptron using features, calculated for the rest of images.

**Group B – Complementary Part**

1. Create an *.m file and name it (e.g., ann_lab_b.m).
2. Write the given discrete signal values into the row vector.
3. Create a neural network of the desired type.
4. Prepare the input and output values for the neural network.
5. Train the neural network using `train` function.
7. Plot the output of the network using `plot` function with parameters: ‘r*’.
8. Create an input vector with additional, intermediate values.
9. Simulate the neural network with the new input vector.
10. Plot the network simulation result using `plot` function with parameters: ‘kx’.

7.4 Tasks for Laboratory Work

Group A – Classification using Perceptron – Mandatory Part

Task: **Classification using Single Perceptron**

Using `perceptron` or `newp` (used in previous versions of the toolbox) function create a Single Perceptron to classify apples and pears given in a set of images. Train created Perceptron using three images with apples and two images with pears. Test the trained Single Perceptron using images of apples and pears, not used for training.

Group B – Complementary Part

**Alternative 1 – Data Approximation**

Task: **Data Approximation using Feedforward Network**

Using `feedforwardnet` or `newff` (used in previous versions of the toolbox) function create a feedforward network to approximate the discrete signal values, given below:

\[-0.96 - 0.577 - 0.0730.3770.6410.660.4610.134 - 0.201 - 0.434 - 0.5 - 0.393 - 0.1650.0990.3070.3960.3450.182 - 0.031 - 0.219 - 0.32\]

Task: **Data Approximation using Radial Basis Function Network**

Using `newrb` function create a neural network to approximate the discrete signal values, given below:
7.5 Questionnaire

Q1. What for the Single-Layer Perceptron is usually used?

Q2. How we can decide, for which class an object (defined by features used as perceptron inputs) is classified if a non-linear activation function is used and output is not equal to 1 or 0?

Q3. What are the limitations for using a Single-Layer Perceptron for classification tasks?

Q4. Is a Single-Layer Perceptron activation function always a signum function?

Q5. Can we use a string variable as an input to Single-Layer Perceptron?

Q6. Is the signum function used for MLP neurons as an activation function?

Q7. How we can select initial weights for neural network before training procedure?

Q8. What for the training of the neural network is needed?

Q9. Do we need to train a neural network each time when a new object is classified?

Q10. What is the difference between the MLP and RBF network?

Bibliography


8.1 The Aim

Knowledge about the nature, structure and properties of speech signals (Navakauskas, Paulikas 2004: Chapter 4) was acquired during lectures and self-study at home.

This laboratory work aims to enforce that knowledge throughout execution of practical task based on main procedures of speech signal processing and analysis (Rabiner, Schafer 1988: Chapters 3, 4 and 6).

It stipulates and develops programming, data analysis and processing skills, abilities to independently apply computerized techniques in order to solve specific practical speech signal analysis problems in time and frequency domains.

8.2 Important Material

<table>
<thead>
<tr>
<th>Related MATLAB™ commands and functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>disp, fft, figure, filter, fix, for, gca, get, grid, hamming, hold, if, length, log10, max, ones, plot, return, sound, sqrt, title, wavread, xlabel, ylabel, zeros.</td>
</tr>
</tbody>
</table>

8.2.1 Speech Signal Segmentation

Speech signal segmentation and phonetical discrimination are main problems in speech processing (Burrus et al. 1994: Chapter 10). It is quite difficult to automate the process as human (having required skills and deep analytical knowledge) intervention and fine-tuning is usuusally necessary.

That is why identification of speech signal parts and assignment them to particular phonemes is very instructive.
Example 8.1 Speech signal waveform presentation

Let us develop function (ksv.m) that reads speech signal from a disk and presents it's waveform in screen. Input data for the function will be speech signal file name fname, while function will output speech signal data s and it's sampling frequency fs.

```matlab
function [s, fs] = ksv(fname)
% function [s, fs] = ksv(fname) -
% Reading and presentation of speech signal:
% fname - file name;
% s - signal;
% fs - sampling frequency of the signal.

[s, fs] = wavread(fname);
plot((1:length(s))/fs, s); grid on;
title(['File ' fname]); xlabel('Time, s'); ylabel('Amplitude');
```

In the presented function: at L8 speech signal is read out, while at L9 it is presented in axis with grid as the waveform. Keep noted that in the same line discrete instances are mapped to time. At L10 figure is embellished with additional axes labels and title with file name of the speech signal included.

Example 8.2 Speech signal segmentation

Using in Example 8.1 showed speech signal presentation function let us analize saved in to the file (sound1.wav) fragment of a song. For the recording 16 bits analog/digital converter with 8 kHz sampling frequency was used.

```matlab
figure(1);
[s, fs] = ksv('sound1.wav');
yl = get(gca, 'ylim');
x = [3500 4400 4900 5400 10600 10900 18900... 19400 2750 28500 31300 32000 43500]/fs;
hold on;
plot([x; x],...
[yl;yl;yl;yl;yl;yl;yl;yl;yl;yl;yl;yl]',...
'r','linewidth',1);
hold off;
```
8.2 Important Material

Here at L1 first graphical window is selected and at L2 speech signal waveform is plotted. By L3 y axis limits are read-out (about program MATLAB™ graphics object cf. to (The MathWorks, Inc. 2011: see Handle Graphics in Section MATLAB)). Phonemes limits are set at L4-5. Between L6 and L10 graphical axes are holded in order to include there additional data. Keep noted about the use of suspension symbol – that is a special program MATLAB™ symbol to note command line continuation.

![Fig. 8.1 Results of function test4ksv.m](image)

Phonetic symbol system is used for the phonetic description of produced speech sounds (shown in Table 8.1 on the next page).

8.2.2 Speech Preemphasis

Spectrum of speech signal falls off at high frequencies. Frequently it is desirable to compensate this high frequency fall off – perform speech signal preemphasis. Simple and often used method of preemphasis is linear filtering by first order (“first difference”) filter (see filter) that can be expressed by:

\[ y(n) = x(n) - b_1 x(n - 1), \]  

(8.1)

here \( x(n) \) – the input speech signal; \( y(n) \) – the output preemphasized speech signal; \( b_1 \) – filter coefficient that needs to be tuned.
Table 8.1  Phonetic symbol system – ARPABET

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Example</th>
<th>Symbol</th>
<th>Example</th>
<th>Symbol</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>IY</td>
<td>beat</td>
<td>AY</td>
<td>buy</td>
<td>F</td>
<td>fat</td>
</tr>
<tr>
<td>IH</td>
<td>bit</td>
<td>OY</td>
<td>boy</td>
<td>TH</td>
<td>thing</td>
</tr>
<tr>
<td>EY</td>
<td>bait</td>
<td>Y</td>
<td>you</td>
<td>S</td>
<td>sat</td>
</tr>
<tr>
<td>EH</td>
<td>bet</td>
<td>W</td>
<td>wit</td>
<td>SH</td>
<td>shut</td>
</tr>
<tr>
<td>AE</td>
<td>bat</td>
<td>R</td>
<td>rent</td>
<td>V</td>
<td>vat</td>
</tr>
<tr>
<td>AA</td>
<td>Bob</td>
<td>L</td>
<td>let</td>
<td>DH</td>
<td>that</td>
</tr>
<tr>
<td>AH</td>
<td>but</td>
<td>M</td>
<td>met</td>
<td>Z</td>
<td>zoo</td>
</tr>
<tr>
<td>AO</td>
<td>bought</td>
<td>N</td>
<td>net</td>
<td>ZH</td>
<td>azure</td>
</tr>
<tr>
<td>OW</td>
<td>boat</td>
<td>NX</td>
<td>sing</td>
<td>CH</td>
<td>church</td>
</tr>
<tr>
<td>UH</td>
<td>book</td>
<td>P</td>
<td>pet</td>
<td>JH</td>
<td>judge</td>
</tr>
<tr>
<td>UW</td>
<td>boot</td>
<td>T</td>
<td>ten</td>
<td>WH</td>
<td>which</td>
</tr>
<tr>
<td>AX</td>
<td>about</td>
<td>K</td>
<td>kit</td>
<td>EL</td>
<td>battle</td>
</tr>
<tr>
<td>IX</td>
<td>roses</td>
<td>B</td>
<td>bet</td>
<td>EM</td>
<td>bottom</td>
</tr>
<tr>
<td>ER</td>
<td>bird</td>
<td>D</td>
<td>debt</td>
<td>EN</td>
<td>button</td>
</tr>
<tr>
<td>AXR</td>
<td>butter</td>
<td>H</td>
<td>get</td>
<td>DX</td>
<td>batter</td>
</tr>
<tr>
<td>AW</td>
<td>down</td>
<td>HH</td>
<td>hat</td>
<td>Q</td>
<td>(glottal stop)</td>
</tr>
</tbody>
</table>

8.2.3 Short-Time Fourier Analysis

Short-Time Fourier Transforms (SFT) can be expressed by:

\[ X_n(e^{j\lambda}) = \sum_{m=-\infty}^{\infty} w[n-m]x[m]e^{-j\lambda m} = \]  \( (8.2) \)

\[ = e^{-j\lambda n} \sum_{m=-\infty}^{\infty} w[-m]x[n+m]e^{-j\lambda m} = \]  \( (8.3) \)

\[ = e^{-j\lambda n} \tilde{X}_n(e^{j\lambda}), \]  \( (8.4) \)

here \(-\infty < n < \infty\) and \(0 \leq \lambda < 2\pi\) (or at any other \(2\pi\) duration interval).
SFT can be computed for a set of discrete frequencies $\lambda_k = \frac{2\pi k}{N}$ at fixed time $n$ by the use of Discrete-Time Fourier Transform (DFT) or its partial case — Fast Fourier Transform (FFT). If we assume that the window $w[-m] = 0$ for all $m < 0$ and $m > L - 1$, (8.2) can be re-written as:

\[
X_n[k] = X_n(e^{j(2\pi/N)k}) = \sum_{m=n}^{n+L-1} w[n-m]x[m]e^{-j(2\pi/N)km} =
\]

\[
e^{-j(2\pi/N)kn} \sum_{m=0}^{L-1} w[-m]x[n+m]e^{-j(2\pi/N)km} = \]

\[
e^{-j(2\pi/N)kn} \tilde{X}_n[k],
\]

here if $\tilde{w}[m] = w[-m]$ then:

\[
\tilde{X}_n[k] = \sum_{m=0}^{L-1} \tilde{w}[m]x[n+m]e^{-j(2\pi/N)km}, \quad k = 0, 1, \ldots, N - 1.
\]

$X_n[k]$ from $\tilde{X}_n[k]$ differs only by the exponential phase factor $e^{-j(2\pi/N)kn}$ thus $|X_n[k]| = |\tilde{X}_n[k]|$. (8.8) implies that $\tilde{X}_n[k]$ can be calculated according to Algorithm 8.1.

---

**Algorithm 8.1**

**Short-Time Fourier analysis**

A. Select $L$ samples of the signal at $n$th time instance, \{x[n], x[n-1], ..., x[n+L-1]\}. (For symmetrical windows $n$ usually is accepted as a center of the window.)

B. Multiply fragment of speech signal with window samples and form new set \{\tilde{w}[m]x[n+m]\}, $m = 0, 1, \ldots, L - 1$.

C. Calculate $N$-points DFT of cut off (with added zeros if $N > L$) speech segment.

D. Multiply result with $e^{-j(2\pi/N)kn}$ (skip this step if only amplitude of SFT is necessary).

E. Repeat steps (A)–(D) for each value $n$. 
Main parameter of SFT is the width of window. If window width is narrow in comparison with the part of speech under analysis then SFT can track changes of the utterance. If window width is wider then all changes of the utterance will be “blurred” in time. At the same time frequency resolution \((k)\) of SFT will be improved.

**Example 8.3**  
SFT window length investigation

Let us develop program MATLAB function `comp4wl.m` that will enable to investigate SFT window width influence on speech signals. Input data will be: speech signal \(s\), index of the middle sample of the window \(nc\), a set of window widths \(wl\), number of SFT samples \(nfft\) and offset of characteristics \(offset\), dB.

```matlab
function comp4wl(s, nc, wl, nfft, offset)
% function comp4wl(s, nc, wl, nfft, offset)
% SFT window width investigation
% s - speech signal;
% nc - index of the middle sample of the window,
% wl - set of window width (odd), e.g., [101, 201]
% nfft - number of SFT samples,
% offset - offset of characteristics, dB.

if ( (nc - fix(max(wl)/2) < 1) | ...
   (nc + fix(max(wl)/2) > length(s)) )
   disp(’Window is too long’);
   return
end;

nw = length(wl);
X = zeros(nfft, nw);
con = 1;
coninc = 10^(offset/20);
for k=1:nw,
    n1 = nc - fix(wl(k)/2);
    n2 = nc + fix(wl(k)/2);
    X(:, k) = con * fft( s(n1:n2).*hamming(wl(k)), nfft );
    con = con/coninc;
end;
f = (0:nfft/2)*(8000/nfft);
```
8.2 Important Material

X = sqrt(-1)*20* log10(abs(X(1:nfft/2+1,:)))+...
(ones(nw,1)*f).';
plot(X); grid on;
title('Spectrum using different window width');
xlabel('Frequency, Hz'); ylabel('Amplitude, dB');

L10–14 window widths are checked out and in case they are selected incorrectly function execution is stopped. Keep noted that at L22 SFT is performed using program MATLAB™ function fft initially prefiltering signal with Hamming window (hamming). At L26–27 only half of complex spectrum is taken as after FFT double length symetric spectrum is calculated. At L28 spectrums are presented in a graphical window (note that plot is used to present multiple data in the same axes).

Spectrums using different width windows

![Spectrums using different width windows](image)

**Fig. 8.2** Function `comp4wl.m` results

8.2.4 Formant Tracking

While interpreting prediction error filter (more about it can be found in Burrus et al. (1994)) often it is assumed that roots of the polynomial $A(z)$ (i.e., poles of the vocal tract filter) represent formants calculated from signal fragment (window) that was
used to derive prediction error filter. That is why roots express through angles, i.e., analog frequencies, are sometimes used as estimates of formant frequencies.

Fig. 8.3 shows by program MATLAB™ command `plot(F', 'r.')` presented estimates of formant frequencies calculated by the use of Algorithm 8.2.

![Fig. 8.3 Formant frequencies estimated by linear prediction](image)

**Algorithm 8.2**

**Calculation of formant frequencies**

A. **Starting from** `nbeg` **read-out required number of samples into array** `s`. **Let** `n=1`.

B. **Starting from** the sample `n` **calculate prediction filter coefficients for** `nw` **wide Hamming window filtered speech signal**.

C. **Calculate magnitudes** of the angles of prediction error filter roots. **Convert them into frequency, Hz** (taking into account sampling rate). **Save the result as new column of matrix** `F`.

D. **Let** `n=n+ninc`, **here** `ninc` **is shift of window in samples**. **Repeat processing from step (B) if condition** `n+nw<=length(x)` **is true**.
8.3 Procedure

8.3.1 Software Tools and Signal Sources

A subset of MATLAB™ software package functions (The MathWorks, Inc. 2011) in the following groups – elementary math, math constants, elementary matrices and arrays, basic functions, rounding and remainder, Fourier transforms, filtering and convolution, control flow, handle graphics, audio recording and playback – as well as some functions from Signal Processing Toolbox (The MathWorks, Inc. 2010) need to be used (Hunt et al. 2006; Navakauskas, Serackis 2008).

Signal data have to be downloaded from the course Web-site.

8.3.2 Work Order

The laboratory work consists of mandatory and complementary parts. Mandatory part (A and B group) tasks need to be done first (sequentially, in declared order). Only if time will permit the task from complementary part (C group) should be approach.

As some work group tasks have alternative work orders, please make sure that you follow the correct one work description.

Exact task numbers (from A, B and C groups) will be given individually in the class by laboratory work supervisor.

8.4 Tasks for Laboratory Work

Group A Tasks

Task: Speech preemphasis

1. Record 2–3 s duration three realizations of your speech.
2. Prepare script that:
   a) takes written sound file;
   b) preemphasizes speech signal with given coefficient $b_1$;
   c) merges original speech signal with 1 s duration silence and preemphasizes speech signal into output signal;
   d) presents output signal graphically and plays it.
3. Run written script on all three speech signal realizations, carefully listen to the sound, compare waveforms and draw conclusions related to influence of initial speech contents and effect of filter coefficient.

---

**Task: Voiced or unvoiced?**

1. Record 2–3 s duration realization of your speech.
2. On the foundation of `test4ksv.m` prepare your own script that segments recorded by you speech according to previously described ARPABET system. Be very careful in places where phonemes are missing or very unexpressed. Keep in mind and mark accordingly places where there are pauses or just a background noise.
3. Tune segmentation of the speech signal done visually by listening (by program MATLAB™ function `sound`) all fragments individually. Where did most adjustments happen?
4. Prepare two modifications of the speech signal where:
   ✓ vowels are replaces by zero amplitude signal;
   ✓ consonants are replaces by zero amplitude signal.
5. Listen to both signals. Which one of modified speech signals is more intelligible?

---

**Group B Tasks**

**Task: Sound spectrum**

Using already developed function `comp4wl.m` and downloaded from the course web page sound file write down script for the spectrum of the sound file investigation:

1. Try `nc = 4700, 10800, 30000` and determine corresponding phonemes.
2. Use different window widths `wl = 401, 201, 101, 51` and `nfft = 512`. What is the effect of window shortening?
3. Repeat investigation on the preemphasized speech signal. Is the effect of window shortening the same?
1. Create program MATLAB™ function `comp4wp.m` that will calculate SFT and present graphically magnitude for the same speech signal process with the width window however at different window positions. As a startup take below presented fragment of the function.

```matlab
function comp4wp(s, nc, ninc, wl, nfft, nw, offset)
% function comp4wp(s, nc, ninc, wl, nfft, nw, offset) -
% SFT window position investigation:
% s - speech signal;
% nc - index of the middle sample of the window,
% ninc - window shift,
% wl - window width (odd number),
% nfft - number of SFT samples,
% nw - number of windows,
% offset - offset of characteristics, dB.
```

2. Try created function in a three cases \( nc = 4700, 10800, 30000, \) i.e., similarly to previous task.
3. Repeat investigation on the preemphasized speech signal.

Group C Task

Solely basing on previously shown Algorithm 8.2 one can implement function that will produce also false values for angles of linear prediction coefficients, i.e., when \( w = 0 \) and \( w = \pi \) (see Fig. 8.3). Compose the function that calculates angles of linear prediction coefficients according to Algorithm 8.2 however additionally filters out false values.

8.5 Questionnaire

Q1. Name elementary part of speech used for segmentation.
Q2. What is an ARPABET? Please give some examples of this system.
Q3. How does coefficient of speech signal preemphasis filter need to be selected in order for filter to emphasize high frequencies?

Q4. What needs to be done with speech signal if the signal to be preemphasized is too long to be processed at once?

Q5. What is the difference between Discrete Fourier Transform and Fast Fourier Transform?

Q6. Name main parameters of Short-Time Fourier Transform and explain their influence on the transformation results.

Q7. How formant frequencies can be estimated by the use of signal modelling?

Q8. Explain briefly the essence of prediction error filtering.

Bibliography


SPEECH SIGNAL MODELLING

9.1 The Aim

A discrete-time system model (see Oppenheim, Schafer (1999) as a classical reference) that is employed to generate samples of speech signal forms the foundation of modern speech signal processing algorithms.

This laboratory work aims to enforce the knowledge about speech models used in algorithms for speech synthesis, coding and recognition (see Rabiner, Schafer (1988) as another old but in the field still valuable book).

The laboratory work stipulates and develops programming, data analysis and processing skills, abilities to independently apply computerized techniques in order to solve specific practical speech signal synthesis problems.

9.2 Important Material

Related MATLAB™ commands and functions

| abs, conv, cos, filter, fliplr, flipud, for, if, length, log, ones, plot, roots, size, subplot, zplane |

9.2.1 Glottal Pulse Modelling

The block diagram of speech signal formation process that is often used in speech synthesizers is shown in Fig. 9.1 on the next page.

During speech formation voiced sounds are excited by quazi-periodical opening/closing of glottis, i.e., vibration of vocal chords. That activity is modelled by two coupled blocks: the Impulse Series Generator and the Glottis Filter (signal period) Model.

Achieved shape of the impulse $A_v$ determines spectral amplitude and phase characteristics of the outputed signal $p_l(n)$ of the whole speech signal model.
Fig. 9.1 Discrete time system model for speech signal formation

Simple glottis one period signal model, that we will name *Exponential model*, can be expressed by:

$$G(z) = \frac{-ae \ln(a) z^{-1}}{(1 - az^{-1})^2}, \quad (9.1)$$

here $e = 2.71828 \ldots$ is the base of natural logarithm; $a$ – tunable parameter.

Pioneering work on glottis signal research for speech modelling was done by Rosenberg (1971). Based on received experimental data Rosenberg proposed such empirical glottis signal model:

$$g_R(n) = \begin{cases} \frac{1}{2} [1 - \cos (\pi n/N_1)], & 0 \leq n \leq N_1; \\ \cos [\pi (n - N_1) / (2N_2)], & N_1 \leq n \leq N_1 + N_2; \\ 0, & \text{otherwise}. \end{cases} \quad (9.2)$$

*Rosenberg model* takes into account main features of glottis signal that was captured by inverse filtering and high speed shooting of vocal tract motion.

**9.2.2 Vocal Tract Modelling**

One of possible ways to model sound propagation in the vocal tract (cf. Fig. 9.1 block $V(z)$) is the use of connected set of lossless acoustic tubes (see Fig. 9.2 on the next page).
Basing on acoustic speech formation theory, system regularity and assumption that acoustic tubes are lossless it is possible to prove that system can be expressed by relatively simple wave equation with simple boundary conditions at the tube junctions. For sampled signals with sampling period $T = 2\tau$, the structure on Fig. 9.2 corresponds to discrete-time lattice filter (Fig. 9.3(a)) and can be represented alternatively by Figures 9.3(b) or (c).

Lossless tube models are important for mastering theory of acoustic speech formation and are advantageous in the design of speech synthesis systems. It is proven that when $r_G = 1$ discrete-time vocal model composed from similar length $N$ tubes can be expressed by the following system function:

$$V(z) = \prod_{k=1}^{N} (1 + r_k) z^{-N/2} / D(z).$$  \hspace{1cm} \text{(9.3)}$$

In the expression denominator $D(z)$ obeys following polynomial recursion (Rabiner, Schafer 1988) conditions:

$$D_0(z) = 1;$$
$$D_k(z) = D_{k-1}(z) + r_k z^{-k} D_{k-1}(z^{-1}), \quad k = 1, 2, \ldots, N; \hspace{1cm} \text{(9.4)}$$
$$D(z) = D_N(z),$$
here $r_k$ are reflection coefficients that can be calculated by:

$$r_k = \frac{A_{k+1} - A_k}{A_{k+1} + A_k}.$$  \hspace{1cm} (9.5)

Derivating recursion (9.4) it was assumed that at the glottis is lossless ($r_G = 1$), while lossess at the lips are assessed by:

$$r_N = r_L = \frac{A_{N+1} - A_N}{A_{N+1} + A_N},$$  \hspace{1cm} (9.6)

here $A_{N+1}$ is the area of matched impedance tube that can be selected to introduce losses in the system.
Example 9.1 Lossless tube model for speech signal

Let us assume that we know cross-section areas of lossless tubes model and would like to get it’s system function in order to model it by program MATLAB\textsuperscript{TM} function \texttt{filter}, i.e., we are willing to get following system function:

\[ V(z) = \frac{G}{D(z)} = G \left/ \left( 1 - \sum_{k=1}^{N} \alpha_k z^{-k} \right) \right. \]  \hspace{1cm} (9.7)

Keep noted that in (9.7) $N/2$ samples delay is not used as it is not necessary for speech synthesis. Program MATLAB\textsuperscript{TM} function \texttt{A2V} code is presented below. It evaluates reflection coefficients of lossless tubes model by the use of following parameters: \texttt{A} – cross-section areas and \texttt{rN} – reflection coefficient at lips side.

\begin{verbatim}
function [r, D, G] = A2V(A, rN)

% function [r, D, G] = A2V(A, rN) -
% Lossless tube model calculation:
% A - array of tubes cross-section areas,
% rN - reflection coefficient at lips side (< 1);
% r - array of reflection coefficients,
% D - system function denominator,
% G - system function numerator.

[M, N] = size(A);
if (M =1), A = A'; end;
N = length(A);
r = [ ];
for m=1:M-1,
    r = [r (A(m+1)-A(m))/(A(m+1)+A(m))];
end;
r = [r rN];
D = [1]; G = 1;
for m=1:N,
    G = G*(1+r(m));
    D = [D 0] + r(m).*[0 fliplr(D)];
end;

end;
\end{verbatim}
9.2.3 Vowel Modelling

For the voiced sound modelling instead of model presented in Fig. 9.1 it is possible to employ simplified model showed in Fig. 9.4 below.

\[ e(n) \rightarrow \text{Glottis Signal Model, } G(z) \rightarrow \text{Vocal Tract Model, } V(z) \rightarrow \text{Emission Model, } R(z) = 1 - z^{-1} \rightarrow s(n) \]

Fig. 9.4 Simplified model of voiced sound synthesis

In order to develop scripts for periodic signal synthesis already presented program MATLAB™ functions together with standard function such as filter, conv, zfill and ones can be used.

9.3 Procedure

9.3.1 Software Tools and Signal Sources

A subset of MATLAB™ software package functions (The MathWorks, Inc. 2011) in the following groups – basic functions, basic plots and graphs, elementary matrices and arrays, array manipulation, polynomials, filtering and convolution, control flow – need to be used (Hunt et al. 2006; Navakauskas, Serackis 2008).

Signal data have to be downloaded from the course Web-site.

9.3.2 Work Order

The laboratory work consists of mandatory and complementary parts. Mandatory part – A and B group – tasks need to be done first (sequentialy, in the declared order). Only if time will permit the tasks from complementary part – C group – should be approached.

As some work group tasks have alternative work orders, please make sure that you follow the correct one work description.

Exact task numbers (from A, B and C groups) will be given individually in the class by laboratory work supervisor.
9.4 Tasks for Laboratory Work

Group A Tasks

**Task: Exponential model**

Compose program MATLAB™ function `glottalExp` that according to *Exponential model* (9.1) will generate $N_s$ samples of glottal signal. As a result function has to output magnitude of frequency response of the glottal signal model to the graphical window. Invocation of the function needs to be as follows:

\[
[gE, GE, W] = \text{glottalExp}(a, Ns, Nf);
\]

Here $gE$ – $N_s$ samples length signal of Exponential glottal model; $GE$ – $N_f$ samples of magnitude of frequency response, calculated for the Exponential glottal model at the frequencies $W$ in the range from 0 to $\pi$. Program MATLAB™ functions `log`, `abs` and `plot` can be useful in your work.

**Task: Rosenberg model**

Compose program MATLAB™ function that according to *Rosenberg model* (9.2) will generate $N_1 + N_2 + 1$ samples. As a result function has to output magnitude of frequency response of the glottal signal model to the graphical window. Invocation of the function needs to be as follows:

\[
[gR, GR, W] = \text{glottalRos}(N1, N2, Nf);
\]

Here $gR$ – $N_1+N_2+1$ samples length signal of Rosenberg glottal model; $GR$ – $N_f$ samples of magnitude of frequency response, calculated for the Rosenberg glottal model at the frequencies $W$ in the range from 0 to $\pi$. Program MATLAB™ functions `cos`, `abs` and `plot` can be useful in your work.

Group B Tasks

**Task: Lossless tube model conversion to system function**

1. Let us assume that vowel is described by the following lossless tube model parameters: $A = \{1.6, 2.6, 0.65, 1.6, 2.6, 4, 6.5, 8, 7, 5\}$. Moreover let us consider two cases: $r_N = 1$ and $r_N = 0.7$. 
2. Using program MATLAB™ function \texttt{A2V} calculate vocal tract system function denominator $D(z)$ and frequency response magnitudes of considered lossless tube models. Present results in the graphical window at the same time showing results of both cases.

3. Factor both polynomials $D(z)$ and represent their poles in $z$-domain by the use of program MATLAB™ function \texttt{zplane}. For the first case represent poles by “o” whilst for the second case – by “+” symbols. Where reside roots in a completely lossless case and how they move when lossess appear?

4. Calculate effective length of the vocal tract assuming that speech signal sampling frequency is 10 000 samples/s.

**Task: System function conversion to lossless tube model**

Inverse problem arrises when system function (9.7) is known however we are interested in determination of lossless tubes model parameters: reflection coefficients and cross-section areas. The system function denominator $D(z)$ obeys (9.4) conditions thus it can be used for required model parameters’ calculation.

1. Acknowledge that $r_N$ is equal to $z^{-N}$ coefficient of $V(z)$ denominator, i.e., $r_N = -\alpha_N$.

2. By the use of (9.4) it can be proved that:

$$D_{k-1} = \frac{D_k(z) - r_k z^{-k} D_k(z^{-1})}{1 - r_k^2}, \quad k = N, N - 1, \ldots, 2.$$

3. By the use of (1) and (2) rezults and knowing that $D_N(z) = D(z)$ write down $r_{N-1}$ expression.

Now, on the foundations of (1)–(3) rezults compose the algorithm that calculates reflection coefficients $r_k, k = 1, 2, \ldots, N$ and acoustic tubes cross-section areas $A_k, k = 1, 2, \ldots, N$. Implement the algorithm by program MATLAB™ function \texttt{V2A} (with similar to \texttt{A2V} calculation order) that needs to have heading as follows:

```matlab
function [r, A] = V2A(D, A1)
% function [r, A] = V2A(D, A1) -
% Calculation of lossless tubes model parameters
```
9.4 Tasks for Laboratory Work

% (reflection coefficients and cross-section areas):
% D - array of denominator coefficients,
% A1 - cross-section area of the first tube,
% r - array of reflection coefficients,
% A - array of cross-section areas of tubes.

---

**Task: Vowel synthesis**

1. Let us assume that sampling frequency is 10 000 samples/s. Create 1 000 samples length signal having 100 Hz fundamental frequency.

2. By the use of program MATLAB™ functions filter and conv compose your script that simulates in Fig. 9.4 presented system model. For that purpose consider the use of developed excitation signal \( e(n) \), Exponential and Rosenberg glottal models’ simulation functions (glottalExp and glottalRos), and emission model of the form \( R(z) = 1 - z^{-1} \).

3. With program MATLAB™ functions plot and subplot in a figure present 1 000 samples length synthetic speech signal fragments when:
   - ✓ Exponential glottal signal model parameter – \( a = 0.91 \);
   - ✓ Rosenberg glottal signal model parameters: \( N1 = 40 \) and \( N2 = 10 \);
   - ✓ Vowel /a/ vocal tract model:
     \[
     D(z) = 1 - 0.046z^{-1} - 0.6232z^{-2} + 0.3814z^{-3} - 0.2443z^{-4} + 0.1973z^{-5} + 0.2873z^{-6} + 0.3655z^{-7} - 0.4806z^{-8} - 0.1153z^{-9} + 0.7100z^{-10}.
     \]  

---

**Group C Tasks**

**Task: Glottal signal models’ comparison**

Compare three glottal signal models:

1. By the use of previously (in Tasks 9.1 and 9.2) developed functions, calculate \( Ns \) samples of \( gE \) and \( gR \) glottal signals when Exponential model has \( a = 0.91 \), whilst for Rosenberg model
has $N_1=40$ and $N_2=10$.

2. By the use of program MATLAB™ functions `fliplr` or `flipud` construct $g_R$ signal reflection. That is equivalent to composition of the following signal:

$$g_{RF}(n) = g_R(- (n - N_1 - N_2)). \quad (9.9)$$

3. Employing program MATLAB™ function `plot` present 51 sample length all three glottal signals in a one figure. Also in the second figure show all models frequency response magnitudes in dB scale. Re-calculate everything several times in order to find out how model parameters influences time and frequency characteristics of the glottis signal.

---

**Task: Rosenberg model’s investigation**

1. Employing program MATLAB™ function of Rosenberg model create several glottal signals when $N_2 = 10, 15, 25$ and $N_1 + N_2 = 50$. Present results into the same graphical axis. In the second figure present corresponding frequency response magnitudes. What is the influence of parameter $N_2$ on the Fourier transformation of glottal signal?

2. Exponential model has one zero at $z = 0$ and double pole at $z = a$. By the use of program MATLAB™ function `roots` and parameter values – $N_1=40$, $N_2=10$ – calculate zeros of $z$-transform of Rosenberg and inverse Rosenberg glottal signal models. Present results using program MATLAB™ function `zplane`. Why is inverse Rosenberg glottal signal model called *minimal phase system*? How can then Rosenberg glottal signal model be called?

---

**Task: Whispered vowel synthesis**

1. Whispered vowel signals can be modelled exciting vocal tract and emission models by random noise (corresponding to turbulent glottis airflow). Using program MATLAB™ function `rand` and previously developed ones compose script that synthesizes
vowel /a/ described by following vocal tract model:

\[ D(z) = 1 - 0.046z^{-1} - 0.6232z^{-2} + 0.3814z^{-3} \]
\[ - 0.2443z^{-4} + 0.1973z^{-5} + 0.2873z^{-6} \]
\[ + 0.3655z^{-7} - 0.4806z^{-8} - 0.1153z^{-9} + 0.7100z^{-10}. \]  

2. Let \( R(z) = 1 - z^{-1} \). Present system’s \( H(z) = G(z)V(z)R(z) \) function graphically as frequency response magnitude.

3. In another graphical window present resulting 1000 samples length signal. Try listening at least 0.5 s duration signal.

9.5 Questionnaire

Q1. Entitle and explain individually blocks that are used to model speech signal formation process.

Q2. What is a purpose of Glottis Signal Generator and how is it modelled?

Q3. What is a lossless tube model and how is it used in acoustic speech formation theory?

Q4. Explain boundary conditions for lossless tube model used in vocal tract modelling.

Q5. How vowel sounds can be modelled?

Bibliography


**LIST OF MATLAB™ FUNCTIONS**

<table>
<thead>
<tr>
<th><strong>A</strong></th>
<th><strong>B</strong></th>
<th><strong>C</strong></th>
<th><strong>D</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>abs</td>
<td>bilinear</td>
<td>cheb1ap</td>
<td>decimate</td>
</tr>
<tr>
<td>The absolute value of the elements</td>
<td>Bilinear transformation with optional frequency prewarping</td>
<td>Chebyshev type I analog lowpass filter prototype</td>
<td>Resample data at a lower rate after lowpass filtering</td>
</tr>
<tr>
<td></td>
<td>buttap</td>
<td>cheb1ord</td>
<td>demod</td>
</tr>
<tr>
<td>Butterworth analog lowpass filter prototype</td>
<td>Chebyshev type I filter order selection</td>
<td>Chebyshev type II analog lowpass filter prototype</td>
<td>Signal demodulation for communications simulations</td>
</tr>
<tr>
<td></td>
<td>butter</td>
<td>cheb2ap</td>
<td>design</td>
</tr>
<tr>
<td>Butterworth digital and analog filter design</td>
<td>Chebyshev type II filter order selection</td>
<td>Chebyshev type I digital and analog filter design</td>
<td>Digital filter design</td>
</tr>
<tr>
<td></td>
<td>buttord</td>
<td>cheby1</td>
<td>dfilt</td>
</tr>
<tr>
<td>Butterworth filter order selection</td>
<td>Chebyshev type I digital and analog filter design</td>
<td>Chebyshev type II digital and analog filter design</td>
<td>Digital filter implementation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cheby2</td>
<td>dfilt.cascade</td>
</tr>
<tr>
<td></td>
<td></td>
<td>conv</td>
<td>Create a cascade of discrete-time filters</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cos</td>
<td></td>
</tr>
<tr>
<td>Cosine of argument in radians</td>
<td></td>
<td></td>
<td>Direct-form I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dfilt.parallel</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Create a parallel system of discrete-time filter objects</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>disp</td>
</tr>
<tr>
<td>Display array</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
E

ellip
Elliptic or Cauer digital and analog filter design

ellipap
Elliptic analog lowpass filter prototype

ellipord
Elliptic filter order selection

else
Used with if

end
Terminate scope of for, while, switch, try, and if statements

F

fdatool
Filter design and analysis tool

fdesign.bandstop
Band-stop filter structure design

feedforwardnet
Feedforward network creation

fft
Discrete Fourier transform

fft2
Two-dimensional discrete Fourier transform

fftshift
Shift zero-frequency component to center of spectrum

figure
Create figure window

filter
One-dimensional digital filter

fir1
IR filter design using the window method

fir2
FIR arbitrary shape filter design using the frequency sampling method

fircls1
Low and high pass FIR filter design by constrained least-squares

firls
Linear-phase FIR filter design using least-squares error minimization

fix
Round towards zero

fliplr
Flip matrix in left/right direction

flipud
Flip matrix in up/down direction

for
Repeat statements a specific number of times

freqz
Digital filter frequency response

ftrans2
2-D FIR filter using frequency transformation

function
Add new function

fvtool
Filter visualization tool
<table>
<thead>
<tr>
<th>G</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>gca</td>
<td>Get handle to current axis</td>
</tr>
<tr>
<td>get</td>
<td>Get object properties</td>
</tr>
<tr>
<td>grid</td>
<td>Grid lines</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>H</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>hamming</td>
<td>Hamming window</td>
</tr>
<tr>
<td>help</td>
<td>Display help text in command window</td>
</tr>
<tr>
<td>hold</td>
<td>Hold current graph</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>if</td>
<td>Conditionally execute statements</td>
</tr>
<tr>
<td>ifft2</td>
<td>Two-dimensional inverse discrete Fourier transform</td>
</tr>
<tr>
<td>ifftshift</td>
<td>Inverse discrete Fourier transform shift</td>
</tr>
<tr>
<td>imag</td>
<td>Complex imaginary part</td>
</tr>
<tr>
<td>imhist</td>
<td>Display histogram of image data</td>
</tr>
<tr>
<td>impinvar</td>
<td>Impulse invariance method for analog to digital filter conversion</td>
</tr>
<tr>
<td>impz</td>
<td>Impulse response of digital filter</td>
</tr>
<tr>
<td>imread</td>
<td>Read image from graphics file</td>
</tr>
<tr>
<td>imshow</td>
<td>Display image in handle graphics figure</td>
</tr>
<tr>
<td>init</td>
<td>Initialize a tscollection object with new time or new time series</td>
</tr>
<tr>
<td>interp</td>
<td>Resample data at a higher rate using lowpass interpolation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>Length of vector</td>
</tr>
<tr>
<td>log</td>
<td>Natural logarithm</td>
</tr>
<tr>
<td>log10</td>
<td>Common (base 10) logarithm</td>
</tr>
<tr>
<td>lookfor</td>
<td>Search all m-files for keyword</td>
</tr>
<tr>
<td>lp2lp</td>
<td>Lowpass to lowpass analog filter transformation</td>
</tr>
<tr>
<td>M</td>
<td>max</td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>modulate</td>
</tr>
<tr>
<td>N</td>
<td>nargin</td>
</tr>
<tr>
<td></td>
<td>nargout</td>
</tr>
<tr>
<td></td>
<td>newff</td>
</tr>
<tr>
<td></td>
<td>newp</td>
</tr>
<tr>
<td></td>
<td>newrb</td>
</tr>
<tr>
<td>O</td>
<td>ones</td>
</tr>
<tr>
<td>P</td>
<td>perceptron</td>
</tr>
<tr>
<td></td>
<td>plot</td>
</tr>
<tr>
<td>R</td>
<td>resample</td>
</tr>
<tr>
<td></td>
<td>residuez</td>
</tr>
<tr>
<td></td>
<td>return</td>
</tr>
<tr>
<td></td>
<td>rgb2gray</td>
</tr>
<tr>
<td></td>
<td>roots</td>
</tr>
<tr>
<td>S</td>
<td>sim</td>
</tr>
<tr>
<td></td>
<td>size</td>
</tr>
<tr>
<td></td>
<td>sound</td>
</tr>
<tr>
<td></td>
<td>sqrt</td>
</tr>
<tr>
<td></td>
<td>stem</td>
</tr>
<tr>
<td></td>
<td>subplot</td>
</tr>
<tr>
<td>T</td>
<td>tf2sos</td>
</tr>
<tr>
<td>-----</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>tf2zp</td>
</tr>
<tr>
<td></td>
<td>title</td>
</tr>
<tr>
<td></td>
<td>train</td>
</tr>
<tr>
<td>U</td>
<td>uint8</td>
</tr>
<tr>
<td></td>
<td>unwrap</td>
</tr>
<tr>
<td></td>
<td>upfirdn</td>
</tr>
<tr>
<td>W</td>
<td>watershed</td>
</tr>
<tr>
<td></td>
<td>wavread</td>
</tr>
<tr>
<td></td>
<td>while</td>
</tr>
<tr>
<td></td>
<td>window</td>
</tr>
<tr>
<td>X</td>
<td>xlabel</td>
</tr>
<tr>
<td>Y</td>
<td>ylabel</td>
</tr>
<tr>
<td></td>
<td>yulewalk</td>
</tr>
<tr>
<td>Z</td>
<td>zeros</td>
</tr>
<tr>
<td></td>
<td>zp2sos</td>
</tr>
<tr>
<td></td>
<td>zp2tf</td>
</tr>
<tr>
<td></td>
<td>zplane</td>
</tr>
</tbody>
</table>
SUBJECT INDEX

A
abs .................. 26, 29, 33, 60, 71, 105
algorithm
  formant frequencies ........ 94
  Short-Time Fourier analysis .91
approximation ................. 79
attenuation
  stopband ................... 51
B
bilinear .................. 37
buttap .................... 37, 40, 44
butter ................. 38, 40, 43, 44, 51
butterord ................. 43, 44, 51
C
cheb1ap .................. 40
cheb1ord .................. 43
cheb2ap .................. 40, 45
cheb2ord .................. 44
cheby1 .................... 38, 40, 43
cheby2 .................... 38, 40, 44
classification ............... 79
conv ....................... 104, 107
cos ......................... 105
D
decimate ................. 59, 62, 63
decimation .................. 57
demod ....................... 64, 65
demodulation ................ 57
denominator ................ 48, 50
design
  filter ....................... 35
  analog ..................... 37
  Elliptic .................. 38
design .................... 63, 64
dfilt ....................... 50
dfilt.cascade ............ 51, 63, 64
dfilt.df1 ................... 51
dfilt.parallel ............. 51
disp ........................... 92
domain
  frequency ................ 68
E
ellip .................... 38, 40, 44
ellipap ..................... 40
ellipord ................... 38, 44
else ......................... 15
end ......................... 15, 16
error
  mean square ............... 59
expression
  partial-fraction .......... 36
F
fdatool ................. 29–31, 39, 41–43, 52
fdesign.bandstop ........... 63, 64
feedforwardnet ............. 84
fft ................. 16, 26, 28, 29, 33, 92, 93
fft2 ....................... 69, 71
fftshift ................... 71
figure ....................... 88
filter
  analog .................. 35, 36
    Butterworth ........... 37
  band-pass ............... 36, 49, 50
  band-stop ............... 49, 58
  bandstop ................ 38
  coefficients .......... 35
digital .................. 35, 36
    Butterworth ........... 51
design
  expression
    partial-fraction ....... 49, 50
    FIR ........... 57, 58
first order difference.............89
frequency
cutoff..........................49
function
system ..........................38
transfer ..........................48
glottis model ......................99
high-pass ..........................49, 50
IIR ................................21, 35, 47, 58
image ..............................67
lattice ..............................102
low-pass ............................36, 49, 50, 58, 59
multiband ............................50, 51
one-dimensional ......................68
poles ...............................36, 50
prediction errors ......................93
preemphasis .......................89–90
realisation ..........................47
Cascade ............................49–51
Direct Form I .....................47, 48, 51
Direct Form II .....................48
Parallel .............................49
parallel ..............................50
structures ............................47
two-dimensional .....................68
zeros ................................50
filter ..............................61, 89, 103, 104, 107
filtering
inverse ..............................100, 105
noise ...................57
time-invariant .......................58
fir1 ..........................24, 25, 27, 32, 59, 69, 77
fir2 ..........................25, 27, 28, 32, 77
fircls1 ............................29, 33
firls ..............................59
fix ..................................92
fliplr ............................103, 108
flipud ................................108
for ..................................71, 92, 103
formant ............................93–95
cutoff
passband ............................37
stopband ............................37
normalised ..........................38, 50
Nyquist ............................38
sampling ............................37, 38
frequency response
magnitude ..........................105, 108
freqz ..................25, 28, 29, 33, 38, 44, 45
ftrans2 ..................68, 69, 72
function
activation ..........................79–81
arguments .........................14–15
local ..............................15–16
non-linear .............................80
private ..........................16–17
sigmoidal .............................80
signum .............................79
structure ..........................13–14
function ..........................13, 15, 16
fvtool .............................25, 32, 43, 44, 51
G
gca ..................................88
get ..................................88
graphics
menu
Favorites ............................11
grid ...............................88, 93
H
hamming ............................92, 93
help ..................................14
hold ..................................88
I
if ..................................15, 92, 103
ifft2 ...............................69, 71, 72
ifftshift ............................69, 71, 72
imag ..............................26, 28, 33
image
analysis ....................... 68
Discrete Fourier Transform . . 69
filtering .......................... 68
RGB ............................. 69
imhist ........................... 77
impinvar .......................... 40, 44, 45
impz .............................. 25, 28, 29, 33, 44, 45
imread ............................. 68, 70–72, 83
imshow ............................. 69, 71, 72
init ................................. 82
interp .............................. 59, 62
interpolation ...................... 57
length ...................... 92, 103
log .............................. 26, 28, 105
log10............................... 93
lookfor............................ 14
lp2lp .............................. 37, 40, 45
max ................................. 92
model
  discrete-time .................. 99
  exponential .................... 99, 105, 107
  glottal signal .............. 99, 105, 107–108
  lossless acoustic tubes .... 100, 101, 103
  lossless acoustic tubes ..... 106
  minimal phase ............... 108
  minimal phase ................ 107
  Rosenberg .................. 100, 105, 107, 108
  speech signal.............. 99
  voiced sound ............... 104
modulate .......................... 61, 64
modulation ...................... 57, 58
artificial ........................ 79
bias ............................... 80, 82
desired response................ 80
examples .......................... 82
inputs ............................. 80
layer .............................. 81
training .......................... 80, 81
weights ............................ 79, 80, 82
newff ............................. 84
newp ............................. 81, 82, 84
newrb ............................. 84
numerator .......................... 48, 50
ones ............................... 93, 104
perceptron
  multi-layer .................... 80
  single-layer ................... 79, 81
phoneme .......................... 88, 96, 97
phonetic symbol system ....... 89
pixel .............................. 68
plot ............................... 62, 64, 84, 88, 93, 94, 105, 107
polynomial recursion ......... 101–102
reference
  Burrus et al. (1994) .... 87, 93, 98
  Haykin (2009) .............. 79–81, 85
  Hunt et al. (2006) 17, 20, 95, 98, 104, 109
  Ingle, Proakis (1997) .... 18, 20
  Knight (2000) ............... 13, 20
  Navakauskas, Paulikas (2004) 87, 98
  Navakauskas, Serackis (2008) 17, 20, 26, 34, 38, 46, 59, 66, 69, 72, 78, 82, 85, 95, 98, 104, 109
Oppenheim, Schafer (1999)... 99, 109
Oppenheim, Schafer (2009)... 21, 23, 34, 36, 46, 49, 56
Proakis, Manolakis (2006)... 22, 34–36, 46
Rosenberg (1971) .......... 100, 109
Rumelhart et al. (1986) .... 80, 85
Semmlow (2004) .......... 72, 78
The MathWorks, Inc. (2010) . 95, 98
The MathWorks, Inc. (2010a) . 24, 26, 27, 34, 37, 39, 46, 52, 56, 60, 66, 69, 78, 82, 86
The MathWorks, Inc. (2010b) 24–27, 34, 37, 39, 46, 49, 52, 56, 59, 60, 66, 69, 78, 81, 82, 86
The MathWorks, Inc. (2011)... 17, 20, 89, 95, 98, 104, 110
reflection coefficient .... 102, 103, 106–107
resample ..................... 59, 63
resampling .................. 57, 59
residuez ..................... 50
response
  frequency .................. 36
  impulse .................. 35, 36, 59
  infinite ................ 47
  infinite ................ 35
  magnitude .............. 38
return ...................... 92
rgb2gray ................... 70, 71
roots ...................... 108

S
s-plane .................... 36
sampling rate ................ 57, 58
sampling frequency 88, 94, 100, 105, 107
segmentation ....... see speech signal
  audio ...................... 57
  glottis ................. 100, 105
  processing
    multirate ............ 59
  spectrum ................ 58
  transmission .......... 58
two-dimensional ........ 67, 68
sim .................. 82–84
size ................... 69, 71, 103
sound ................... 96
speech signal
  analysis ............... 90–95
  modelling ............. 103
  preemphasis .......... 89–90
  segmentation .......... 87–89
  synthesis ............ 104, 107
sqrt ..................... 16, 93
stem .................... 26, 60, 61
subplot ............... 69, 71, 72, 107
system function ...... 101, 105, 106, 108

T
tf2sos .................... 50
tf2zp ...................... 49, 50
title ..................... 88, 93
train ...................... 81–84
transform
  Biliniear ............... 36
discrete Fourier .......... 91
  fast Fourier .......... 91
  short-time Fourier .... 90
two-dimensional discrete Fourier
  67
<table>
<thead>
<tr>
<th><strong>z</strong></th>
<th>Page 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>transforma</td>
<td></td>
</tr>
<tr>
<td>Discrete Fourier</td>
<td>107</td>
</tr>
<tr>
<td>transformation</td>
<td></td>
</tr>
<tr>
<td>discrete Fourier</td>
<td>108</td>
</tr>
<tr>
<td>z</td>
<td>108</td>
</tr>
</tbody>
</table>

**U**

<table>
<thead>
<tr>
<th>uint8</th>
<th>Page 69</th>
</tr>
</thead>
<tbody>
<tr>
<td>unwrap</td>
<td>26, 28</td>
</tr>
<tr>
<td>upfirdn</td>
<td></td>
</tr>
</tbody>
</table>

**W**

<table>
<thead>
<tr>
<th>watershed</th>
<th>Page 77</th>
</tr>
</thead>
<tbody>
<tr>
<td>wavread</td>
<td>60, 61, 65, 88</td>
</tr>
<tr>
<td>while</td>
<td>16</td>
</tr>
<tr>
<td>window</td>
<td>91, 93</td>
</tr>
<tr>
<td>graphical</td>
<td>88, 93, 105</td>
</tr>
<tr>
<td>Hamming</td>
<td>94</td>
</tr>
<tr>
<td>position</td>
<td>97</td>
</tr>
<tr>
<td>symmetric</td>
<td>91</td>
</tr>
<tr>
<td>width</td>
<td>92–93, 96</td>
</tr>
</tbody>
</table>

**window**

| xlabel                   | 88, 93   |

**Y**

| ylabel                   | 88, 93   |
| yulewalk                 | 41, 45   |

**Z**

<table>
<thead>
<tr>
<th>z-plane</th>
<th>36</th>
</tr>
</thead>
<tbody>
<tr>
<td>zeros</td>
<td>92</td>
</tr>
<tr>
<td>zp2sos</td>
<td>50</td>
</tr>
<tr>
<td>zp2tf</td>
<td>37</td>
</tr>
<tr>
<td>zplane</td>
<td>52, 106, 108</td>
</tr>
</tbody>
</table>