

EVOLUTION OF FILTER ORDER EQUATIONS FOR LINEAR-PHASE FIR FILTERS USING GENE EXPRESSION PROGRAMMING

David González Muñoz, Oscar Gustafsson, and Lars Wanhammar

Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden,
{davgo210, oscarg, larsw}@isy.liu.se

ABSTRACT

Estimation of the minimum filter order for linear-phase FIR filters is commonly performed during the design of DSP systems. In this work gene expression programming is used to discover new equations for the linear-phase FIR filter order. The results are shown to be as least as accurate as previously proposed estimates.

1. INTRODUCTION

Frequency selective digital filters are key components in many DSP systems. One class of digital filters is finite-length impulse response (FIR) filters. The main advantages of FIR filters are that they can be designed with exact linear-phase and that they are inherently stable when realized nonrecursively. The transfer function for an FIR filter is

$$H(z) = \sum_{i=0}^N h_i z^{-i} \quad (1)$$

where N is the filter order.

Assume a low-pass filter with passband edge, $\omega_c T$, stopband edge, $\omega_s T$, passband ripple, δ_c , and stopband ripple, δ_s . These specifications are illustrated in Fig. 1. The required filter order for a linear-phase FIR filters with these specifications can be estimated as [1]

$$N \approx 2\pi \frac{-20 \log_{10}(\sqrt{\delta_c \delta_s}) - 13}{14.6(\omega_s T - \omega_c T)} \quad (2)$$

Here it is assumed that a minimax criterion is used in the design. This type of filter can be designed using either the Remez exchange algorithm [2]–[5] or linear programming [5],[6]. More accurate estimations of the filter order are proposed in [7]–[9].

In this work we use gene expression programming [10]–[11], an automated method for discovering equations, to evolve novel equations for the linear-phase FIR filter order. Accurate filter order estimations are important during the system design phase of DSP systems as the implementation complexity is roughly proportional to the filter order.

2. GENE EXPRESSION PROGRAMMING

Gene expression programming (GEP) is an algorithm that belongs to the class of Genetic Algorithms, as well as its predecessors, genetic algorithms (GAs) and genetic programming (GP). All of them use populations of individuals, select the individuals according to fitness, and introduce genetic variation using one or more genetic operators. It is the nature of the individuals what differentiates these algorithms: in GAs the individuals are symbolic strings of fixed length (chromosomes); in GP they are non-linear entities of different sizes and shapes (parse trees); and in GEP, non-linear entities of different sizes and shapes (expression trees) as well, but these complex entities are encoded as simple strings of fixed length (chromosomes).

In contrast to GAs and GP there are never an invalid expression tree (program) using GEP. In GP, most modifica-

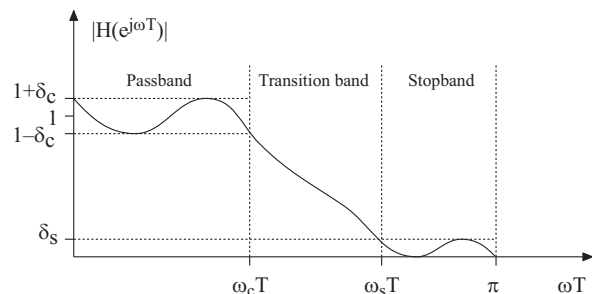


Figure 1. Specifications for a lowpass FIR filter.

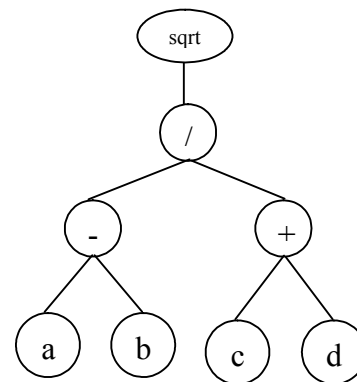


Figure 2. Expression tree for the equation in (3).

tions made on parse trees result in invalid structures. The fact is that only a very limited number of modifications can be made on GP parse trees in order to guarantee the creation of valid structures. The problem with this system is in two ways: it uses a huge amount of computational resources editing the illegal structures and certain efficient search operators, such as point mutation, cannot be used.

2.1. The Genome

The GEP genome consists of a linear, symbolic string of fixed length composed of one or more genes. GEP chromosomes can represent expression trees with different sizes and shapes despite their fixed length.

For example, consider the following expression:

$$\sqrt{\frac{a-b}{c+d}} \quad (3)$$

It can also be represented as a diagram or expression tree (ET) as shown in Fig. 2, where Sqrt represents the square root function. The corresponding genome is shown in Table 1.

Table 1. Example genome corresponding to (3).

Position	1	2	3	4	5	6	7	8	9
Content	Sqrt	/	-	+	a	b	c	d	a
	Head				Tail				

This notation differs from both the postfix and prefix representations used in different GP implementations with arrays or stacks.

The start point of a gene is always the first position, the termination point does not always coincide with the last position due to there are usually non-coding regions downstream of the termination point. These non-coding regions do not interfere with the product of expression but play an important role for evolution. The function of the non-coding regions at the end of a chromosome is to allow modification of the genome using several genetic operators without restrictions, always producing syntactically correct programs.

The reason that GEP always produces genes with a corresponding valid expression tree is that the genes of GEP are composed of a head and a tail. The head contains elements representing both functions and terminals, whereas the tail contains only terminals. For each problem, the length of the head h is chosen, whereas the length of the tail t is a function of h and the maximum number of for a function n (called maximum arity):

$$t = h(N - 1) + 1 \quad (4)$$

Hence, by choosing the tail size according to (4), all operators will have enough input arguments. In Table 1 the head and the tail are marked. The non-coding region is here composed of a single terminal node in position 9.

Despite its fixed length, each gene has the potential to represent expression trees of different sizes and shapes, being the simplest composed of only one node (when the first element of a gene is a terminal) and the largest composed of as many nodes as the length of the gene (when all the elements of the head are functions with maximum arity). Thus, in GEP, what varies is number of nodes used in the expression tree.

GEP chromosomes are usually composed of more than one gene of equal length. For each problem, the number of genes, as well as the length of the head, are chosen a priori. Each gene codes for a sub-ET and the sub-ETs interact with one another forming a more complex entity. The different sub-ETs are linked together by a particular linking function (addition, subtraction, multiplication or division).

To express fully a chromosome, the information concerning the kind of interaction between the sub-ETs must also be provided. Consequently, the linking function is chosen a priori.

2.2. Example

To illustrate how the genome and the expression tree is connected, the expression tree in Fig. 2 will be derived from the genome in Table 1. First, the start of a gene corresponds to the root of the expression tree (the root is at the top of the tree, though), forming this node the first line of the expression tree.

Second, depending on the number of arguments of each element (functions may have a different number of arguments, whereas terminals have an arity of zero), in the next line are placed as many nodes as there are arguments to the elements in the previous line.

Third, from left to right, the new nodes are filled, in the same order, with the elements of the gene, as shown in Fig. 5.

This process is repeated until a line containing only terminals is formed leading to the expression tree in Fig. 2. With this step, the expression tree is complete as the last line contains only nodes with terminals. This is due to the structure with a head and a tail.

2.3. Genetic Operators

In GEP, individuals are selected according to fitness by roulette-wheel sampling to reproduce with modification, creating the necessary genetic diversity allowing for adaptation in the long run [10].

All the genetic operators (mutation, transposition and recombination) randomly pick up the chromosomes to be subjected to a certain modification. However, except for mutation, each operator is not allowed to modify a chromosome more than once. Thus, a chromosome might be randomly chosen to be modified by more than one genetic operator at a time.

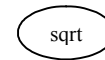


Figure 3. Initial expression tree obtained by the first row (first element of the gene in Table 1).

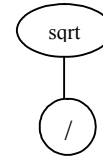


Figure 4. Expression tree obtained after the two first rows are constructed (two first elements of the gene in Table 1).

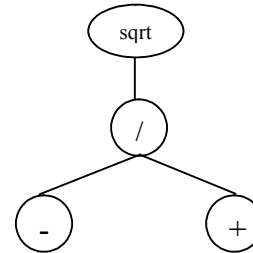


Figure 5. Expression tree obtained after the three first rows are constructed (four first elements in Table 1).

In GEP, mutations can occur anywhere in the chromosome. However, the structural organization of chromosomes must be preserved. Thus, in the heads, any symbol can change into another (function or terminal), in the tails, terminals can only change into terminals. This way, the structural organization of chromosomes is maintained, and all the new individuals produced by mutation are structurally correct programs.

The most important genetic operator for GEP problems is point mutation [10]. When the chromosome is replicated, the genetic information is passed on to the next generation. Sometimes the sequence of the daughter chromosome differs from that of the mother in one or more points due to a mismatched nucleotide has been introduced in the newly synthesized strand. In GEP, most mutations have a profound effect in the structure and function of expression trees.

The second most important genetic operator according to the results obtained is recombination. During recombination, two chromosomes are paired and exchange some material between them, forming two new daughter chromosomes. However, a fragment of a particular gene occupying a particular position in the chromosome is never exchanged for a fragment of a gene in a different position.

The last genetic operator is transposition. Transposable genetic elements are genes that can move from place to place within the chromosome. In GEP, transposable elements were chosen to transpose only within the same chromosome and they might be entire genes or fragments of a gene, without requirements for particular identifying sequences. The transposable element is copied in its entirety at the target site and deleted in the place of origin. Whereas in fragment transposition the donor sequence stays unchanged, usually producing two homologous sequences resident in the same chromosome.

3. SOLUTION APPROACH

To solve our GEP problem the software Automatic Problem Solver (APS) are used. As an input a number of filter specifications were provided. Each with four input parameters, passband and stopband ripples and paddband and stopband edges. In addition to that, a column with the minimum filter order for a linear-phase FIR filter that satisfies those requirements.

The input file were generated by iterating the `firpm` command in MATLAB™ with varying filter orders. As the cornerstone for evolving a good solution is the input data set, it really pays to take a good look at the data before embarking on a complex, usually time consuming modelling process. The data set should be well balanced and the preparations must be done before loading the data into APS, although APS helps us finding missing and invalid data. These considerations were taken into account when developing the MATLAB file and when choosing the number of samples for training.

To determine the settings to be used in APS, several hundred runs were made. For each parameter a number enough or runs were done to test which value lead to the best result. To determine the set of allowed operators to use, previous work in the area were used as inspiration [9].

After these runs it was decided to use a chromosome with four genes that were linked using addition, i.e., the results of each gene were accumulated. Using these settings several solutions were evolved in APS.

4. RESULTS

After each run of APS the best evolved equation can be obtained. The four best results were kept and used in the comparisons in this section.

In Fig. 6, the estimated order for ten random filters are shown using the four evolved equations. The required order were obtained by running the Remez exchange algorithm on the problem iteratively with varying filter order. From here it can be seen that all the evolved equations gives reasonable estimates. For most cases the first and second equation are the best. Using the best evolved equation, a similar comparison with previously proposed estimations are shown in Fig. 7. Ichige denotes the work in [9], Kaiser the work in [1], and `firpmord` is the function in the latest Matlab™ Signal Processing Toolbox.

Figures 8 and 9 show the estimated filter order using the best evolved equation for varying passband edge and varying ripple, respectively. In Fig. 9 it is clear that there is a deviation for ripple smaller than approximately 0.003. This is due to the training data not including such small ripples. Hence, the evolved equations gives good estimated in the range of the training data.

Here, it is evident that, with exception for small ripples due to the fact mentioned above, our evolved estimation is approximately as good as the previously proposed ones.

As the evolved filter order equation have four genomes and an additive linking function it can be written as

$$N = N_1 + N_2 + N_3 + N_4 \quad (5)$$

The evolved N_1 to N_4 are given in (7)–(10), respectively. Note that the passband and stopband angles in the filter order equation are the normalized angles $\omega_c = \omega_c T / \pi$ and $\omega_s = \omega_s T / \pi$.

There are two main observations that can be made from (7)–(10). First, from (7) it is clear that the GEP method do not provide any simplifications. The expression

$$\frac{\omega_c \delta_s}{\frac{\omega_c}{\delta_s^2}} \quad (6)$$

in (7), can easily be simplified to $1/\delta_s$. For GEP the simplification is not better than (6) as the approximations are as close. Second, interestingly enough both (9) and (10) have a denominator of $\omega_s - \omega_c$. This is similar to (2) and follows the “rule” that the filter order is inversely proportional to the passband.

Also, note that there are no numerical constants in the expressions. This is a choice when configuring the GEP runs. Generally, the search process is more complex when numerical constants are included [10].

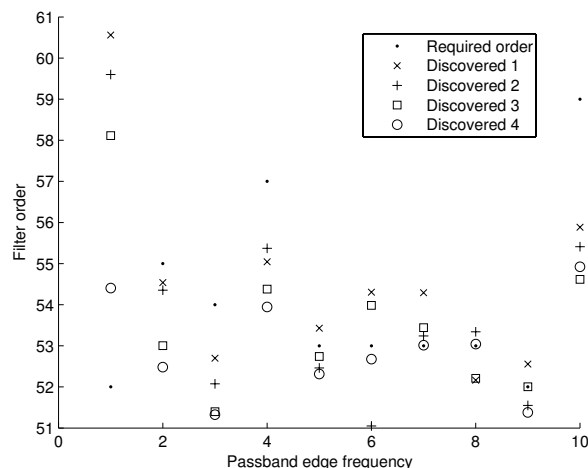


Figure 6. Estimated filter order for ten random filters using the evolved equations.

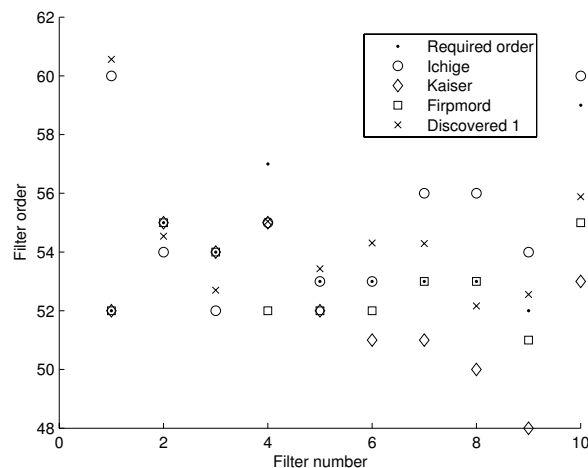


Figure 7. Estimated filter order for ten random filters comparing the best evolved equation with previously proposed.

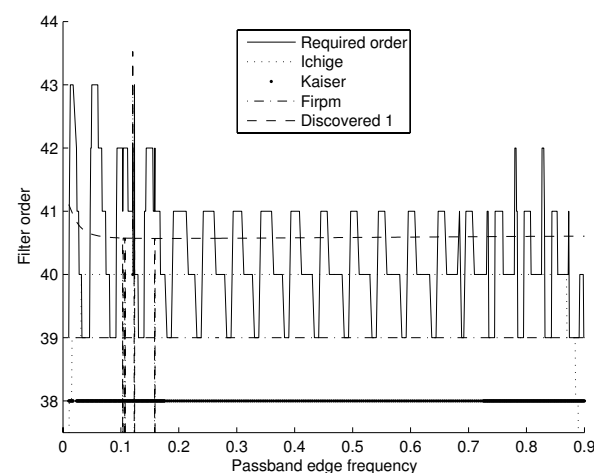


Figure 8. Estimated filter order for varying passband edge (normalized by π) comparing the best evolved equations with previously proposed. $\omega_s T = \omega_c T + 0,2\pi$, $\delta_c = \delta_s = 0,01$

$$N_1 = \delta_s \delta_c \left(\frac{\left(\frac{\omega_c}{\delta_s} \right)^{\delta_s} + \omega_s}{\omega_s} \right) \left(\left(\frac{\omega_s - \omega_c}{\delta_s - \omega_s} \right) \operatorname{atan}(\delta_s - \omega_s) - \frac{\omega_c \delta_s}{\delta_s^2} - \omega_s \right) \quad (7)$$

$$N_2 = \frac{\operatorname{atan}(\delta_s - \delta_c^{\omega_c} \omega_c \omega_s) - \log(\delta_c)}{\omega_s (\delta_s (\omega_s - \omega_c + \delta_c))^r}, \quad r = \frac{\delta_s}{\operatorname{atan}(\delta_c)} \left(\frac{\omega_c}{\delta_c} \right)^{\omega_c \omega_s} \quad (8)$$

$$N_3 = \frac{\frac{\delta_c \delta_s}{\left(\frac{\delta_s}{\frac{\delta_c}{\omega_c} - \omega_c^{\omega_s} \delta_c} \right) \left(\omega_c - \delta_s + \frac{\omega_s^2}{\delta_s} \right) - \omega_s} - \log(\delta_s)}{\omega_s - \omega_c} \quad (9)$$

$$N_4 = \frac{\operatorname{atan} \left(\delta_s + \operatorname{atan}(\delta_c) \left(\frac{\omega_c}{\omega_s} \right)^{\frac{\delta_c}{\omega_c}} - \frac{\left(\frac{\omega_s}{\omega_c} \right) \operatorname{atan}(\delta_s)}{\delta_s^{\delta_s}} \right) - \log(\delta_c)}{\omega_s - \omega_c} \quad (10)$$

REFERENCES

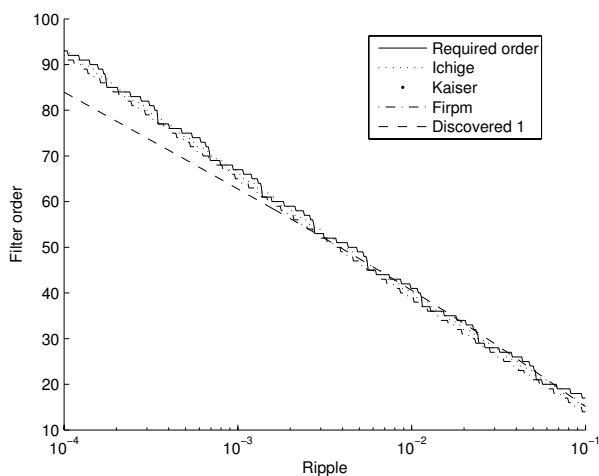


Figure 9. Estimated filter order for varying ripple comparing the best evolved equations with previously proposed.

$$\omega_c T = 0,2\pi, \quad \omega_s T = 0,3\pi, \quad \delta_c = \delta_s.$$

5. CONCLUSIONS

In this work gene expression programming was used to evaluate novel equations for the minimum filter order estimation of linear-phase FIR filters. It was shown that the obtained equations were competitive with previously given estimations. It was also identified that the range of the training data, not surprisingly, were very important to obtain good results.

Only lowpass filters were covered in this work. However, it should be possible to apply the same methodology to high-pass, bandpass, and bandstop filters as well. Furthermore, filter order estimations for minimum phase FIR filters should also be evaluated.

- [1] J. F. Kaiser, "Nonrecursive digital filter design using I_0 -sinh window function," in *Proc. IEEE Int. Symp. Circuits Syst.*, Apr. 1974, pp. 20–23.
- [2] T. W. Parks and J. H. McClellan, "Chebyshev approximation for nonrecursive digital filters with linear phase," *IEEE Trans. Circuit Theory*, vol. 19, pp. 189–194, Mar. 1972.
- [3] L. R. Rabiner and O. Herrmann, "On the design of optimum FIR low-pass filters with even impulse response duration," *IEEE Trans. Audio Electroacoust.*, vol. 21, pp. 329–336, Aug. 1973.
- [4] J. H. McClellan, T. W. Parks, and L. R. Rabiner, "A computer program for designing optimum FIR linear phase digital filters," *IEEE Trans. Audio Electroacoust.*, vol. 21, pp. 506–526, Dec. 1973.
- [5] L. Wanhammar and H. Johansson, *Digital Filters*, Linköping University, 2002.
- [6] L. R. Rabiner, "The design of finite impulse response digital filters using linear programming techniques," *Bell Sys. Tech. J.*, vol. 51, no. 6, pp. 1177–1198, July–Aug. 1977.
- [7] L. R. Rabiner, "Approximate design relationships for low-pass digital filters," *IEEE Trans. Audio Electroacoust.*, vol. 21, pp. 456–460, Oct. 1973.
- [8] O. Herrmann, L. R. Rabiner, and D. S. K. Chan, "Practical design rules for optimum finite response lowpass digital filters," *Bell Syst. Tech. J.*, vol. 52, no. 6, pp. 769–799, July–Aug. 1974.
- [9] K. Ichige, M. Iwaki, and R. Ishii, "Accurate estimation of minimum filter length for optimum FIR digital filters," *IEEE Trans Circuits Syst.–II*, vol. 47, no. 10, pp. 1008–1016, Oct. 2000.
- [10] C. Ferreira, *Gene Expression Programming, Mathematical Modeling by an Artificial Intelligence*, Angra do Heroísmo, Portugal, 2002.
- [11] C. Ferreira, "Gene expression programming: A new adaptive algorithm for solving problems," *Complex Systems*, vol. 13, no. 2, pp. 87–129.