České vysoké učení technické v Praze
Fakulta elektrotechnická

BACHELOR THESIS

Digital audio amplifier

Praha, 2010 **Author: Le Huy**

Declaration

I declare that this Bachelor Thesis has been completed by myself independently without outside help and only the defined sources and study aids were used. Sections that reflect the thoughts or works of others are made known through the definition of sources.

Acknowledgements

I would also like to express my sincere thanks to my supervisor: Ing. Petr Kujan, Ph.D. for his constant, and constructive guidance throughout the study. To all other who gave a hand, I say thank you very much!

Abstract

The aim of this bachelor thesis is to familiarize with the problem of computation of digital audio amplifier and to implement the algorithm for solving the optimal pulse width modulated (PWM), odd single-phase bi-level waveform.

The algorithm has been written in C using a freely available GCC compiler.Final application can be used in a microcontroller to drive PWM signal output which is then demodulated providing power output for driving loudspeakers.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

During the last 20 years has appeared a new technology for audio amplification called Class-D amplifiers or switching amplifiers. Its important property is very high power conversion efficiency, usually over 90% vs. 50% for a linear amplifer. This means that more then 90% of the energy consumed goes into powering your speakers and the remaining over 50% is converted into heat. Their foundation is that these amplifiers produce a PWM equivalent (this modulated signal has the same root mean square (RMS) value as the original signal) of the analog input signal which is fed to the loudspeaker via a suitable filter network to block the carrier and recover the original audio.

The main task of this thesis is to implement the algorithm for computation of optimal switching angles for bi-level odd symmetry waveforms in programming language C. Firstly, I would like to introduce the short overview of Class-D amplifiers then I will describe an established method for generating bi-level waveforms with low base-band distortion. The principal problem is to determine the switching times (angles) so as to produce the base-band and not to generate specific higher order harmonics. Such a way, which is possible to separate the undesirable highest harmonics. Finally, I will show step by step procedure to implement the given algorithm.

Chapter 2

Overview of Class-D amplifiers

The "D" in class-D is sometimes said to stand for "digital". This is not correct because the operation of the class-D amplifier is based on either analog or digital principles. The first group of Class-D amplifiers allowing to process the input signal directly in digital form see Figure 2.4 and the second group requires necessary to work with the analog input signal see Figure 2.2. However, the same character for both groups of amplifiers is their fundamental structure and it is displayed in Figure 2.1. That block structure is presented for both types.

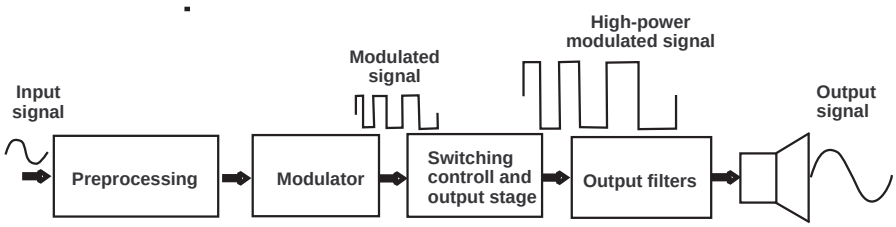

Figure 2.1: Principal block scheme of Class-D amplifiers.

Input signal is typically introduced into the block **Preprocessing**. Here it is possible to realize the needed gain for the analog input signal or to recover the digital input signal. This block often also implements its own impedance of the audio amplifier to separate out the signal source e.g. CD player, PC, etc. . . .

Another important part of class-D amplifiers is **Modulator**. Primary function of this block is a transfer of input signal into PWM signal, that makes a main difference of two variants.

- The classic method of generating the drive signal by the analog principle is to use a differential comparator **C**. One input is driven by the incoming audio signal **Input**, and the other by a **Triangular wave generator** at the required switching frequency. Then the voltage output of the comparator is applied to the input of a complementary common-source MOSFET output stage. Each transistor operates as a switch. A basic analog class-D amplifier is shown in Figure 2.2, and the PWM process is illustrated in Figure 2.3.

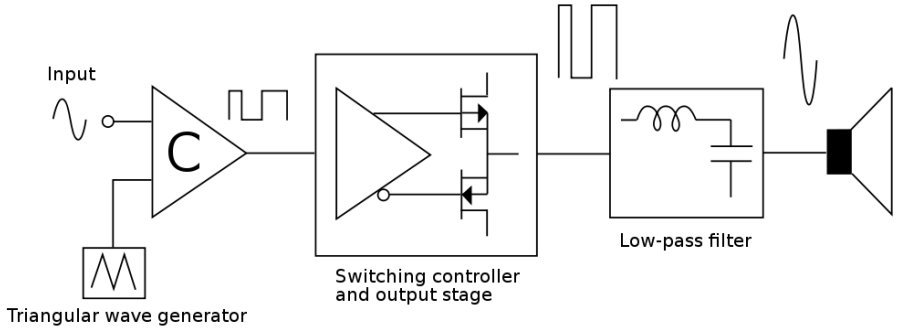

Figure 2.2: Block diagram of an Analog class-D amplifier.(assumed from [7])

The biggest problem of this method is that clearly the triangular wave needs to be linear (i.e., with constant slope) to prevent distortion being introduced at this stage. There are other ways to create the required waveform, such as a sigma-delta modulator, but in the practical reality it never can be achieved. Thereby causing the quality of the processed signal i.e. lower value of distortion.

- In the case of digital solutions can evade this problem, because entire signals processing in digital signal processor (DSP) and it is illustrated in Figure 2.4. For example using specialised microprocessors such as the MSC825X produced by Freescale Semiconductor, the DA8x or TAS3308 from Texas Instruments etc, ... For faster applications often might be used an integrated circuit, the Field-programmable gate array (FPGA) or digital signal processing can be implemented on Embedded Systems using powerful PCs with a Multi-core processor.

Last part of class-D amplifiers is block **Output filters**.The purpose of this block is to get low-frequency useful audio signals (useful information), to reduce unwanted high-frequency spectral components, that are present in the modulated signal and also to improve efficiency. So It is best suited to use here low-pass filter.

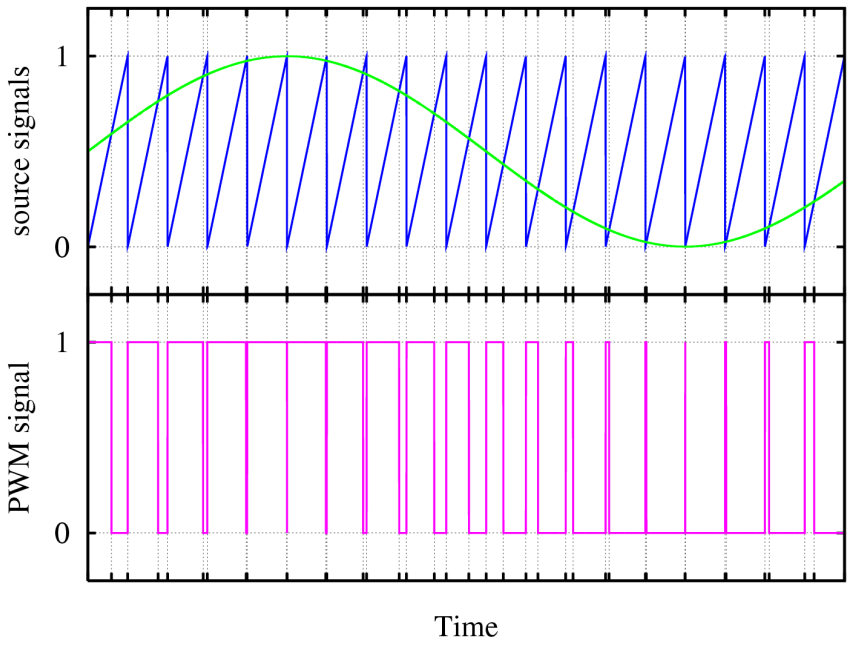

Figure 2.3: A simple method to generate the PWM pulse train corresponding to a given signal is the intersective PWM: the signal (here the green sinewave) is compared with a sawtooth waveform (blue). When the latter is less than the former, the PWM signal (magenta) is in high state (1). Otherwise it is in the low state (0). (assumed from [8])

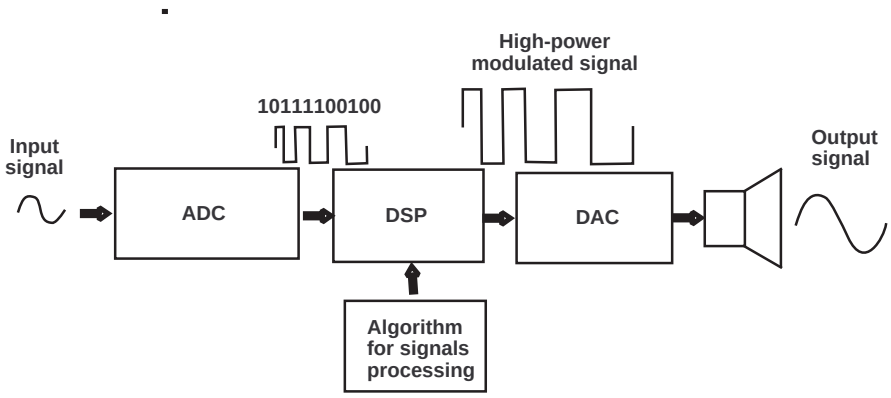

Figure 2.4: Principal block scheme of a Digital Class-D amplifiers.The ADC block is an analog-to-digital converter, the DAC is an digital-to-analog converter

Chapter 3

Optimal odd single-phase bi-level problem

3.1 Optimal PWM problem

As already indicated, the whole process for digital audio signal processing takes place in a DPS to that output signal is modulated PWM. Find an efficient algorithm for DSP is the task of designers. One of the many options can be seen from Figure 3.1. Key issue of the optimal PWM problem is to determine the switching times (angles) so as to produce the signal portion (base-band) and not generate specific higher order harmonics (guard band or zero band). This spectral gap separates the base-band which has to be identical to the required output waveform, from an uncontrolled higher frequency portion. The required output signal can be recovered by means of an analog low-pass filter (LPF) with cutoff frequency in the guard band.

Methods described in this section are based on exploiting appropriate trigonometric transcendental equations that define the harmonic content of the generated periodic PWM waveform $p(t)$ which is equal to required finite frequency spectrum of $f(t)$. The main problem lies in solving these systems of equations.

The solution of the optimal PWM problem is a sequence of switching times $\overline{\alpha}^{\star} = (\alpha_1, \ldots, \alpha_n)$. This sequence is obtained from the solution of the system of equations

$$\begin{aligned} & a_{p0}(\overline{\alpha}) = a_{f0}\ , \\ & \left.\begin{aligned} a_{pk}(\overline{\alpha}) &= a_{fk} \\ b_{pk}(\overline{\alpha}) &= b_{fk} \end{aligned}\right\} \text{ for all } \quad k \in \mathcal{H}_C, \\ & \left.\begin{aligned} a_{pk}(\overline{\alpha}) &= 0 \\ b_{pk}(\overline{\alpha}) &= 0 \end{aligned}\right\} \text{ for all } \quad k \in \mathcal{H}_E, \\ & \text{subject to} \qquad 0 < \alpha_i < T, \end{aligned}$$

where $\overline{\alpha} = (\alpha_1, \ldots, \alpha_n)$ are unknown variables, a_{p0} and a_{pk}, b_{pk} are zeroth and k-th cosine, respectively sine Fourier coefficients of the generated waveform $p(t)$, a_{f0} and a_{fk}, b_{fk} are zeroth and k-th cosine, sine Fourier coefficients of the required

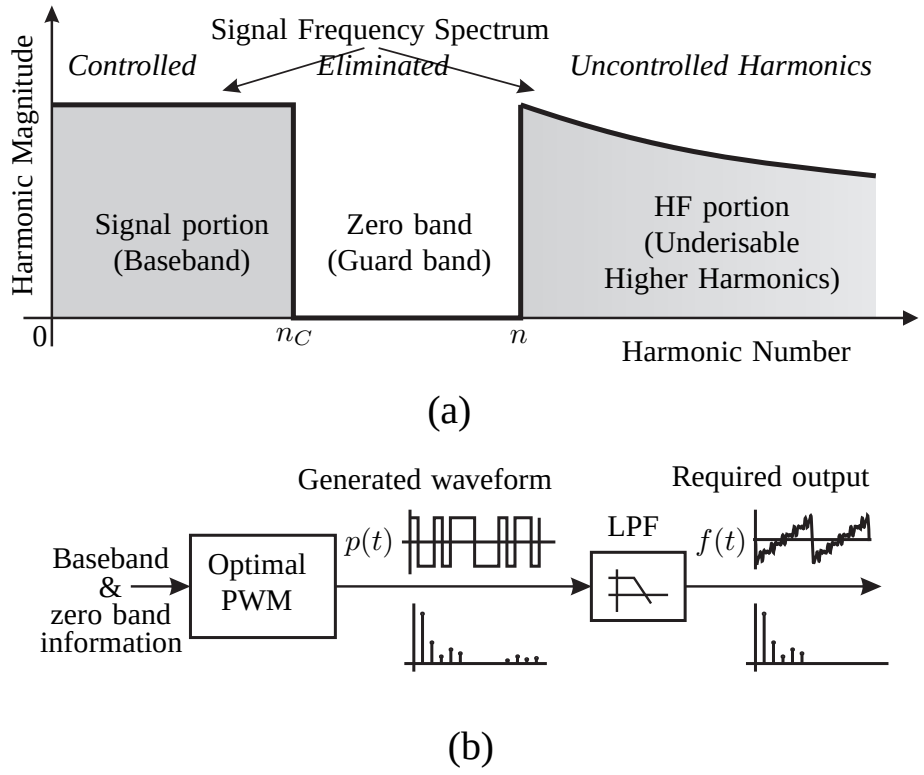

Figure 3.1: (a) Frequency spectrum of a separated base-band signal. The base-band can be recovered by an LPF. (b) Principal scheme for optimal PWM problem.

output waveform $f(t)$. The $\mathcal{H}_C$ is the set of controlled harmonics and the number of elements is n_C. The $\mathcal{H}_E$ is the set of eliminated harmonics and the number of elements is n_E. The number of equations is $n = 1 + 2(n_C + n_E)$.

3.2 Switching odd bi-level PWM Waveforms

The Fourier series of T periodic odd bi-level PWM waveform $p(t)$ with amplitude A (see the Figure 3.2) is sine with the following coefficients

$$b_k = \frac{4A}{k\pi}\left(o_{n+k} + \sum_{i=1}^{n}(-1)^i \cos(\omega k \alpha_i)\right), \qquad (3.2)$$
$$k = 1, 2, \ldots,$$

where $0 < \alpha_1 < \alpha_2 < \cdots < \alpha_n < T/2$ are the unknown switching times and $b_k, k = 1, 2, \ldots, n_C$ on the hand side (RHS) of equations are real numbers defining the required signal $f(t)$ (base-band frequency spectrum). The integer n_C defines the number of controlled harmonics.

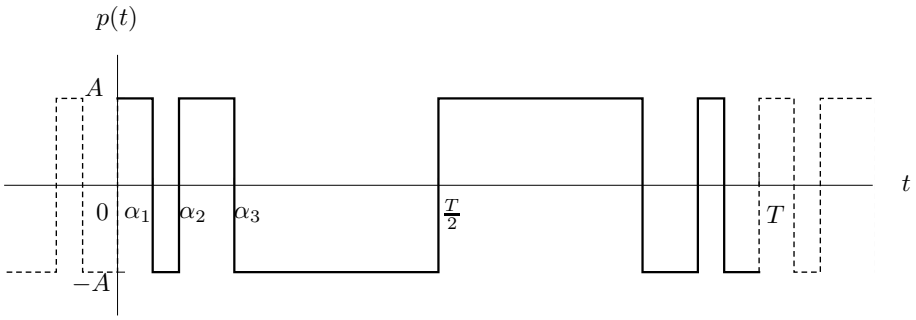

Figure 3.2: Odd bi-level PWM waveform.

Chapter 4

Implementation

4.1 pseudocode algorithm

In this section I use a pseudocode to describe the algorithms. This pseudo-code specifies the form of the input to be supplied and the form of the desired output. Not all numerical procedures give satisfactory output for arbitrarily chosen input. As a consequence, a stopping technique independent of the numerical technique is in corporated into each algorithm to avoid infinite loops.

A pseudo-code algorithm which embodies the above strategy is shown below:

4.1.1 Optimal PWM

Algorithm 1 (OptimalPWM): Compute optimal PWM problem.

Input:

n	... the number of switching times (it is equal to number of controlled harmonics n_C plus number of zero harmonics n_E),
$(b_{f_1}, \ldots, b_{f_{n_C}})$	... the sequence of controlled harmonics,
ω	... frequency,
A	... amplitude of PWM waveform.

Output:

$(\alpha_1, \ldots, \alpha_n)$	... the optimal switching times.

1. Compute the RHS of composite sum of powers $p_i,\ i = 1, \ldots, n$ using

$$p_{2i} = o_n - 2^{-2i}\frac{\pi}{A}\sum_{j=1}^{\underline{K}}\binom{2i}{i-j} j\ b_{f_{2j}}, \tag{4.1a}$$

$$\underline{K} := \begin{cases} i & \ldots\ i < \lfloor n_C/2 \rfloor \\ \lfloor n_C/2 \rfloor & \ldots\ i \geq \lfloor n_C/2 \rfloor \end{cases}, \quad i = 1, 2, \ldots, \lfloor n/2 \rfloor, \tag{4.1b}$$

$$p_{2i-1} = o_{n+1} - 2^{-2i}\frac{\pi}{A}\sum_{j=1}^{\overline{K}}\binom{2i-1}{i-j}(2j-1)\ b_{f_{2j-1}}, \tag{4.1c}$$

$$\overline{K} := \begin{cases} i & \ldots\ i < \lceil n_C/2 \rceil \\ \lceil n_C/2 \rceil & \ldots\ i \geq \lceil n_C/2 \rceil \end{cases}, \quad i = 1, 2, \ldots, \lceil n/2 \rceil. \tag{4.1d}$$

for bi-level PWM odd waveform, where o_n is the odd parity test:

$$o_n = \frac{1-(-1)^n}{2} = \begin{cases} 0 & \text{for even } n, \\ 1 & \text{for odd } n. \end{cases} \tag{4.2}$$

2. Compute composite sum of powers

$$y_1^i + \cdots + y_{\lceil n/2 \rceil}^i - y_{\lceil n/2 \rceil + 1}^i \cdots - y_n^i = p_i, \quad i = 1, \ldots, n \tag{4.3}$$

using Algorithms PadeCSoP.
Set $(\overline{y}^+, \overline{y}^-) =$ PadeCSoP $(p_1, \ldots, p_n)$.

3. **if** $\overline{y}^+ \in R^{(-1,1)} \wedge \overline{y}^- \in R^{(-1,1)}$ **then** continue **else** exit - no exact solution

4. **end if**

5. Set $\overline{y}_s^+ = (y_{s_1}^+, y_{s_2}^+, \ldots, y_{s_{\lceil n/2 \rceil}}^+) = \text{sort}_> \overline{y}^+$, where $y_{s_1}^+ > y_{s_2}^+ > \ldots$
Set $\overline{y}_s^- = (y_{s_1}^-, y_{s_2}^-, \ldots, y_{s_{\lfloor n/2 \rfloor}}^-) = \text{sort}_> \overline{y}^-$, where $y_{s_1}^- > y_{s_2}^- > \ldots$

6. Set $\overline{x} = (x_1, \ldots, x_n) = \text{riffle}(\overline{y}_s^+, \overline{y}_s^-) = (y_{s_1}^+, y_{s_1}^-, y_{s_2}^+, y_{s_2}^-, \ldots)$

7. Return $\overline{\alpha}^\star = (\alpha_1^\star, \alpha_2^\star, \ldots, \alpha_n^\star) = 1/\omega \arccos \overline{x}$

4.1.2 Padé method

Algorithm 2 (PadeCSoP): Compute composite sum of powers for optimal odd PWM problem via solving Hankel linear system (Padé approximation).

Input:

$p_1, \ldots, p_n$ ... the right hand side of composite sum of powers, solved according to (4.1).

Output:

$(\overline{y}^+, \overline{y}^-) = ((y_1, y_2, \ldots, y_{\lceil n/2 \rceil}), (y_{\lceil n/2 \rceil + 1}, y_{\lceil n/2 \rceil + 2}, \ldots, y_n))$
... the solution of composite sum of powers (4.3).

1. Set $p = (-1)^n p$ (* for condition $k \leq \lfloor n/2 \rfloor$ *)
2. Set $k = \lfloor n/2 \rfloor$
3. Compute the moments $\mu_i,\ i = 1, \ldots, n$ according to

$$\mu_0 = 1,\ \mu_i = -\frac{1}{i} \sum_{j=1}^{i} p_j \mu_{i-j} \tag{4.4}$$

4. **if** n is odd integer **then**
5. Solve linear Hankel system for

$$\begin{bmatrix} \mu_0 \ldots \mu_k \\ \vdots \ \ddots \ \vdots \\ \mu_k \ldots \mu_{2k} \end{bmatrix} \cdot \begin{bmatrix} w_{k+1,0} \\ \vdots \\ w_{k+1,k} \end{bmatrix} = - \begin{bmatrix} \mu_{k+1} \\ \vdots \\ \mu_{2k+1} \end{bmatrix} \tag{4.5}$$

6. Solve matrix equation with triangular Hankel matrix

$$\begin{bmatrix} v_{k,k-1} \\ \vdots \\ v_{k,0} \end{bmatrix} = \begin{bmatrix} 0 \ldots \ \mu_0 \\ \vdots \ \ddots \ \vdots \\ \mu_0 \ldots \mu_{k-1} \end{bmatrix} \cdot \begin{bmatrix} w_{k+1,1} \\ \vdots \\ w_{k+1,k} \end{bmatrix} + \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_k \end{bmatrix}$$

7. Set $W_{k+1}(y) = x^{k+1} + \sum_{i=0}^{k} w_{k+1,i} x^i$ and $V_k(y) = x^k + \sum_{i=0}^{k-1} v_{k,i} x^i$
8. Return $(\overline{y}^+, \overline{y}^-) = (\text{roots}(W_{k+1}(y)),\ \text{roots}(V_k(y)))$
9. **else**
10. Solve linear Hankel system for

$$\begin{bmatrix} \mu_1 \ldots \ \mu_k \\ \vdots \ \ddots \ \vdots \\ \mu_k \ldots \mu_{2k-1} \end{bmatrix} \cdot \begin{bmatrix} w_{k,0} \\ \vdots \\ w_{k,k-1} \end{bmatrix} = - \begin{bmatrix} \mu_{k+1} \\ \vdots \\ \mu_{2k} \end{bmatrix}$$

11. Solve matrix equation with triangular Hankel matrix

$$\begin{bmatrix} v_{k,k-1} \\ \vdots \\ v_{k,0} \end{bmatrix} = \begin{bmatrix} 0 \ldots \ \mu_0 \\ \vdots \ \ddots \ \vdots \\ \mu_0 \ldots \mu_{k-1} \end{bmatrix} \cdot \begin{bmatrix} w_{k,0} \\ \vdots \\ w_{k,k-1} \end{bmatrix} + \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_k \end{bmatrix}$$

12. Set $W_k(y) = y^k + \sum_{i=0}^{k-1} w_{k,i} y^i$ and $V_k(y) = y^k + \sum_{i=0}^{k-1} v_{k,i} y^i$
13. Return $(\overline{y}^+, \overline{y}^-) = (\text{roots}(V_k(y)),\ \text{roots}(W_k(y)))$
14. **end if**

For more detail obout this method see [4].

4.2 Implementation in C

Programming language C allows programmers to break up programs into smaller components (such as classes and subroutines) and distribute those components among many translation units (typically in the form of physical source files), which the system can compile separately - here header files (*.h). Thus, completed aplication to compute the optimal switching times of PWM output signal, implemented in programming language contains five header files, main file, Make file, which compiles the program and one text file containing the input, i.e. the sequence of controlled harmonics $(b_{f_1}, \ldots, b_{f_{n_C}})$. In this case I declared $b_{f_{n_C}}$ to equal five. For programming I used many literatures see [2], [5], [6]. The source files as follows.

4.2.1 PWMPowerSum.h

This header file contains functions, that allow to calculate the RHS of composite sum of powers p_i. They are function **pEven( )** for computing even elements of p_i and **pOdd( )** to calculate the other elements. Look at that equations (4.1a) and (4.1c). There I face two problems to be solved. First one is that how to write $\sum$ in C and the second problem of calculating binomial coefficients.

1. The $\sum$ can be implemented in C using one **for loop**. For example

$$\sum_{i=m}^{n} x_i = x_m + x_{m+1} + x_{m+2} + \cdots + x_{n-1} + x_n,$$

and equivalent write in C:

```
int i,sum = 0;
for(i = n; i <= m; i++)
  sum += x[i];
```

2. Recursive formula for binomial coefficients as follows

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \quad \text{for all integers} \quad n, k > 0,$$

with as initial values

$$\binom{n}{0} = 1 \quad \text{for all} \quad n \in \mathbb{N},$$

$$\binom{0}{k} = 0 \quad \text{for all integers} \quad k > 0.$$

It may be re-written

$$\binom{n}{k} = sum_k = \sum_{i=1}^{k} \frac{sum_{i-1}(n-k+i)}{i} \tag{4.6a}$$

$$\text{where} \quad sum_0 = 1$$

Thanks to equation (4.6a) I can directly implement a function **binom(unsigned n, unsigned k)** with two input variables n and k and this function returns required value of $\binom{n}{k}$. The next step necessary to do is that determine values of $\underline{K}$ and $\overline{K}$, which are defined in (4.1b) and (4.1d), then they can be achieved by method **UnderK(int i, int nC)** and **OverK(int i, int nC)**, which returns the value to variable i or nC.

Everything is already available for implementation of function pEven( ) and pOdd( ). Proposed **pEven(double * data,unsigned int num,int Amp,double *Even)** has a parametr *data, which is pointer to array type double and its elements are the sequence of controlled harmonics. Variable num is the number of switching times and Amp is elected value of amplitude of PWM waveform. The routine returns the roots p_{2i} and the *Even part to be added it.

Using same concept could be made the funtion **void pOdd(double * data,unsigned int num,int Amp,double *Odd)** to compute the p_{2i-1} then will add their address into *Odd.

After a call of two functions pEven( ) and pOdd( ) I'm getting the calculated values of p_{2i} and p_{2i+1} . The following function **join(double * joinEven,double * joinOdd, unsigned int num,double *Join)** could join p_{2i} with p_{2i-1} then will add them to *Join. Finally I'm just getting a sequence $(p_1, p_2, \ldots, p_{n-1}, p_n)$.

The next step in strategy computing optimal PWM problem is that finding of composite sum of powers (see (4.3)) using Padé method. To accomplish that I have created three functions i.e. function moments( ), Hankel( ), GaussResol( ) and polyroots( ). Now I would like to describe function of different methods:

4.2.2 Moments.h

In this header file has been located function **moments(double *data, unsigned int num, double *uArray)**, which allows to compute the moments μ_i. The argument *data is input pointer to array containning the p_i from previous item. Variable num is the number of switching times and returned adrress of unknown μ_i could be added into the pointer *uArray. It is according to (4.4).

4.2.3 Hankel.h

In linear algebra, a Hankel matrix is a special case of a square matrix with constant positive sloping diagonals. In mathematical terms is defined as follows:

$$a_{i,j} = a_{i-1,j+1} \tag{4.7}$$

For example, if I want to creat a Hankel matrix of numbers from 1 to 7, I'm getting a 4x4 dimensional matrix and it looks like:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{bmatrix}.$$

It's clear to see one interesting of Hankel matrix. For any sequence of numbers $num \geq 4$ plus num must be odd number is possible to creat some Hankel matrix with dimension $\left\lceil \frac{num}{2} \right\rceil \mathrm{x} \left\lceil \frac{num}{2} \right\rceil$. So it will very easy to fill a Hankel matrix with a array elements in C using only double for loops.

A upper triangular Hankel matrix is a special case of the Hankel matrix,where the entries above the main diagonal are zero. For example upper triangular Hankel matrix of numbers from 1 to 7 has a form:

$$\begin{bmatrix} 0 & 0 & 0 & 4 \\ 0 & 0 & 4 & 5 \\ 0 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{bmatrix}$$

In programming language C can be created by an analogous procedure with the method of Hankel matrix, nevertheless the first $(\left\lceil \frac{num}{2} \right\rceil - 1)$ values are replaced by zeros.

This file contains function **Hankel(double *Udata, unsigned int num, const caseN typ, double ** uHankel)**. It's first parametr *Udata is pointer to input elements, the μ_i. The next parametr is length of a required Hankel matrix, a variable num and it's value equals $\left\lceil \frac{\text{number of elements} \quad \mu_i}{2} \right\rceil$. The variable **typ** offers three distinct options to set a form of Hankel matrix - typ equals 1 for case, when the number of switching times n is odd integer, typ = 2 for case even n and the last case, when the typ equals 3, I can set it like upper triangular Hankel matrix.Then the address of final matrix can be written in **uHankel with dimension $(\left\lceil \frac{\text{number of elements} \quad \mu_i}{2} \right\rceil + 1)\mathrm{x} \left\lceil \frac{\text{number of elements} \quad \mu_i}{2} \right\rceil$. Into the last row of this matrix have been added elements of right hand side of linear Hankel system

4.2.4 GaussResol.h

A part of this Padé method demands solving systems of linear equation see Algorithm 2 equation (4.5) (lines 5 a 10). For finding roots of such systems becomes reality using many different methods like Gaussian elimination, LU decomposition, cholesky decomposition, Levinson recursion etc, ... For simplicity, I chose method Gaussian elimination, that elementary row operations are used to reduce a matrix to triangular form then the next step uses back substitution to find the solution of the system above. As an example look at the following linear system

$$\begin{bmatrix} 1 & -2 & 1 \\ 3 & 2 & -2 \\ -2 & 4 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 2 \end{bmatrix}, \tag{4.8}$$

then it can be rewritten to allowing form:

$$\left[\begin{array}{ccc|c} 1 & -2 & 1 & 5 \\ 3 & 2 & -2 & 4 \\ -2 & 4 & 1 & 2 \end{array}\right]$$

After an elimination,

$$\left[\begin{array}{ccc|c} 3 & 2 & -2 & 4 \\ 0 & 8 & -5 & -11 \\ 0 & 0 & 3 & 12 \end{array}\right]$$

Now to use the back substitution to find the solution of the system. It follows,

$$z = \frac{12}{3} = 4;\ y = \frac{-11 + 5.4}{8} = 1.125;\ z = \frac{4 + 2.4 - 2.1.25}{3} = 3.25$$

Implemented method **GaussResol(double **a, unsigned int n, double *x)** solves linear system by Gausian elimination with input Hankel matrix a, n number of row's matrix a. Roots of linear system will be added into $*x$.

4.2.5 polynsolve.h

Knowledge of algebra can be said that every function $P(x)$, which has the following form

$$P(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i, \tag{4.9}$$

$$\text{where} \quad a_0, a_1, \ldots, a_n \in \mathbb{R}, \text{ respectively in } \mathbb{C} \quad \text{and } a_n \neq 0.$$

It's called polynomial in x of degree n and $a_0, a_1, \ldots, a_n$ are the coefficients of this polynomial. By the way I can rewrite (4.9) into another representation, the factored form of the polynomial simultaneously I set $P(x) = 0$, then I'm getting a polynomial equation and the solutions to the equation are called the roots of the polynomial, that here are the α_i. Note, the α_i in this subsection are not the optimal switching times!

$$P(x) = \underbrace{a_0(x-\alpha_1)^{k_1}\ldots}_{\text{real roots}}\underbrace{(x-\alpha_m)^{k_m}\ldots}_{\text{complex roots}}$$

$$P(x) = a_n(x-\alpha_1)^{k_1}\ldots(x-\alpha_r)^{k_r}\ldots[(x-c_1)^2+d_1^2]^{r_1}\ldots[(x-c_s)^2+d_s^2]^{r_s},$$

where $\alpha_1,\ldots,\alpha_m$ real roots of multiplicity $k_1,\ldots,k_r$ and $(c_1 \pm jd_1),\ldots,(c_s \pm jd_s)$ complex roots of multiplicity $r_1,\ldots,r_s$.

Finding exact roots of polynomials is often impossible to obtain in practice (a problem of continuous mathematics - round off errors in digital computers, a number is not rational etc, ...). So, it is necessary to use a numerical analysis,which are concerned with obtaining approximate solutions while maintaining reasonable bounds on errors. Well-known methods of numerical analysis for finding roots of polynomial are as Newton-Raphson, Muller's method, Laguerre' method etc, ... Each method has its advantages and disadvantages, for more details see [3].

In this part I chose a robust strategy for finding all real and complex roots of real polynomials by C. Bond. The main reasons for choosing this method is its stability and very high accuracy of test results. However it has one restriction that only can

solve a polynomial of order n = 22, although for computation PWM problem with the number of switching times n = 22 is more than enough. Original method was implemented in C++ by its author. So I had to re-write it into C to correspond my aplication. For more information obout this method see [1] A description in pseudo-code of the method as follows,

1. Copy the original (monic) polynomial to $p_a, p_a \longleftarrow P$,
2. Copy polynomial (monic) derivative to $p_b, p_b \longleftarrow p'_a$,
3. Find GCF of p_a/p_b using Euclid's algorithm, $(q, r) = p_a/p_b$,
4. If GCF is 1, submit p_a to estimator/solver, add roots to root list, goto 11
5. Divide GCF out of $p_a, q = p_a/r$,
6. If order of q is less then 3, add roots to root list, goto 10,
7. Submit q to estimator/solver, add roots to root list,
8. If order of r less than 3, add roots to roots list, goto 10,
9. Copy r to p_a,
10. If more rooots to find, goto 2,
11. End.

In this code, P is the original polynomial, n is its order, p'_a is the monic form of p_a derivative and (q, r) respresents the quotient and remainder after division. If the algorithm terminates with $r = 0$ the polynomial representing the greatest common factor(GCF) of the original polynomial and its derivative is in b. Note that monic polynomial with degree n is such polynomial, when its coefficient $a_n = 1$.

Finally the method for searching of roots of a polynomial implemented in C looks as **polyroots(double *a, int n, double *wr, double *wi)**. Its first parametr $*a$ is a pointer to array involving coefficients of the polynomial, note they are sorted from the highest oder to the lowest i.e. $(a_n, a_{n-1}, \ldots, a_1, a_0)$. The second parametr n is a order of the polynomial ekvivalent of the number of switching times. Then real roots of polynomial will be added into a array using pointer $*wr$, similarly complex roots using $*wi$.

4.2.6 OptimalPWM.C

Complete program was written in this file using all header files, which have been described. When the program starts. Firstly, it opens the text file "Input.txt". The sequence of input audio spectrums has been saved here, since takes first five value of them correspond of the sequence of controlled harmonics. Next step with available Input computes the RHS of composite sum of power using method pEven( )and pOdd( ), then they are joined using method join( ). The Values obtained are now

as input values for calculating of the moment μ_i thanks method moments( ). Then it solves a linear Hankel system (4.5)or via method GaussResol( ), while the Hankel matrix is created by Hankel( ). After resolution of matrix equation it based on the values $w_{k,i}$ and $v_{k,i}$, which are coefficients of the apropriate polynomials $W_k(y)$ and $V_k(y)$. Therefore, available method polyroots( ) it easy finds their roots, that are just the unknown composite sum of powers, since this values must be sorted according to item 8. or item 13. in Algorithm 2 and item 5. in Algorithm 1 via function qsort( ). After all the program computes the requested switching times of PWM waveform see item 7. in Algorithm 1. Note, an input sampling frequency w can be written as following

$$w = 2\pi f = \frac{2\pi}{T},$$

where T is a sampling interval of sampling audio signal.

4.3 Test of algorithm

In this section I carry out a verification of the proposed algorithm with some numerical experiment. For example:

$$(b_{f1}, b_{f2}, b_{f3}, b_{f4}, b_{f5}) = (4, -5, 3, 1, -2), A = 10, T = 1,$$

for two case $n = 17$ and $n = 18$, then I obtained their corresponding results, which are imported into the following tables:

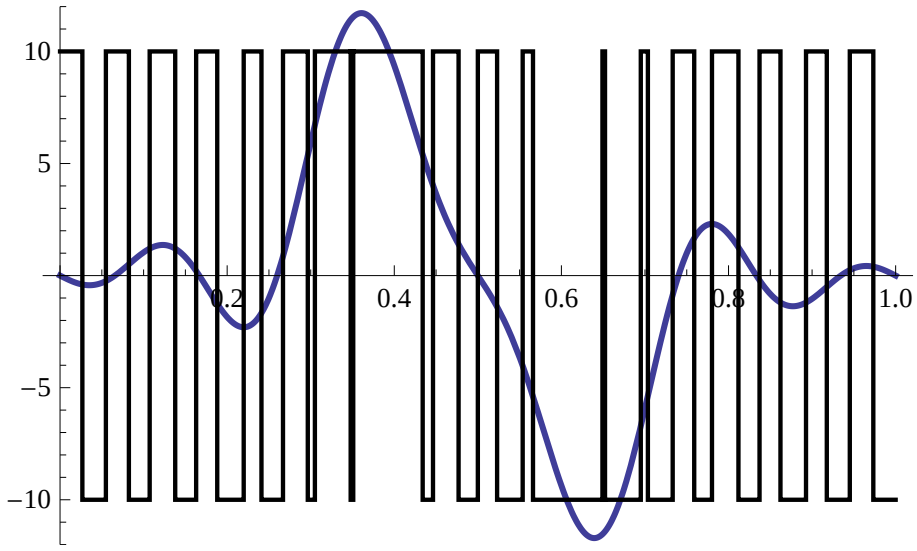

Figure 4.1: Odd bi-level PWM waveform for case n = 17

Figure 4.1 and 4.3 show the input waveform and its computed PWM waveform for both case n -odd and -even. Setting of the values n and T has a big influence on a quality of signal. Indeed, the value of n is inversely proportional to the value of

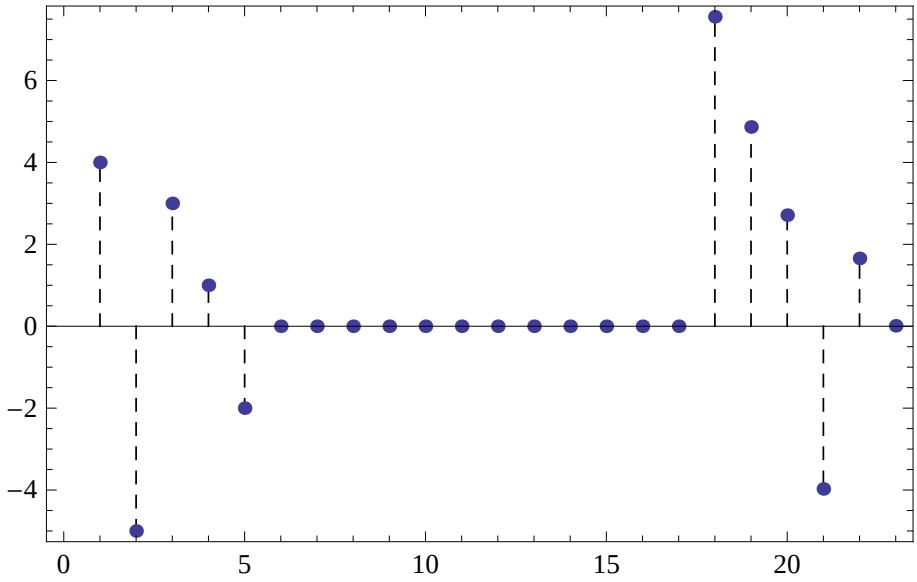

Figure 4.2: Spectrums of Odd bi-level PWM waveform for case n = 17

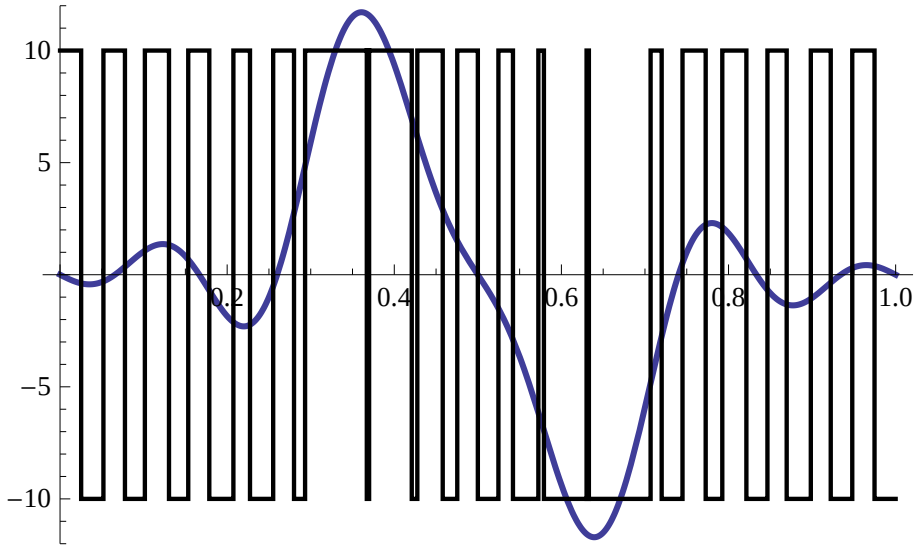

Figure 4.3: Odd bi-level PWM waveform for case n = 18

total harmonic distortion(THD), that says, how the output signal is different than the input. THD is defined:

$$THD(\overline{\alpha})[\%] = 100 \sqrt{\frac{\sum_{i=n_c+1}^{n+N} \left(\frac{a_{p_i}(\overline{\alpha}) + b_{p_i}(\overline{\alpha})}{i} \right)^2}{\sum_{i=1}^{n_c} \left(\frac{a_{p_i}(\overline{\alpha}) + b_{p_i}(\overline{\alpha})}{i} \right)^2}}. \tag{4.10}$$

On the basis of equation (4.10) it's clear to see, that the THD is minimal, if

$$\sum_{i=n_c+1}^{n+N} \left(\frac{a_{p_i}(\overline{\alpha}) + b_{p_i}(\overline{\alpha})}{i} \right)^2 \tag{4.11}$$

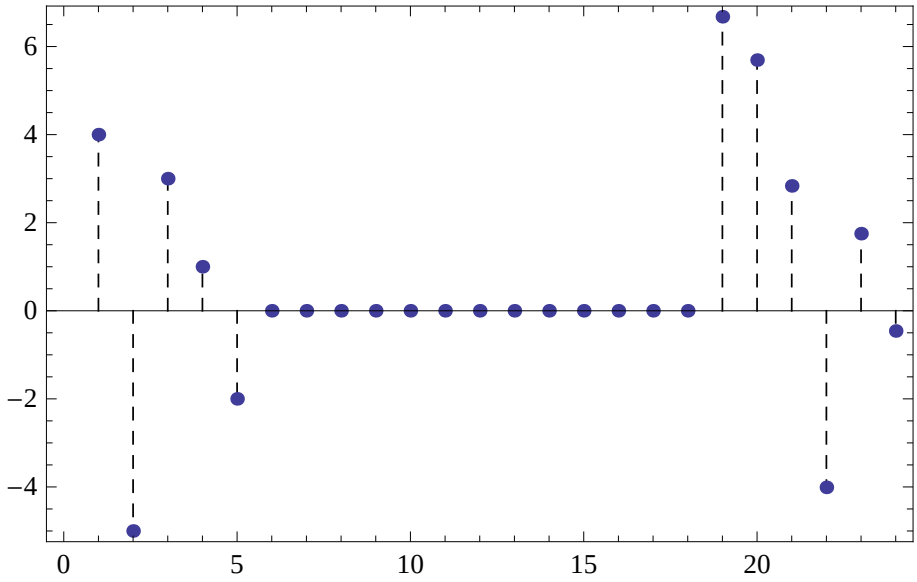

Figure 4.4: Spectrums of Odd bi-level PWM waveform for case n = 18

is minimal, ideally $THD = 0$ if the term (4.11) equals zero, that requires the elements of the guard band and uncontrolled harmonics to be minimal or zero (see the section Optimal PWM problem). In case of setting T is necessary to adhere to the Nyquist-Shannon sampling theorem, which says, that the sampling frequency must be at least two times greater than the highest frequency in the original signal to allow for perfect reconstruction. So in practice it is always absolutely necessary to find a good compromise for them.

The spectrums of the obtained PWM signal, that are shown in 4.2 and 4.4. It really match the input spectrums $(b_{f1,2,3,4,5}) = (4, -5, 3, 1, -2)$ plus the cutoff frequency in the guard band and higher harmonics. After calculating the THD according to the equation (4.10) I obtained: for $n = 17$, $THD = 11.33[\%]$, $n = 18$, $THD = 10.56[\%]$.

Table 4.1: The partial results for case n = 17

i	p_{f_i}	μ_i	w_i	v_i	y_i	α_i	b_{p_i}
0	-	1	-	-	-	-	-
1	-0.314	-0.314	-0.002	-0.467	0.985	0.026	4
2	1.392	0.745	0.031	-1.466	-0.989	0.054	-5
3	-0.412	-0.361	0.055	0.590	0.868	0.082	3
4	1.353	0.658	-0.445	0.594	-0.915	0.107	1
5	-0.368	-0.362	-0.255	-0.163	0.647	0.137	-2
6	1.309	0.602	1.677	-0.073	-0.592	0.162	0
7	-0.317	-0.351	0.355	0.008	0.379	0.188	0
8	1.274	0.559	-2.260	0.001	-0.287	0.219	0
9	-0.276	-0.336	-0.153	-	0.056	0.241	0
10	1.248	0.524	-	-	0.941	0.266	0
11	-0.242	-0.322	-	-	0.781	0.296	0
12	1.227	0.495	-	-	-0.576	0.293	0
13	-0.216	-0.308	-	-	0.188	0.304	0
14	1.211	0.469	-	-	-0.338	0.3477	0
15	-0.195	-0.295	-	-	-0.105	0.433	0
16	1.197	0.447	-	-	-0.943	0.446	7.554
17	-0.178	-0.284	-	-	0.520	0.476	4.864

Table 4.2: The partial results for case n = 18

i	p_{f_i}	μ_i	w_i	v_i	y_i	α_i	b_{p_i}
0	-	1	-	-	-	-	-
1	0.685	-0.685	0.001	-0.413	0.947	0.025	4
2	0.392	0.038	0.018	-2.233	-0.988	0.052	-5
3	0.587	-0.114	-0.020	0.896	0.803	0.077	3
4	0.353	0.028	-0.345	1.637	-0.898	0.102	1
5	0.631	-0.077	0.188	0.152	0.570	0.130	-2
6	0.309	0.036	1.426	-0.413	-0.671	0.153	0
7	0.682	-0.066	-0.429	0.152	0.264	0.178	0
8	0.274	0.040	-2.084	0.015	-0.268	0.207	0
9	0.723	-0.059	0.272	-0.004	-0.031	0.227	0
10	0.248	0.040	-	-	0.883	0.255	0
11	0.757	-0.322	-	-	-0.965	0.280	0
12	0.227	0.0402	-	-	0.682	0.293	0
13	0.783	-0.051	-	-	-0.683	0.369	0
14	0.211	-0.048	-	-	0.433	0.367	0
15	0.804	-0.295	-	-	-0.187	0.420	0
16	0.197	0.038	-	-	0.142	0.427	0
17	0.822	-0.045	-	-	0.987	0.457	6.676
18	0.185	0.037	-	-	-0.879	0.475	5.692

Chapter 5

Conclusion

This thesis deals with the solution of the optimal PWM problem, i.e. how to determine the switching times of PWM modulated output signal so that this signal shoudl have the same spectrums as the spectrums of original signal and in addition n harmonics zero. This was achieved using an algorithm that is described in Chapter 3 and it was implemented in programming language C see Chapter 4. Finally, I verified functionality of this procedures graphically in program Mathematica using the application of my supervisor.

However, the problem can be resolved through other means and using other available public libraries, but most of them are always multiple function, universal and not special for this problem. By the way, the next reason especially to be in control of the aplication, that is very easy to compile with a compiler and latterly can be used on a FPGA using an embedded microcontroller or on a general embedded systems. To deal with specific types microcontroler is just beyond the framework of this topic so I would like to leave it as future work.

Bibliography

[1] C. Bond. A robust strategy for finding all real and complex roots of real polynomials ,http://www.crbond.com[on-line]. 2003.

[2] I. Horton. *Beginning C From Novice to Professional, Fourth Edition*. Apress, 2006.

[3] D. E. Knuth. *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998.

[4] P. Kujan. Optimal odd multilevel problem - homepage, http://support.dce.felk.cvut.cz/pub/kujanp/software.html [on-line], 2009.

[5] B. Mann. *C pro mikrokontrolry*. BEN- technick literatura, 2003.

[6] B. W. K. R. Pike. *The Practice of Programming*.

[7] Wikipedia. Class d amplifier, http://en.wikipedia.org/wiki/Class_D_Amplifier [on-line]. 2010.

[8] Wikipedia. Pulse-width modulation, http://en.wikipedia.org/wiki/Pulse-width_modulation [on-line]. *Wikipedia*, 2010.

Appendix A

CD-ROM OptimalPWM

In addition a CD

To this work is accompanied by a CD, which preserves the source.

- Files C: The sources code of aplication were stored in this directory.
- Files text: The final document written in LaTeX can be found here.