Princess Sumaya University for Technology جامعة الأميرة سميّة للتكنولوجيا

Department of Computer Engineering

Emerging Optimization Techniques

FIR Filter Design Using Genetic Algorithms with MATLAB

Name: Mohammed Salama Ibrahim

Student ID: 20228187

Submission Date: Saturday, June 8th 2024

Abstract

This project discusses the evaluation of FIR filter coefficients using genetic algorithm, where multiple parameters are used to evaluate the filter coefficients given some design specifications in comparison with the famous windowing method as well as the Parks-McClellan algorithm as benchmarks. Several settings for the genetic algorithm have been tested and the optimal settings for this problem were obtained with statistical significance. Multiple runs have been conducted and the results had few discrepancies between them, showing the reliability of the use of genetic algorithm and chosen settings.

1. Introduction

Filtering is a critical part in a plethora of uses in analog and digital circuits and electronics, and the implementation of filters has been a long-time challenge in the field of signal processing, either using active or passive electronic components. Since filters are mathematically-ideal concepts, their physical realization with specific design requirements remains a challenge to this day, where abiding by such involves adding more hardware and resource complexity to the design, to compensate for the tradeoff from mathematical models and physical system requirements.

Filters are devices that remove unwanted components or features from a signal in an electric or electronic circuit. This amounts for the passing (and rejection) of certain frequencies of a signal. Ideally, the terms passing and rejection of some frequencies from a signal are defined by multiplying the frequency response of that signal by 1 and 0, respectively. Mathematically, we can define the filtered signal $X_f(\omega)$ as having the original signal $X(\omega)$ passed within a range of frequencies Ω_1 (called the passband region) and rejected within another range Ω_2 (called the stopband region). Briefly, we can say

$$X_f(\omega) = \begin{cases} X(\omega), & \omega \in \Omega_1 \\ 0, & \omega \in \Omega_2 \end{cases}$$

We can factor out the original signal $X(\omega)$ and define the piecewise function as the filter characteristic, as follows

$$X_f(\omega) = X(\omega) \begin{cases} 1, & \omega \in \Omega_1 \\ 0, & \omega \in \Omega_2 \end{cases} = X(\omega)H(\omega)$$

Hence

$$H(\omega) = \begin{cases} 1, & \omega \in \Omega_1 \\ 0, & \omega \in \Omega_2 \end{cases}$$

Ideally, this model assumes a sharp transition between the borders of regions Ω_1 and Ω_2, which begs for an instantaneous transition between one frequency to another, and this physically requires an infinite amount of energy. To that end, practical filters are implemented with a tradeoff of some filter characteristics, such as the transition region(s), passband ripple (variation around unity) and stopband attenuation (minimum value), amongst other things as well. The tradeoff between such characteristics is studied through a variety of methods and theories.

For the purpose of digital signal processing, the most common type of filters used in digital systems is that of Finite Impulse Response (FIR) filters. FIR filters are characterized by having a finite duration response to an impulse input, meaning they settle to zero in a finite amount of time. FIR filters are widely used due to their inherent stability, linear phase properties, and flexibility in designing various filter responses.

Characteristics of FIR Filters

Linearity and Time-Invariance: FIR filters are linear time-invariant (LTI) systems, meaning their output for a given input does not change over time, and the principle of superposition applies.

Impulse Response: The impulse response of an FIR filter is finite and determined entirely by its coefficients. An impulse input produces a finite series of outputs that eventually return to zero.

Stability: FIR filters are inherently stable because their output is a weighted sum of a finite number of past inputs.

Linear Phase Response: FIR filters can be designed to have a linear phase response, which means that all frequency components of the input signal are delayed by the same amount of time, preserving the wave shape of signals within the passband.

Mathematical Representation

The output $y[n]$ of an FIR filter is given by the convolution of the input signal $x[n]$ with the filter's impulse response $h[n]$ of order $M-1$ and M coefficients

$$y[n] = \sum_{k=0}^{M-1} b_k x[n-k]$$

The implementation of such filters is used using a series of buffers, gain blocks and adders as shown in Figure 1 below [1].

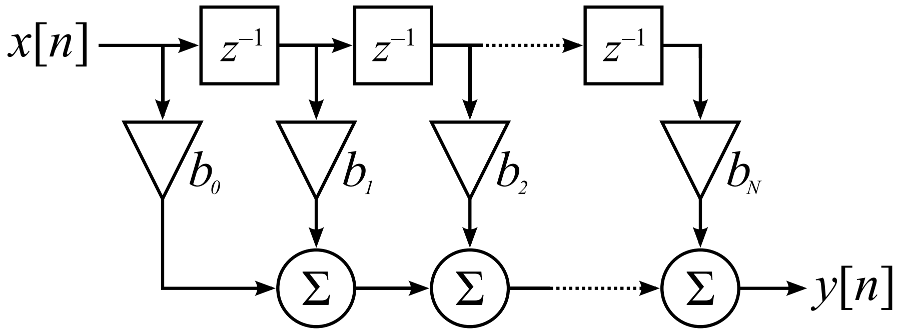


Figure 1: FIR filter block diagram.

To return the impulse response, we apply an impulse signal to the input, where we obtain

$$h[n] = \sum_{k=0}^{M-1} b_k \delta[n-k] = \begin{cases} b_n, & 0 \le n \le M-1 \\ 0, & \text{otherwise.} \end{cases}$$

The coefficients b_n are what represent the FIR filter characteristic, and the search for such coefficients represents the purpose and endeavor of our project. For the simplicity of our project, we shall only tackle designing **lowpass filters,** while the other filter types can be derived without hassle.

Problem Statement and Motivation

The purpose of this project is to attempt to solve one of the most common problems in digital signal processing, which is the design of *digital filters*, which can be used in a variety of applications in digital electronic circuits to operate on and process discrete-time signals used in power electronics, telecommunication systems, control systems, computers and many more devices. The filters we will be considering for this project are called *finite-impulse response* (FIR) filters, which are represented using a finite number of "taps" or time delays in their impulse response representation and are finite in time, meaning that they truncate after a certain number of taps, contrary to another type called *infinite-impulse response* (IIR) filters, whose impulse responses are infinite in duration and can be matched to analog filters [2]. The choice of the finite impulse response as an optimization problem is the presence of a trade-off between a number of quantities and attributes of the filter, including but not limited to the gain, passband ripple, stopband ripple and transition band. The presence of these quantities can create a need for having the least possible compromise between these attributes, without sacrificing a lot of the desired performance and functionality of the filter. The frequency response of the digital filter and its parameters are shown in Figure 2 below, indicating the parameters and their effect on the filter response.

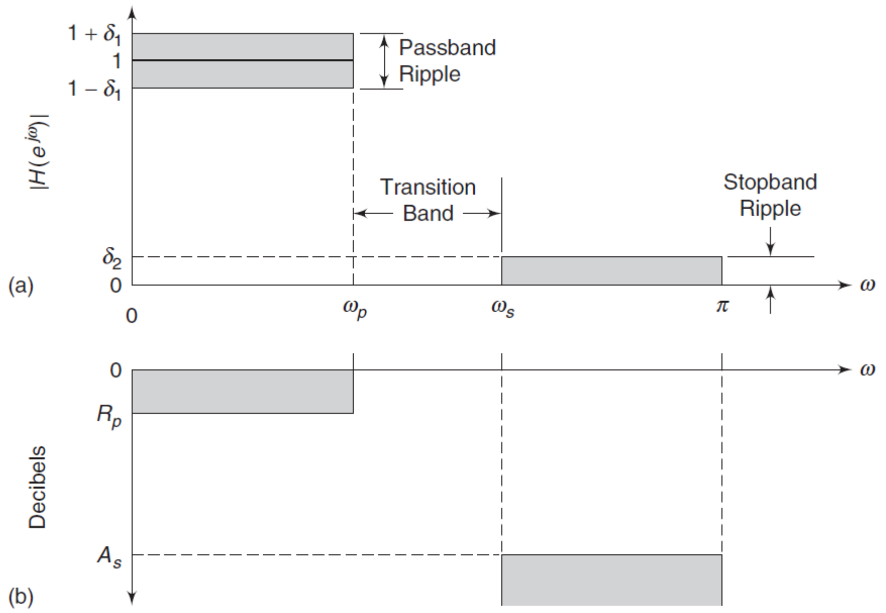


Figure 2: FIR filter specifications (Ingle and Proakis).

Most of the already-existing approaches use either a mathematical representation of a "near-ideal" impulse response which can arise with tradeoffs of its own (e.g. windowing method, such as Hamming, Bartlett, Blackman or Kaiser windows) or using an iterative algorithm that requires an initial guess for a number of parameters and may or may not converge after a number of iterations (e.g. Parks-McClellan or PM algorithm) [3]. Both of these approaches require a great deal of trial-and-error, since optimizing one quantity of the filter response often comes at the cost of another one. To that end, we shall attempt to use a metaheuristic optimization method that to tackle the compromise between the many parameters and attributes of such quantities.

2. Related Work

The attempt to design FIR filters using metaheuristic optimization algorithms have not been overlooked in scientific literature, varying between genetic algorithms (GA) and particle swarm optimizations (PSO), most recently by Rajasekhar [4, 2024], where a number of evolutionary algorithms has been used to optimize a number of parameters, such as cuckoo search algorithm (CSA), grasshopper optimization algorithms (GOA) and benchmarking them against the aforementioned PM algorithm. However, the use of genetic algorithm has been implemented on the Type-3 and Type-4 linear phase differentiator filters in 2022 by Asmae et al. [5, 2022] and the results were also benchmarked against the PM algorithm and compared with the Artificial Bee Colony swarm-based optimization. Another notable attempt was by Boz and Garip [6, 2018] providing a user-based approach to enter the specifications of the filter, while the GA handles the search for the desired filter coefficients. One more approach was via comparing the GA, PSO and BAT algorithms by Das et al. [7, 2016]. As we observe, the literature on such a topic are spaced-out temporally, yet the research for it still remains an untapped potential area of research that may be sped up drastically in the future.

3. Problem Formulation

The optimization of a digital filter response has been a pressing problem in the field of digital signal processing, since the shape of its frequency response is characterized by a number of parameters highlighted in Figure 1. Desired ideal filter characteristics require instantaneous transitions in the frequency response, which are physically unrealizable due to the finite nature of physical systems.

Search Space

The impulse response of the FIR filter can be characterized by the system linear constant-coefficient difference equation (LCCDE) as follows

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + \cdots + b_{N-1} x(n-M+1)$$

The impulse response is then given by the discrete impulses

$$h(n) = b_0 \delta(n) + b_1 \delta(n-1) + b_2 \delta(n-2) + \cdots + b_{N-1} \delta(n-M+1)$$
$$= \sum_{k=0}^{M-1} b_k \delta(n-k)$$

where the constant M represents the number of taps in the filter and $M-1$ represents the duration of the signal itself, and the coefficients $b_0, b_1, \dots, b_{M-1}$ represent the coefficients of the signal to be estimated. The complex frequency response can be represented by taking the z-transform of the impulse response as follows

$$H(z) = \mathcal{Z}\{h(n)\} = \sum_{k=0}^{M-1} h(k) z^{-k} = \sum_{k=0}^{M-1} b_k z^{-k}$$

The frequency response is then given by

$$H\left(e^{j\omega}\right) = \sum_{k=0}^{M-1} h(k) e^{-j\omega k} = \sum_{k=0}^{M-1} b_k e^{-j\omega k}$$

As we can observe, the coefficients of the series are what determine the shape of the frequency response of the filter. Since M can either be even or odd, this creates a variation of the filter response characteristic and splits the distinction into several types. We list them as follows.

Type-1 linear-phase FIR filter: Symmetrical impulse response, M odd.

$$H\left(e^{j\omega}\right) = \left[\sum_{n=0}^{(M-1)/2} a_k \cos(\omega n)\right] e^{-j\omega(M-1)/2}$$

Where a_k is obtained from h_k

$$a_0 = h\left(\frac{M-1}{2}\right)$$

$$a(n) = 2h\left(\frac{M-1}{2} - n\right), \quad 1 \leq n \leq \frac{M-3}{2}$$

Type-2 linear-phase FIR filter: Symmetrical impulse response, M even.

$$H\left(e^{j\omega}\right) = \left[\sum_{n=1}^{M/2} b_k \cos\left\{\omega\left(n - \frac{1}{2}\right)\right\}\right] e^{-j\omega(M-1)/2}$$

Where

$$b(n) = 2h\left(\frac{M}{2} - n\right), \quad n = 1,2,\dots,\frac{M}{2}$$

Type-3 linear-phase FIR filter: Asymmetrical impulse response, M odd.

$$H(e^{j\omega}) = \left[\sum_{n=1}^{(M-1)/2} c_k \sin(\omega n)\right] e^{j\left[\frac{\pi}{2}-\left(\frac{M-1}{2}\right)\omega\right]}$$

Where

$$c(n) = 2h\left(\frac{M-1}{2} - n\right), \quad n = 1,2,\dots,\frac{M-1}{2}$$

Type-4 linear-phase FIR filter: Asymmetrical impulse response, M even.

$$H(e^{j\omega}) = \left[\sum_{n=1}^{M/2} d_n \sin\left\{\omega\left(n - \frac{1}{2}\right)\right\}\right] e^{j\left[\frac{\pi}{2}-\left(\frac{M-1}{2}\right)\omega\right]}$$

Where

$$d(n) = 2h\left(\frac{M-1}{2} - n\right), \quad n = 1,2,\dots,\frac{M}{2}$$

The search space in each of the four cases is the coefficient series $a(n), b(n), c(n)$ and $d(n)$. Noting that the rewriting the series more compactly halves the number of terms at most. The search algorithm attempts to take a dab at the terms of each series and find the best possible combination of each of the solutions based on a given order M. Of course, increasing the filter order would mean increasing the accuracy of the filter, but also puts more strain on the search algorithm for the coefficients. To simplify our search, we shall only consider symmetric, even Type-1 filters.

Encoding

The method for encoding the chromosomes (possible solutions) for this problem includes a *vector* of coefficients that whose values are varied between certain ranges to allow a speedy solution, and these vectors are represented by the terms or coefficients of the series $a(n), b(n), c(n)$ and $d(n)$ for each type of the FIR filters. Each generation of the filter coefficients would then be mutated and crossed-over by a certain configuration (to be determined) to generate a possibility of solutions, until the algorithm has reached a good-enough fitness value that represents the best solution reached thus far. Hence, we shall search for a chromosome of the shape

$$\mathbf{h} = [h_0 \ \ h_1 \ \ h_2 \ \ \dots \ \ h_{M-1}]$$

Desired Filter Coefficients

To compare with a desired filter response, that is the ideal lowpass filter, we must first seek to find those ideal coefficients.

For a desired frequency response, $H_d(e^{j\omega})$, which is defined by a cutoff frequency ω_c as

$$H_d(e^{j\omega}) = \begin{cases} 1, & |\omega| < \omega_c \\ 0, & \omega_c < |\omega| < \pi \end{cases}$$

We shall find the filter coefficients $h_d(n)$ by using the inverse discrete-time Fourier transform (IDTFT) given as

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\omega})e^{j\omega n}d\omega = \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} e^{j\omega n}d\omega = \frac{\sin(\omega_c n)}{\pi n}$$
$$= \frac{\omega_c}{\pi}\operatorname{sinc}\left(\frac{\omega_c}{\pi}n\right)$$

The problem with this response is that it is not physically realizable, where no naturally occurring phenomenon can be mimicked to generate such response on a circuit. Furthermore, this response requires that the variable n be defined over all of natural numbers, which violates the causality of physical systems.

To that end, such ideal filters are made realistic by two steps: shifting and windowing. Shifting involves delaying the filter characteristic by a number of samples to be centered at that delay point, and windowing involves limiting this filter to be time-limited, hence the name *finite* impulse response. The shifting is done mathematically by centering h_d to peak at the center of coefficients, given a specific time window of shape $w(n)$, as follows.

$$h_r(n) = h_d\left(n - \frac{M}{2}\right)w(n)$$

In the frequency domain, this operation corresponds to a periodic convolution, as shown in Figure 3 below.

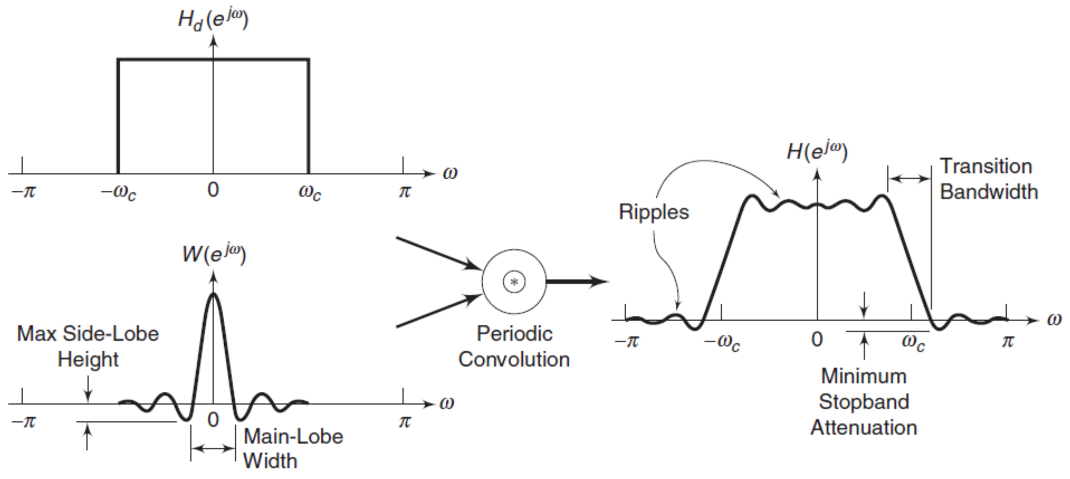


Figure 3: Actual filter response generated by the windowing method (Ingle and Proakis, 4th ed, p.311).

The filter response becomes bounded by the following criteria:

$$\left|H\left(e^{j\omega}\right)\right|_{\max} \leq 1 + \delta_1$$

$$0 \leq \left|H\left(e^{j\omega}\right)\right|_{\min} \leq \delta_2$$

Where the values δ_1 and δ_2 are related to the passband and stopband ripples, respectively, highlighted in Figure 1. In decibels (dB), the passband ripple is defined as follows

$$R_p = -20\log_{10}\frac{1-\delta_1}{1+\delta_1}$$

While the stopband attenuation is defined as

$$A_s = -20\log_{10}\frac{\delta_2}{1+\delta_1}$$

These two values are often given in the design specifications, as well as the filter order $M - 1$.

Fitness Function

The objective function that describes the filter performance can be chosen from a variety of parameters. For the sake of simplicity, we shall consider the magnitude response to be the defining function in the function, where the real desired filter coefficients $h_r(n)$ is compared with the actual response $h(n)$ using the mean-squared error (MSE) criteria. Since the numbers are typically between 0 and 1 and are sensitive to decimal variation, we shall scale the MSE by a factor of 100, as follows

$$F = 100 \sum_{n=0}^{M-1} |h(n) - h_r(n)|^2$$

This function is to be minimized.

4. Methodology

Implementing this project shall be through the use of some comparison with common FIR filter design approaches, which are listed below.

- **Window Method:** This method involves designing an ideal filter and then applying a window function to truncate and smooth the impulse response. Common window functions include Rectangular, Hamming, Hann, Blackman and Kaiser windows.
- **Frequency Sampling Method:** This method samples the desired frequency response at a finite number of points and then uses the Inverse Discrete Fourier Transform (IDFT) to find the filter coefficients.
- **Optimization Techniques:** Algorithms such as Parks-McClellan use optimization to find the filter coefficients that best meet a desired specification, typically involving trade-offs between passband and stopband performance.

For our purpose, we shall use both the windowing method and the Parks-McClellan algorithm to compare the implementation.

Programming Language

Due to the number of built-in functions used for digital signal processing applications, the choice has been made to select MATLAB for the simulation of this project, as well as for the handiness of its global optimization toolbox, and the ease of varying the parameters of the genetic algorithm to explore more angles for approaching this problem. Furthermore, the demand of using some special mathematical functions which are not easily implemented using other languages is mitigated in MATLAB due to their inclusion it its standard libraries.

Evaluation Approach

The FIR filter design is one of the most common and “ad hoc” problems that does not have a universal optimum solution for all approaches. As previously stated, there exist several algorithms for the implementation of a certain filter specification that may or may not work all as expected, therefore exploration of different approaches is a must.

The implementation of the windowing method is made in the simplest approach, that is using a rectangular window of size $M - 1$, as such.

$$w(n) = \begin{cases} 1, & 0 \leq n \leq M - 1 \\ 0, & \text{otherwise} \end{cases}$$

Furthermore, implementing the Parks-McClellan algorithm is quite common, which is an iterative algorithm for finding the optimal Chebyshev finite impulse response filter. The use of Chebyshev polynomials is convenient in this problem since the mathematical form fits perfectly into the description of the four cases of FIR filters (with slight tweaks to some). This algorithm works on minimizing the approximation error using the Remez exchange technique. [8, 1934].

Upon finding the optimal coefficients using GA, we would then compare the performance of this trial with the PM results and check whether the solutions meet the desired optimization constraints and which algorithm achieves superiority.

5. Results and Discussion

The criteria we shall attempt to design throughout this project are given below:

Passband frequency: $\omega_p = 0.4\pi$ rad/sample

Stopband frequency: $\omega_s = 0.6\pi$ rad/sample

Cutoff frequency: $\omega_c = \frac{\omega_p + \omega_s}{2} = 0.5\pi \text{ rad/sample}$

Transition bandwidth: $\Delta\omega = \omega_s - \omega_p = 0.2\pi$ rad/sample

Windowing Method

To demonstrate the use of the windowing method, we have shown previously that in order to analyze the frequency response accurately, we require to perform periodic convolution, which is in most cases an intractable integral. Therefore, the windowing method, with the use of some deliberate shaping functions, can approximate the filter characteristic up to a certain *fixed* stopband attenuation criterion, with given formulas related to filter order.

For simplicity we shall use the simplified *rectangular* or *boxcar* window, which is an impulse train of length M with an amplitude of 1. The use of the rectangular window provides a maximum stopband attenuation of -21 dB, where the ripples in the passband continuously fall below that threshold. The filter order is now given by

$$M = \frac{4\pi}{\Delta\omega} = 20$$

The frequency response for the windowing method compared to the desired frequency response are given in the figure below.

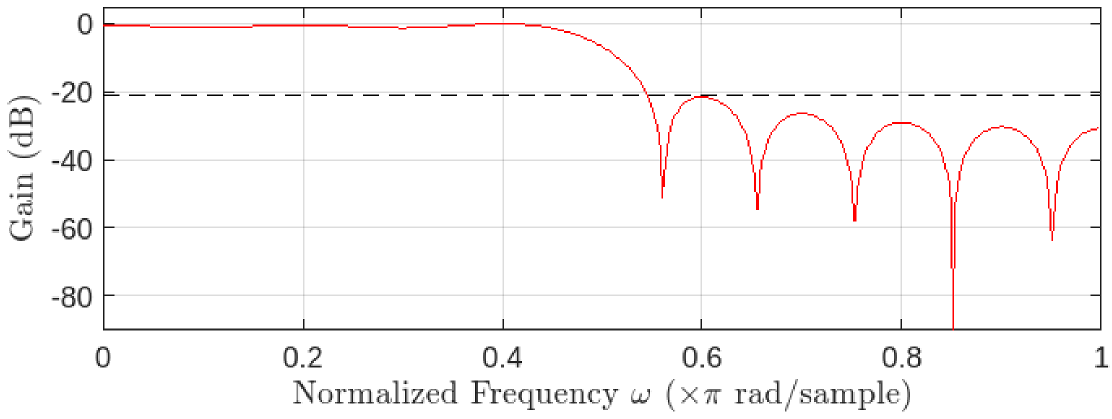


Figure 4: Windowing method results.

As we can see, the use of red color was to indicate the comparison between desired filter and practical filter, and they stack up on each other since the window is a rectangular window. The dashed black line is the limit of -21 dB given by the rectangular window specifications.

Parks-McClellan Algorithm

The Parks-McClellan algorithm has been described previously in the proposal report, where it returns an approximation of the impulse response given some ripple criteria for the passband and the stopband, as well as the passband and stopband frequencies, whose average returns the cutoff frequency of the filter.

Initial testing was performed on a *lowpass filter* design. The filter requirements were as follows:

- Passband frequency: $\omega_p = 0.4\pi$ rad/sample
- Stopband frequency: $\omega_s = 0.6\pi$ rad/sample
- $\delta_p = 0.05$ corresponding to a passband ripple of $R_p = 0.8693$ dB
- $\delta_s = 0.01$ corresponding to a stopband attenuation of $A_s = 40.4238$ dB

The algorithm has an approximate (empirical) formula for estimating the filter order, which is given as

$$M = \left\lceil \frac{-10\log_{10}(\delta_p \delta_s) - 13}{2.324(\omega_s - \omega_p)} \right\rceil$$

Where the $\lceil x \rceil$ represents the ceiling of x.

Upon selecting the value for M, we can pick the type of the FIR filter (1 to 4) and we shall modify it should the value not meet our design criteria.

For our purposes, the values above, the value of $M = 15$.

The criteria for optimizing the Parks-McClellan algorithm uses the minimax optimality criteria, which minimizes the worst-case (maximum) error of the difference between the actual response $H(\omega)$ and desired response $H_d(\omega)$, shaped by a certain weighting function $W(\omega)$, as follows:

$$\min_{h[n]} \max_{\omega\in\Omega} E(\omega) = \min_{h[n]} \max_{\omega\in\Omega} \left| W(\omega)\big(H_d(\omega) - H(\omega)\big) \right|$$

The returned filter coefficients are expressed as follows:

h = [-0.0079, 0.0247, 0.0507, -0.0099, -0.0920, 0.0142, 0.3135, 0.4851, 0.3135, 0.0142, -0.0920, -0.0099, 0.0507, 0.0247, -0.0079]

The plots of the impulse response in the time domain, the actual filter frequency response, and the error of between the desired and actual responses are shown below

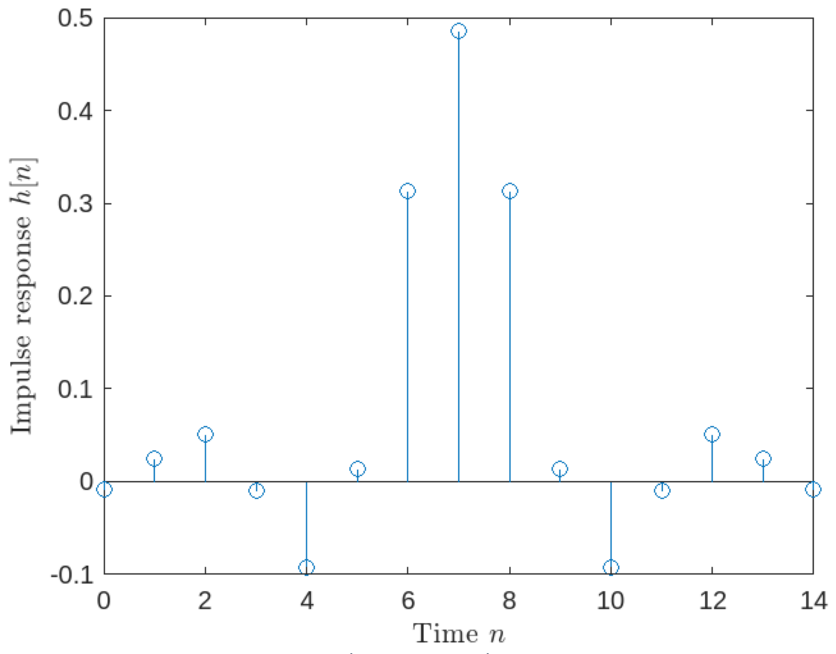


Figure 5: Impulse response plot versus time.

As we observe, the filter coefficients are centered around $n = 7$, which constitutes a Type-2 FIR filter (symmetrical, M even).

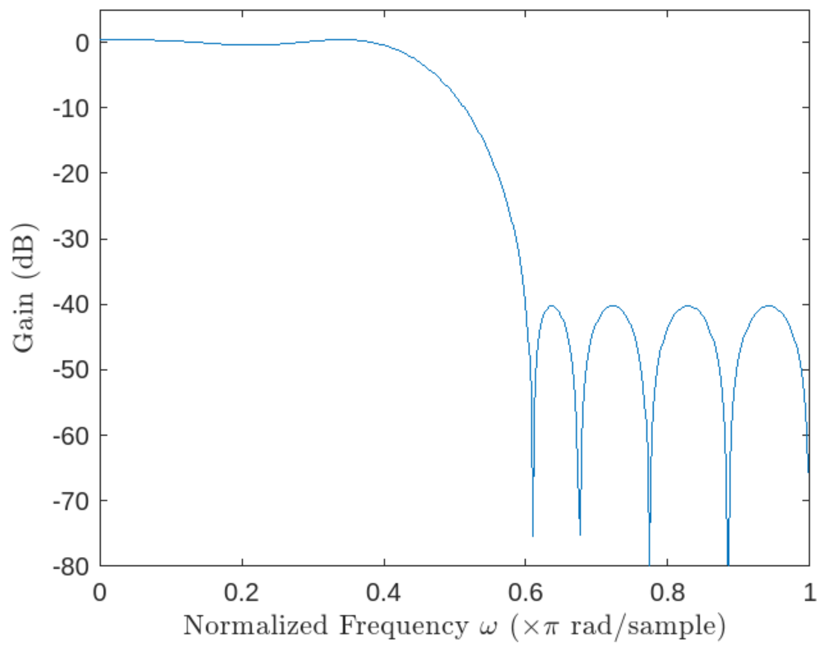


Figure 6: Frequency response (discrete-time Fourier transform) of the filter impulse response.

As we can see in Figure 5, there is a steep drop in the response at $\omega = 0.6\pi$ rad/sample, however, this alone is not a sound measure to estimate filter performance, since we are applying the algorithm for the passband and stopband errors (overshoot) criteria.

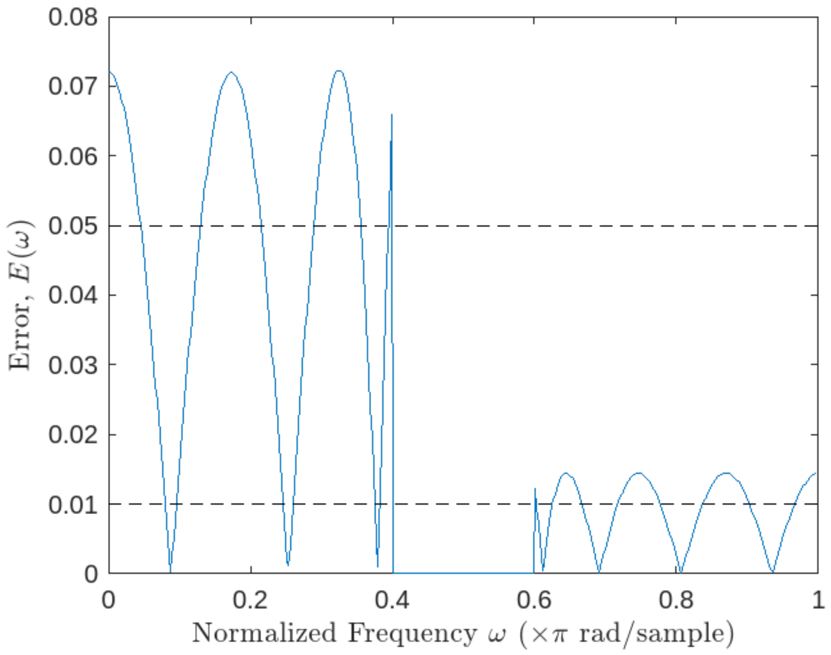


Figure 7: Error difference between actual and desired responses.

In Figure 6, we can observe that the dashed lines represent the passband and stopband errors of 0.05 and 0.01, respectively, and we see that the filter coefficients returned by the algorithm do not fulfill these criteria, so a design change is required.

If we make the filter order $M = 15$ (just one more tap above), the filter turns into Type-1 (symmetrical, M odd), and the response becomes as follows:

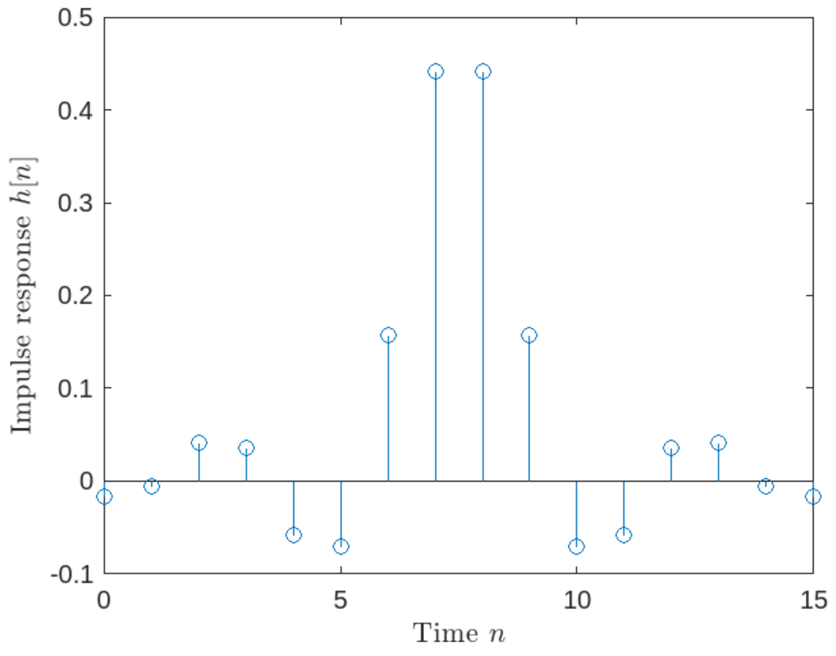


Figure 8: Impulse response for M=15.

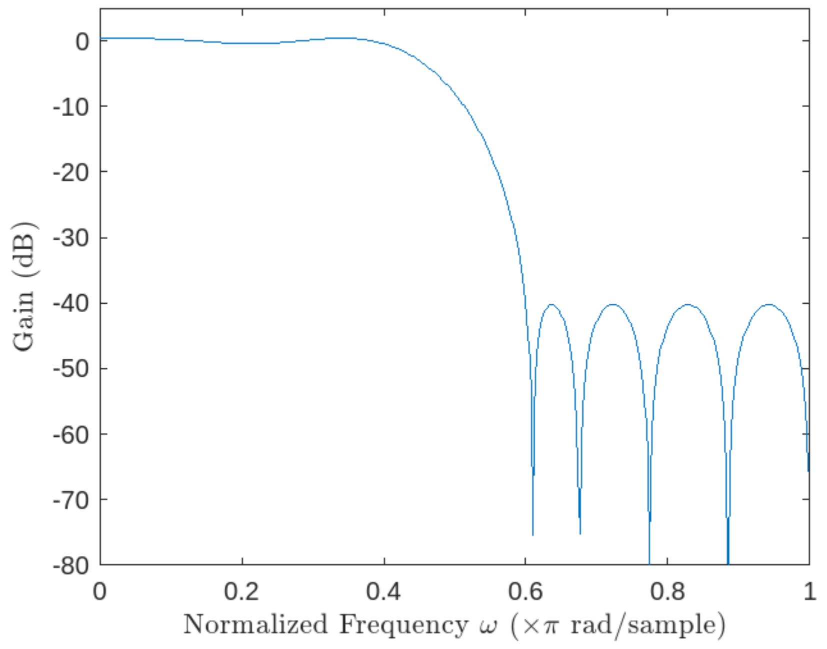


Figure 9: Frequency response for M=15.

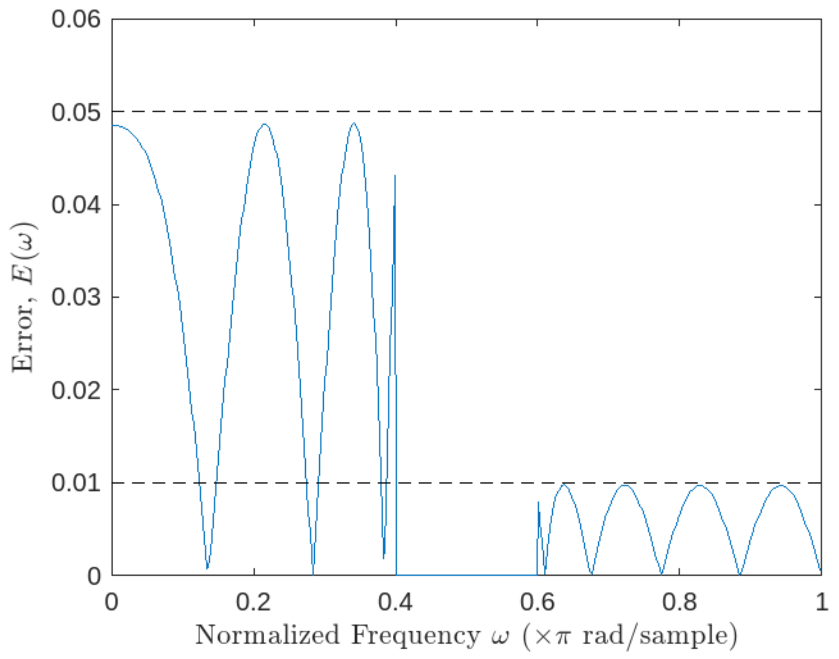


Figure 10: Error response for the filter.

As evident, the frequency response of the filter now produces an error that fits with the criteria.

Genetic Algorithm

The use of the genetic algorithm in finding the filter coefficients is used in hopes to see how metaheuristics can participate in solving the FIR filter design problem. To minimized the fitness function, we have set the same design specifications to the code, and we ran the code using multiple settings, but after a painstaking phase of trial-and-error, we have deduced that the ideal parameters for achieving the best fitness are as follows:

- **Population Size:** 100
- **Number of Generations:** 600
- **Selection Function:** Tournament
- **Crossover Function:** Arithmetic
- **Mutation Function:** Adaptive feasible
- **Crossover Fraction:** 0.4

After running this setting for 20 times, each with a different seed (from 1-20 for simplicity), we have deduced that the fitness function was near-optimum with the following values:

Fitness Runs = [6.7456E-05, 1.9361E-05, 2.0436E-04, 7.7911E-05, 4.2291E-05, 2.3240E-04, 6.1989E-05, 5.1052E-05, 9.4829E-05, 1.1564E-04, 2.7039E-04, 2.5912E-05, 1.0442E-04, 9.1203E-05, 5.9898E-05, 6.5600E-05, 1.8370E-04, 9.6922E-05, 3.9534E-05, 3.8745E-05**]**.

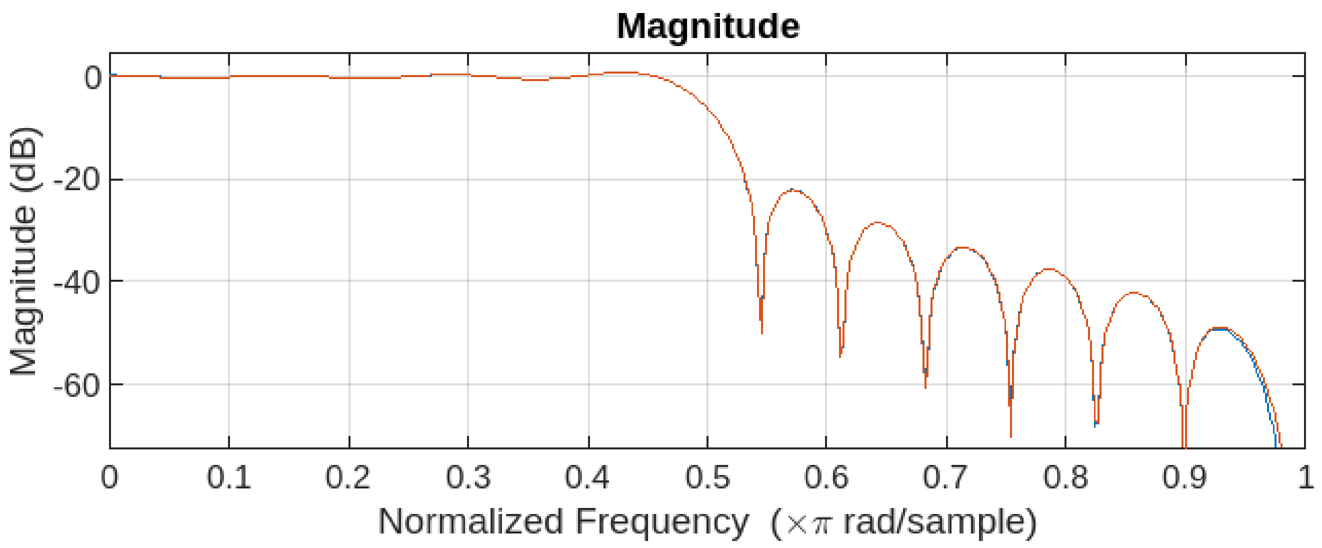
The average fitness value is **9.7181E-05.**

The standard deviation is **7.0929E-05.**

The median fitness value is **7.2684E-05.**

As for the filter coefficients, the average of the 20 runs for coefficients has been taken as our solution, and then the discrete-time Fourier transform has been taken.

It can be observed how the average solution in red compares to the ideal filter response in blue, where the discrepancy can only be seen ever so slightly at the end of the stopband region, where the blue color can be shown.



6. Conclusion

The search for a practical and powerful filter response has always been in the research literature of digital signal processing. As demonstrated, the use of the genetic algorithm can very well maintain the solution of this search quite efficiently, stacking up to famous algorithms such as the windowing method and the Parks-McClellan algorithm, with statistically significant results.

Appendix

A.1 Code for Windowing Method

```
close all
clear
clc
%%
wp = 0.4*pi;        % Passband frequency
ws = 0.6*pi;        % Stopband frequency
wc = 0.5*pi;        % Cutoff frequency
delta_w = ws-wp;    % Transition width
M = ceil(4*pi/delta_w); % Filter coefficients formula (rectangular window)
% Prepare practical filter
n = 0:M-1;
w = boxcar(M)';
hd = (wc/pi)*sinc((wc/pi)*(n-M/2));
h = hd.*w;
% Frequency response
[H,f] = freqz(h,1,512);
H_dB = 20*log10(abs(H)/max(abs(H)));
subplot(2,1,1)
plot(f/pi,H_dB)
xlabel('Normalized Frequency $\omega$ ($\times \pi$
rad/sample)','Interpreter','latex')
ylabel('Gain (dB)','Interpreter','latex')
hold on
plot(f/pi,-21*ones(size(f)),'--k')
ylim([min(H_dB) 5])
subplot(2,1,2)
plot(f/pi,angle(H))
xlabel('Normalized Frequency $\omega$ ($\times \pi$
rad/sample)','Interpreter','latex')
ylabel('Phase (radians)','Interpreter','latex')
```

A.2 Code for Parks-McClellan Algorithm

```
close all
clear
clc
%%
delta_p = 0.05;     % Passband error
delta_s = 0.01;     % Stopband error
wp = 0.4*pi;        % Passband frequency
ws = 0.6*pi;        % Stopband frequency
M = ceil((-10*log10(delta_p*delta_s)-13)/(2.324*(ws-wp)));
% Filter order
w = [0, wp, ws, pi]; % Critical requencies
w_by_pi = w/pi;     % Normalized axis
A = [1 1 0 0];      % Critical amplitude factors
W = [1 delta_p/delta_s];    % Weighting function
```

```
h = firpm(M,w_by_pi,A,W);   % Parks-McClellan algorithm filter coefficients
n = 0:M;             % Discrete time vector
stem(n,h)            % Plot h[n]
xlabel('Time $n$','Interpreter','latex')
ylabel('Impulse response $h[n]$','Interpreter','latex')
[H,omega] = freqz(h,1,512);

% Indices for critical frequencies in the omega vector
i1 = find(abs(omega - wp) <= 0.005);
i2 = find(abs(omega - ws) <= 0.005);
i3 = find(abs(omega-(wp+ws)/2)<=0.005);

% Expanded weighting function (size match)
W2 = [ones(i1(1),1); zeros(i2(1)-i1(1),1); ones(512-i2(1),1)];
Hd = [ones(i1(1),1); zeros(512-i1(1),1)];

% Error (objective) function to be minimized (for the algorithm)
E = abs(W2.*(abs(H)-abs(Hd)));
figure
plot(omega/pi,20*log10(abs(H)))
xlabel('Normalized Frequency $\omega \times \pi$ (rad/sample)','Interpreter','latex')
ylabel('Gain (dB)','Interpreter','latex')
ylim([-80 5])
figure
plot(omega/pi,E,omega/pi,delta_p*ones(size(omega)),'k--
',omega/pi,delta_s*ones(size(omega)),'k--')
xlabel('Normalized Frequency $\omega \times \pi$ (rad/sample)','Interpreter','latex')
ylabel('Gain (dB)','Interpreter','latex')
```

A.3 Code for Genetic Algorithm

```
close all
clear
clc

% Define filter specifications
Fp = 0.4;  % Passband frequency (normalized, 0 to 1)
Fs = 0.6;  % Stopband frequency (normalized, 0 to 1)
Fc = (Fp+Fs)/2; % Cutoff frequency (normalized, 0 to 1)
Rp = 0.1;  % Passband ripple (dB)
As = 60;   % Stopband attenuation (dB)
wp = Fp*pi;
ws = Fs*pi;
wc = Fc*pi;
delta_p = (10^(Rp/20)-1)/(10^(Rp/20)+1);
delta_s = (1+delta_p)*10^(-As/20);
delta_w = ws-wp;
M = ceil((-10*log10(delta_p*delta_s)-13)/(2.285*delta_w));

% Adjust the order to ensure it's even
if mod(M, 2) ~= 0
    M = M + 1;
end
```

```matlab
% Calculate ideal sinc coefficients for a low-pass filter
alpha = (M - 1) / 2;
n = 0:M-1;
h = Fc*sinc(Fc * (n - alpha));
ideal_coefficients = h;

%% Run this for 20 different seeds and record values for statistical inference
record = {};
for k = 1:20
    rng(k);
    % Fitness function handle
    fitnessFcn = @(coefficients) fir_fitness_function(coefficients,
ideal_coefficients, Fp, Fs, Rp, As, M);

    % Genetic Algorithm options
    options = optimoptions('ga', ...
        'PopulationSize', 100, ...
        'MaxGenerations', 600, ...
        'SelectionFcn', @selectiontournament, ... % Tournament selection
        'CrossoverFcn', @crossoverarithmetic, ... % Arithmetic crossover
        'MutationFcn', @mutationadaptfeasible, ... % Adaptive feasible mutation
        'CrossoverFraction', 0.4, ... % Crossover fraction
        'Display', 'iter');

    % Run Genetic Algorithm
    [optimal_half_coefficients, optimal_fitness] = ga(fitnessFcn, (M/2)+1, [], [],
[], [], [], [], [], options);

    % Construct the full set of coefficients for Type-1 FIR filter
    optimal_coefficients = [optimal_half_coefficients,
fliplr(optimal_half_coefficients(1:end-2))];


    % Display results
    disp('Optimal Coefficients:');
    disp(optimal_coefficients);
    disp('Optimal Fitness:');
    disp(optimal_fitness);
    record{1,k} = optimal_coefficients;
    record{2,k} = optimal_fitness;
end

% Calculate the frequency response of the windowed filter
[H_optimal, f] = freqz(optimal_coefficients, 1, 1024, 'whole');
H_optimal_dB = 20*log10(abs(H_optimal)/max(abs(H_optimal)));
f = f / pi; % Normalize frequency

% Plot the frequency response
figure;
plot(f, H_optimal_dB);
title('Frequency Response of the Optimized Lowpass FIR Filter');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Magnitude (dB)');
grid on;
hold on;
```

```
% Plot passband and stopband constraints
[H_ideal,f1] = freqz(ideal_coefficients,1,1024,"whole");
H_ideal_dB = 20*log10(abs(H_ideal)/max(abs(H_ideal)));
f1 = f1/pi;
plot(f1,H_ideal_dB)
legend('Frequency Response','Ideal Response');
%ylim([min(min(abs(H_ideal-dB)),min(abs(H_optimal_dB))) 1])
hold off;
err = h-optimal_coefficients;
figure
stem(err)

coeffs = [];
for k = 1:20
    coeffs = [coeffs;record{1,k}];
end

avgCoeff = [];
for k = 1:28
    avgCoeff(k) = mean(coeffs(:,k));
end

fitnessz = [];
for k = 1:20
    fitnessz = [fitnessz;record{2,k}];
end

function fitness = fir_fitness_function(coefficients, ideal_coefficients, Fp, Fs, Rp,
As, N)
    % Construct the full set of coefficients for Type-1 FIR filter
    % Adjust the order to ensure it's even
    if mod(N, 2) ~= 0
        N = N + 1;
    end

    b = [coefficients, fliplr(coefficients(1:end-2))];

    % Frequency response
    [H, f] = freqz(b, 1, 1024, 'whole'); % Full frequency response
    H_dB = 20*log10(abs(H));

    % Normalize the frequency vector
    f = f/pi;

    % Define frequency bands
    passband = f <= Fp;
    stopband = f >= Fs;

    % Calculate passband ripple and stopband attenuation
    passband_ripple = max(H_dB(passband)) - min(H_dB(passband));
    stopband_min_attenuation = min(H_dB(stopband));

    % Error between GA-generated coefficients and ideal sinc coefficients
    coeff_error = sum((b - ideal_coefficients).^2);
    % Fitness function: a combination of ripple and attenuation with penalties
    % fitness = passband_ripple + abs(stopband_min_attenuation + As) +
passband_penalty + stopband_penalty ;
    fitness = 100*coeff_error;
end
```

References

[1] https://upload.wikimedia.org/wikipedia/commons/9/9b/FIR_Filter.svg

[2] Ingle, V. and Proakis, J.: Digital Signal Processing Using MATLAB: A Problem Solving Companion, 4th ed, Cengage Learning, p370.

[3] McClellan, J.H.; Parks, T.W. (2005). "A personal history of the Parks-Mc Clellan algorithm". IEEE Signal Processing Magazine. 22 (2): 82–86.

[4] Rajasekhar, K. L1-Norm and LMS Based Digital FIR Filters Design Using Evolutionary Algorithms. J. Electr. Eng. Technol. 19, 753–762 (2024). https://doi.org/10.1007/s42835-023-01589-7

[5] E. Beqal Asmae, B. Bachir and Z. Izeddine, "Optimal Design of Type 3 and Type 4 Linear Phase FIR Differentiators using the Genetic Algorithm," 2022 IEEE 3rd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS), Fez, Morocco, 2022, pp. 1-6, doi: 10.1109/ICECOCS55148.2022.9982994

[6] B. Garıp, Z., & Boz, A. F. (2018). The FIR Filter Design based on Genetic Algorithm. Balkan Journal of Electrical and Computer Engineering, 6, 33-36. https://doi.org/10.17694/bajece.410234

[7] P. Das, S. K. Naskar, S. Samanta and S. N. Patra, "An approach to optimize FIR filter coefficients using GA, PSO and BAT algorithm and their comparative analysis," 2016 International Conference on Computer, Electrical & Communication Engineering (ICCECE), Kolkata, India, 2016, pp. 1-6, doi: 10.1109/ICCECE.2016.8009579

[8] E. Ya. Remez, "Sur la détermination des polynômes d'approximation de degré donnée", Comm. Soc. Math. Kharkov 10, 41 (1934);

"Sur un procédé convergent d'approximations successives pour déterminer les polynômes d'approximation, Compt. Rend. Acad. Sc. 198, 2063 (1934);

"Sur le calcul effectiv des polynômes d'approximation des Tschebyscheff", Compt. Rend. Acade. Sc. 199, 337 (1934).