

Digital Communication Filter Design by Stochastic Optimization

Rod KENDALL, Robin BRAUN

Abstract—The subject of this work is to analyze the behavior of high-speed digital QPSK modulated communication channels in the presence of the Additive White Gaussian Noise (AWGN) and Inter Symbol Interference (ISI). The high frequency of the carrier in this application makes digital filtering rather difficult, thus an analogue filter was suggested for the receiver. The paper deals with the problem matching the filter to the signal for best performance.

Bit Error Rate (BER) is the single most important factor determining the performance of a digital communication channel. The receiver filter was thus designed to minimize the BER of the channel, taking into account the pulse shape as well as both the channel noise and the ISI. The filter was designed using a derivative-free optimization technique, known as the Population Based Incremental Learning (PBIL). A sample solution of the search process is presented at the end and the results are analyzed.

Index Terms— Derivative-free Optimization, Inter-symbol Interference (ISI), Population Based Incremental Learning (PBIL), QPSK.

I. INTRODUCTION

MODERN communications demand ever higher signaling rates with performance as close as possible to the theoretically optimal. A successful design must meet several conflicting requirements: low bandwidth, low ISI, good noise performance, etc. The high data rates employed in modern communications often make digital signal processing difficult, expensive or right impossible. For this reason, an analogue receiver filter was proposed here to optimize the performance of the communication channel.

The question remains how to optimize the RX filter for a particular signal waveform. If we assume we can generate waveforms for no ISI at the receiver, the optimal filter will be matched to the signal. However, in the absence of ideal signal waveforms and filters we have to consider ISI. Now we are facing a conflicting requirement: a filter optimized for noise is not necessarily best for ISI reduction and vice versa. The only

true performance indicator has to be the Bit Error Rate (BER). With the complex relationship between the symbol waveform at the receiver, the RX filter shape and the BER, we have chosen to model the channel in the computer and to simulate the BER performance while varying the RX filter makeup. With a single payoff function (BER) it is natural to employ a random search algorithm in order to find the optimal solution.

II. DIGITAL COMMUNICATION CHANNEL

A. Model of the QPSK Channel

Let us assume a QPSK channel, where the signaling pulse is of arbitrary shape, and the carrier frequency is significantly higher than the data rate. Such a channel can be modeled as in Figure 1.

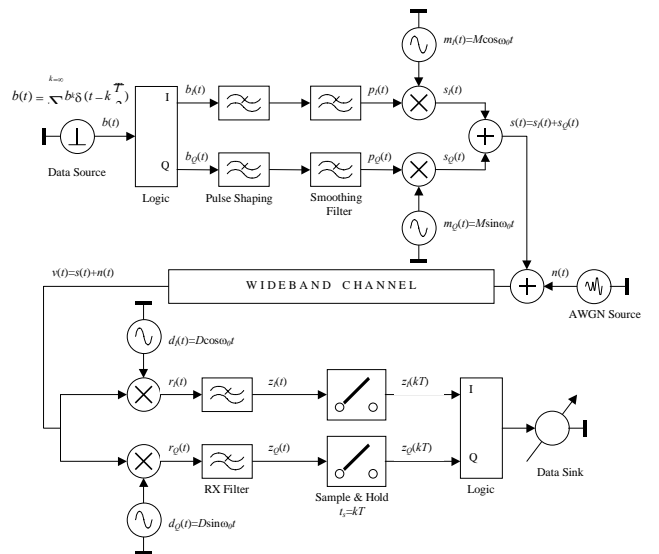


Figure 1: QPSK channel

The data source is modeled as a string of Dirac δ pulses of either polarity. T was chosen as the signaling period of each (I and Q) stream, thus $T/2$ is the period of the combined data stream. A logic circuit is used to split the data $b(t)$ into two separate streams $b_I(t)$ and $b_Q(t)$ (in-phase and quad component). The period of these signals is now T . The pulses are then passed through a pulse-shaping filter to give them the desired waveform.

Manuscript received November 12, 2001.

Rod Kendall is with the CSIRO division of Telecommunications and Industrial Physics, Marsfield, NSW, Australia (rod.kendall@tip.csiro.au).

Robin Braun is with the University of Technology, Sydney, working in collaboration with the CSIRO division of Telecommunications and Industrial Physics, Marsfield, NSW, Australia (Robin.Braun@eng.uts.edu.au).

The QPSK signal is transmitted through a wideband communication channel. The term “wideband” in this context means that the bandwidth of the channel is much higher than any frequencies involved in the data transfer.

B. Equivalent Baseband Channel

For reasons of clarity and simplified processing it is customary to split the QPSK signal into two orthogonal components, effectively reducing QPSK channel to two independent BPSK channels (QPSK has the same BER performance as the BPSK). In addition, it is obvious that the RF has zero information content, so the channel can be translated to its baseband version.

In line with the above conclusions, the following model can be used to investigate the BER of the digital channel:

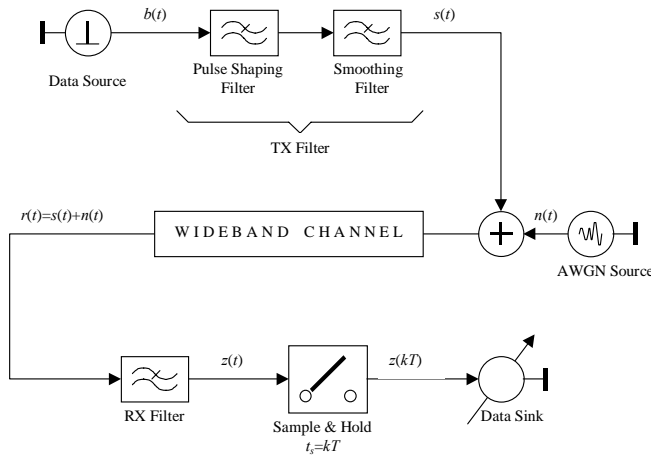


Figure 2: Baseband equivalent BPSK channel

The data source is a string of Dirac δ pulses of either polarity:

$$(1) \quad b(t) = \sum_{k=-\infty}^{k=\infty} b_k \delta(t - kT)$$

where b_k can be either $+B$ or $-B$ (antipodal signaling). Both polarities are assumed equally probable and there is no correlation between successive symbols (there is no way to predict the following symbol, based on the history of previous symbols). In reality this is seldom (if ever) the case, however, we will make this assumption in order to enable the ISI analysis.

The noise in the channel is AWGN:

$$(2) \quad p_n(n) = \frac{1}{\sigma_n \sqrt{2\pi}} \exp\left(-\frac{1}{2} \left(\frac{n}{\sigma_n}\right)^2\right)$$

$$(3) \quad G_n = \frac{N_0}{2}$$

$$(4) \quad P_n = \sigma_n^2 = \int_{-\infty}^{\infty} G_n df = \frac{N_0}{2} \int_{-\infty}^{\infty} df = \infty$$

The resulting waveform $z(t)$ is sampled at regular intervals

$t = kT$ to yield a discrete function of time, $z(kT)$.

III. CHANNEL ANALYSIS

A. RRC Signal

A typical baseband digital system uses a defined pulse shape to carry the information through the channel (Figure 3). These pulses are subjected to low-pass filtering in order to confine the spectrum to a desired bandwidth. This in turn leads to “spreading” of the pulses and Intersymbol interference (ISI). The RX filter is intended to compensate for this effect (equalization).

The transfer functions of the TX filter, the channel and the RX filter combine to

$$(5) \quad H(f) = H_{TX}(f) \cdot H_C(f) \cdot H_{RX}(f)$$

One way of avoiding ISI is to shape the pulses so that they pass through zeros for all previous and all future sampling moments. Nyquist has shown that for a data rate R_s symbols/s, the minimum system bandwidth allowing zero ISI detection is $R_s/2$ Hertz. This is the case when the system transfer function $H(f)$ is rectangular, and the pulse shape is $\text{sinc}(t/T)$, the function that passes through zeros for all sampling moments. However, such a transfer function is unattainable in the real world.

A more acceptable and frequently used system transfer function is the Raised Cosine (RC) function. It has the same problem as the “brickwall” version in that it is not physically realizable with real filters. However, it can be relatively closely approximated and the amplitude around the zero crossing points is lower, leading to more relaxed timing requirements. The penalty is in increased bandwidth.

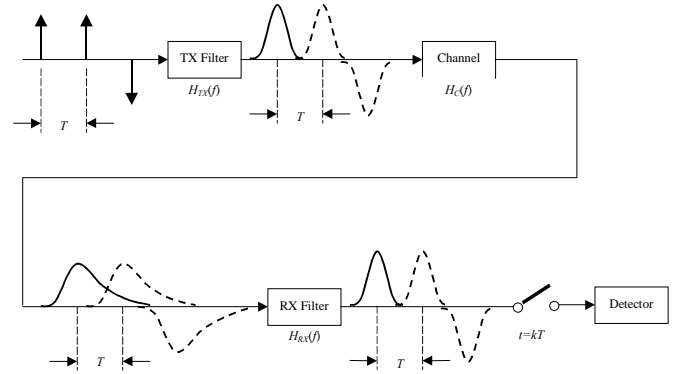


Figure 3: Intersymbol interference

The idea behind the Root Raised Cosine (RRC) response is to divide the transfer function equally between the TX filter defining the pulse shape and the RX filter. The two will combine to form the Raised Cosine pulse with zero ISI.

B. The Smoothing Filter

In the actual implementation of the circuit, the waveform is digitally generated and its spectrum extends far beyond the baseband region. A smoothing reconstruction filter is used to rid the signal of unwanted components. This is essentially a

gentle low-pass filter and has to be compensated for during the matching RX filter design.

C. Detection of BPSK in Gaussian Noise

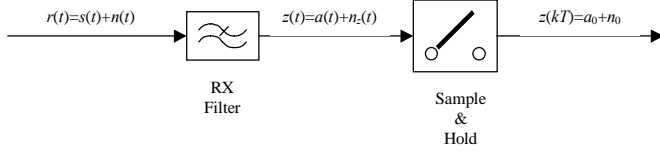


Figure 4: BPSK detector

The detector of the BPSK signal can in general be shown as in Figure 4. The received signal

$$(6) \quad r(t) = s(t) + n(t)$$

consists of the transmitted signal and the AWGN. It is filtered

$$(7) \quad \begin{aligned} a(t) &= s(t) * h_{RX}(t) \\ n_z(t) &= n(t) * h_{RX}(t) \end{aligned}$$

$$(8) \quad \begin{aligned} z(t) &= r(t) * h_{RX}(t) \\ z(t) &= (s(t) + n(t)) * h_{RX}(t) = a(t) + n_z(t) \end{aligned}$$

and then sampled at regular intervals kT

$$(9) \quad z(kT) = a_0 + n_0$$

The filtered noise is Gaussian with the variance (or power) equal to

$$(10) \quad P_z = \sigma_z^2 = \sigma_0^2 = \sigma_n^2 \int_{-\infty}^{\infty} |H_{RX}(f)|^2 df$$

The desired component of the filtered signal $a(t)$ at sampling moments is defined as

$$(11) \quad a(kT) = a(t)|_{t=kT} = s(t) * h_{TX}(t)|_{t=kT}$$

Depending on the symbol transmitted ($+B$ or $-B$), the received signal is $+a_0$ or $-a_0$ respectively. The PDF of each symbol can be described as

$$(12) \quad \begin{aligned} p(z|_{+B}) &= \frac{1}{\sigma_0 \sqrt{2\pi}} \exp\left(-\frac{1}{2} \cdot \left(\frac{z - a_0}{\sigma_0}\right)^2\right), \\ p(z|_{-B}) &= \frac{1}{\sigma_0 \sqrt{2\pi}} \exp\left(-\frac{1}{2} \cdot \left(\frac{z + a_0}{\sigma_0}\right)^2\right) \end{aligned}$$

where σ_0 is the standard deviation of the sampled noise. Both symbols are equally probable, so the maximum likelihood detector threshold is placed in the middle between $-a_0$ and $+a_0$:

$$(13) \quad \gamma_0 = \frac{a_0 + (-a_0)}{2} = 0$$

The situation is illustrated in Figure 5. The probability of bit error P_b is the probability that an incorrect decision has been made due to the presence of noise in the channel.

$$P_b = P(a_0)|_{-B} + P(-a_0)|_{+B}$$

$$P_b = \frac{1}{2} \left(\int_{-\infty}^{\gamma_0} p(z|_{+B}) + \int_{\gamma_0}^{\infty} p(z|_{-B}) \right) dz = \int_{\gamma_0}^{\infty} p(z|_{-B}) dz$$

If we replace the PDF $p(z|_{-B})$ with the expression from (12), we can write

$$(14) \quad P_b = \int_{\gamma_0}^{\infty} \frac{1}{\sigma_0 \sqrt{2\pi}} \exp\left(-\frac{1}{2} \left(\frac{z + a_0}{\sigma_0}\right)^2\right) dz$$

because both symbols are equally probable. This is identical to the shaded area in Figure 5.

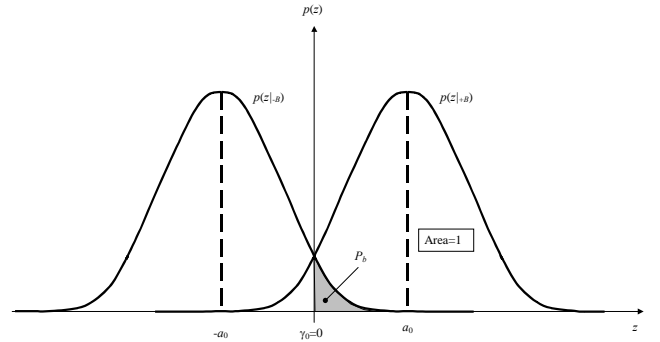


Figure 5: BPSK detection with Gaussian noise

Let $u = (z + a_0)/\sigma_0$. Then $dz = \sigma_0 du$, and

$$(15) \quad P_b = \int_{\frac{a_0}{\sigma_0}}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right) du = Q\left(\frac{a_0}{\sigma_0}\right)$$

The integral above is also known as the error function or

$$(16) \quad Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} \exp\left(-\frac{u^2}{2}\right) du$$

$Q(x)$ can not be analytically evaluated. However, it is available in numerous tables. The function is also built into various mathematical programs and software packages.

D. BER of BPSK Channel with ISI

Above BER applies to a match-filtered BPSK channel with AWGN and no ISI. However, the low pass character of real filters will inevitably lead to some ISI with the corresponding reduction of the BER performance.

The impact of filtering on the BER is less than straightforward. The same RX filter used to limit the noise power and work towards a lower BER will, because it is low pass in nature, cause smearing of the demodulated waveform, giving rise to ISI. The same filter therefore works on both the noise power and the ISI. We have to determine the relationship between the BER and the transfer characteristic of the RX filter.

Successive impulses will be overlapping (Figure 6). At the sampling moment the remnants of several previous and future pulses will be added to the desired amplitude a_0 . These

interfering amplitudes are marked as a_{-3} , a_{-2} , a_{-1} , a_1 , a_2 , and a_3 in Figure 7.

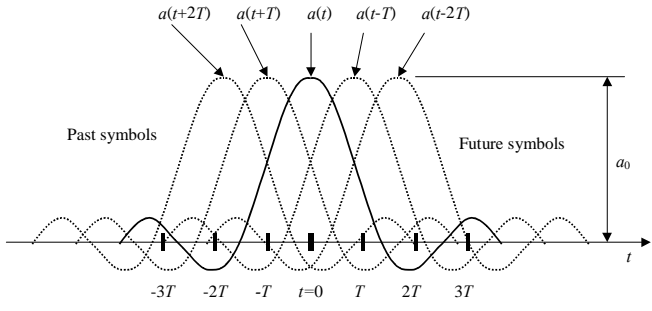


Figure 6: ISI due to pulse overlapping

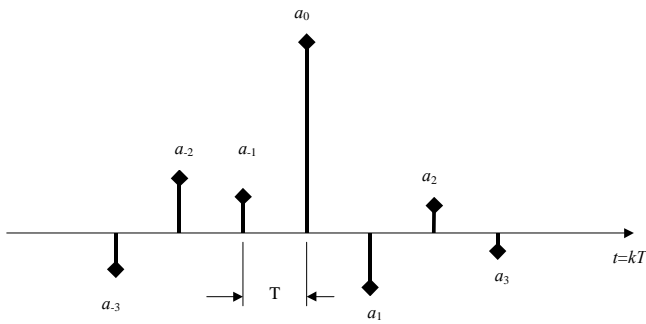


Figure 7: Sampled values at $t=kT$

At the sampling moment, the S&H circuit will sample not only the desired symbol a_0 , but also the remnants of previous and forthcoming pulses. We can write:

$$(17) \quad a_r = \sum_{-N}^N b_i a_i$$

where b_i are previous/current/next bits, and N is the number of pulses each side of the current pulse, giving rise to ISI (we assume the same number of previous/future pulses).

When $+B$ has been transmitted, the detected levels will cluster around a_0 . We can define a vector of all amplitudes at the sampling time $t=kT$ ($-N \leq k \leq N$) as:

$$(18) \quad \mathbf{a} = \begin{bmatrix} a_{-N} \\ a_{-N+1} \\ \vdots \\ a_{-1} \\ a_0 \\ a_1 \\ \vdots \\ a_{N-1} \\ a_N \end{bmatrix}$$

Let us form an auxiliary $(2^{2N} \times 2N)$ matrix \mathbf{d} :

$$(19) \quad \mathbf{d} = \begin{bmatrix} -1, -1, -1, \dots, -1, -1 \\ -1, -1, -1, \dots, -1, +1 \\ \vdots \\ +1, +1, +1, \dots, +1, +1 \end{bmatrix}$$

This is a matrix of all possible combinations of -1 and $+1$. If we split \mathbf{d} vertically in half and add a center column of ones, we will get a new $(2^{2N} \times (2N+1))$ matrix \mathbf{c}

$$(20) \quad \mathbf{c} = \begin{bmatrix} -1, -1, \dots, +1, \dots, -1, -1 \\ -1, -1, \dots, +1, \dots, -1, +1 \\ \vdots \\ +1, +1, \dots, +1, \dots, +1, +1 \end{bmatrix}$$

Then we can write the vector of all possible received amplitudes (when $+B$ has been transmitted) as:

$$(21) \quad \mathbf{a}_r = \mathbf{c} \cdot \mathbf{a}$$

The dimension of \mathbf{a}_r is $(2^{2N} \times 1)$. As all bits are equally probable and there is no correlation between symbols, all possible amplitudes will also be equally probable. There are 2^{2N} possible detected amplitudes, so:

$$(22) \quad P(a_r) = \frac{1}{2^{2N}}$$

The graphical representation of the above (when the transmitted pulse is $+B$) can be found in Figure 8. Only the fully drawn bold lines are in the set of possible values. The dashed lines can not exist as single detected values, unless one or more ISI amplitudes is zero. Then the bold lines will pairwise combine to fall into the dashed positions leading to $2x, 4x, \dots$ the amplitude of $p(z)$.

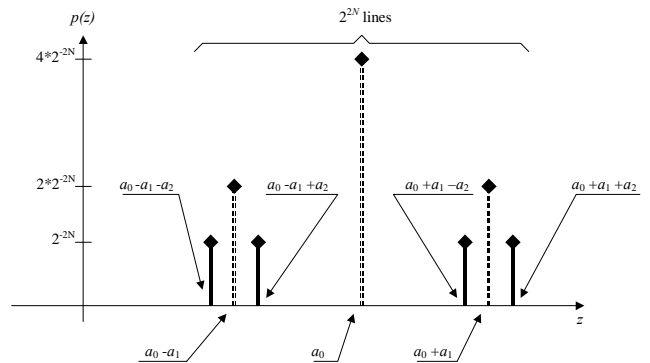


Figure 8: Possible detected amplitudes

The detection threshold γ_0 remains unchanged ($\gamma_0 = 0$). With AWGN superimposed, we have a sum of two independent random processes and the combined PDF is a convolution of both. A graph of the case in Figure 8 with added AWGN can be found in Figure 9.

The resulting PDF generally does not belong to any of the established classes, for which the probabilities have been thoroughly analyzed and documented in the literature. The

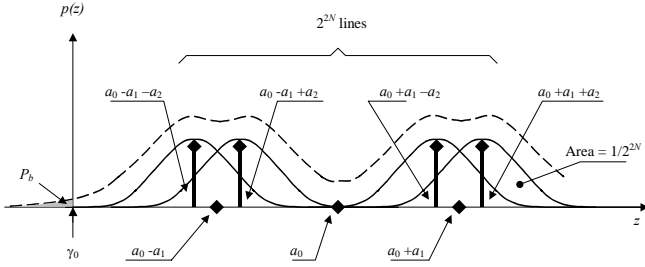


Figure 9: Detected amplitudes with noise

BER is the area under the sum PDF on the wrong side of γ_0 (shaded area in Figure 9). This is the sum of probabilities of each individual PDF crossing the decision threshold. The BER can then be written as

$$(23) \quad P_b = BER = 2^{-2N} \sum_{i=1}^{2^{2N}} Q\left(\frac{a_{ri}}{\sigma_0}\right)$$

This expression can be used as the Output Function (OF) or payoff value in the search for the optimal filter (search for the minimal BER).

IV. DERIVATIVE-FREE OPTIMIZATION

A. General

In the area of function optimization there is a class of algorithms called derivative-free algorithms. They are applicable to a range of optimizations, both discrete and continuous. They are particularly suitable for a range of problems where it is impossible or impractical to evaluate derivatives of the optimized function. Some common characteristics of these algorithms are:

- They do not need or use derivative information in order to optimize a given objective function (OF). Instead, they rely on repeated evaluation of the OF and use various methods to direct the search towards the optimum value.
- The guidelines used by the search are more or less intuitive in nature. Many follow the principles and ideas found in the natural world (evolution of species and natural selection).
- They can often be successfully used for complex problems, where derivatives of OFs can not be readily (or at all) computed, or would require unreasonable computing resources.
- These methods are generally slower than derivative-based methods.
- Most of these methods rely to a large extent on random generators to create potential solutions and to control the direction of the search. As a consequence, while they may relatively quickly find the area of optimal performance, there is no guarantee that the actual optimum solution will be found.
- The algorithms are of iterative nature and based on random generators. This generally means that we can never know with absolute certainty that the optimal solution has been found. A stopping criterion is needed to terminate the

search.

- They are difficult to study analytically. Most of the research of these algorithms is based on empirical studies.

B. Genetic Algorithms

Genetic algorithms (GA) are derivative-free optimization methods, based on the principles of the natural selection and evolution of the species. The idea has been taken from the natural evolutionary processes. These algorithms are often used for both continuous and discrete optimization problems and are very popular as a general-purpose optimization tool.

An alternative view of the GAs is in terms of optimizing a sequential decision process involving uncertainty in the form of lack of a priori knowledge, noisy feedback and a time-varying payoff function. More specifically, given a limited number of trials, how should one allocate them so as to maximize the cumulative payoff in the face of all this uncertainty [2]. Thus GAs, while they can quickly able to find regions of high performance even in the presence of the noise and time-dependent payoff, they are often unable to find the absolute optimum.

GAs encode each point in the space of solutions into a string, called chromosome. Each chromosome has an associated fitness value or Objective Function (OF). GAs normally maintain a multitude of solution points or chromosomes. This is called the population of chromosomes.

The algorithm attempts to evolve the population towards the optimum (minimum or maximum, whichever is the case). In analogy with the natural evolution theory, this is achieved through the recombination of genes and selection of the fittest. In order to prevent the algorithm being caught in a local optimum (a sub-optimal solution), mutation is often introduced. Other concepts like elitism serve to further improve the performance.

1) Encoding of Chromosomes

The parameter space is encoded into a single string, called the chromosome. If the parameter space has more dimensions (this is usually the case), the chromosome is composed of genes, sub-strings describing the coordinate values. For instance, a point in a three dimensional space (4,5,7) could be encoded into a 9-bit binary chromosome with 3 3-bit genes:

$$(24) \quad (4,5,7) = \underbrace{1001}_{4} \underbrace{0111}_{5} \underbrace{111}_{7}$$

Negative or floating point numbers can be coded in the same way. Coding need not be binary. Indeed, the binary representation may give rise to some problems. It is easily seen that in a case of mutation a random change of one digit can represent either a small or a large change of the coordinate in the parameter space, depending on the weight of the mutated position. Coding schemes, where the difference between neighboring positions are only in one bit (Gray code) are normally more suitable.

2) Fitness Evaluation

In order to select the most suitable candidates to pass their genes on to the future generations, we must define a

performance criterion for the selection. Each point in the parameter space has an associated OF or fitness value. A scaling or some other transformation may be required in order to achieve a monotonous measure of performance. Another option is to simply rank the population of the current generation. This has the advantage that the evaluation needs not be very accurate as long as the order is not affected.

3) *Selection of the Fittest*

After the OF of each member of the population is evaluated, the parents for the next generation are selected. The chromosomes with the higher OF have a higher probability of being selected for recombination. This is not to say that only the highest evaluating chromosomes will be selected. Even the poorest chromosome has a (small) chance.

A common strategy is to define the probability of being eligible for mating as the ratio of the member's OF and the sum of all OFs. Thus as long as the chromosome has some OF, it can become a parent.

It is often easy to select the best parents in the beginning of the search, when the differences in OF are substantial. Things can become more difficult towards the end of search, when all OF values are close and, based on probability of selection, there is no guarantee the search will progress in the right direction. In this case it is better to introduce relative ranking or some kind of relative OF evaluation in order to increase the difference between the best and the worst performers.

Another problem can be seen in the fact that the selection is only probabilistic, thus a very good solution may not be selected at all and its genetic information is lost forever. There is no guarantee that this solution will ever be found again. A common strategy to combat this situation is called elitism. It simply means that a few of the best chromosomes are kept and passed on to the next generation unaltered and will replace the worst performers. Other strategies have been developed, but are simply too numerous to list all here.

4) *Crossover*

Once the chromosomes are selected, they can be recombined to create a new generation of "children". In the process they will pass on the information contained in their genes and potentially create a better child than both its parents.

The technique is normally crossover. Two parent chromosomes are sliced into shorter segments and the contents of every even segment swapped. The number of segments can be from two to the length of the chromosome. When the number of segments is random, this is called a uniform crossover.

Various strategies can be employed, most frequently used being:

- One Point Crossover (Two parent chromosomes, A and B. Select a random crossover point and swap the contents of the chromosomes beyond that point)
- Two Point Crossover (Two parent chromosomes, A and B. Select two random crossover points and swap the contents of the second section)
- Uniform Crossover (Two parent chromosomes, A and B.

Swap the contents of the chromosomes randomly)

5) *Mutation*

The crossover will exploit the genetic potential of the population, and given enough time all possible combinations of the genetic information. But if the information is not there (and this is more than likely given that the first population is limited in size and was randomly generated), it simply can not create it.

In addition, the algorithm may begin to converge prematurely towards a sub-optimal solution. When the population begins to converge, the genetic diversity is lost. Most chromosomes are similar and no amount of swapping of similar genes will create new genetic information.

Mutation has been introduced to overcome these difficulties. It is generally implemented as a random inversion of a single chromosome position. The probability of this mutation has to be kept low, or it will become counterproductive and will degrade the performance of the GA. It is generally agreed that the probability of mutation has to be below 0.1. This is an empirical value.

In nature, the selection, crossover and mutation are preformed all in one step. GAs clearly distinguish between these steps. This makes the implementation and the control of the parameters manageable and aids experimentation with GA.

6) *The Role of a Population*

The GAs have the inherent capability to perform optimization in a parallel manner, searching the parameter space from multiple points in parallel. However, during the later stages of the optimization, the population inevitably begins to drift towards a more or less uniform solution. This can often be a sub-optimal case, and the real optimum may be a fair distance away from the currently explored space. Because the genetic diversity is being lost, there is no easy way to force the algorithm to continue to search for the optimum outside the limited set of genes it had drifted towards.

One way of dealing with this problem, the mutation operator, has been addressed above. Another solution is often found in introducing explicit parallelism. The original population is divided into a fixed number of sub-populations and each of them is allowed to evolve separately. A limited exchange of the genetic information allows for the whole search effort to be viewed as a single process, rather than a successive repetition of smaller isolated processes.

C. *Competitive Learning*

Unsupervised Learning (UL) is the name for a class of learning algorithms, where no external teacher or critic is available to instruct the process; only the input vectors can be used for this purpose. The UL detects or categorizes the persistent features in the input data without specific instructions from outside. It is used for data clustering and similarity detection. Competitive Learning (CL) is a popular scheme for UL clustering of data.

D. PBIL

1) Probability Vector

As explained earlier, a single population will inevitably lose its genetic diversity during the later stages of the search. This has given rise to the idea to replace the population with a single Probability Vector (PV). This vector can then be used to produce the members of the next generation.

Because the populations are described only by the probability of genes appearing, a particular PV can represent different populations. The PV can be regarded as a prototype for the current population. Thus the population is not unambiguously described by the PV; rather the PV represents, with high probability, the population of solutions with high OF.

The probability vector is not used to represent the entire population. This vector represents the point of the highest evaluation, around which the next generation of the points is to be created.

2) Competitive Learning in PBIL

There are many potential strategies of updating the PV during the progression of the search. Best results have been achieved by generating a population of points based on the current vector. These points are evaluated for OF and only the best one wins. The PV is then updated to point slightly towards the winning point. The rate of update is again called the Learning Rate (LR). The next generation of the points is then generated based on the updated PV. The process continues until the stopping criteria are met.

3) Mutations in PBIL

For the same reasons as with the GAs, mutations can be introduced to help maintain genetic diversity and prevent the algorithm from getting caught in a local minimum (or maximum).

The points of each generation in PBIL are created randomly, based on PV. As long as the PV stays away from the extremes (0 or 1), the chromosomes generated will randomly assume both values (in proportion with the probability expressed in the PV). In the later stages of the search the PV will converge towards either 1 or 0 and the chromosomes will be increasingly losing their genetic diversity. This is expected if the point the population is converging towards is the solution. In order to prevent premature convergence, some sort of mutation should be introduced. PBIL does not directly deal with the chromosomes, so a different approach has to be adopted. A mutation operator can act on the PV itself by randomly pulling elements of it away from 1 or 0 towards the center of the band (0.5).

In PBIL the importance of the mutations is somehow different than in GAs. Instead of acting on the chromosomes as such, the mutation operator acts on the PV and helps prevent the PV from converging too soon. Again, the mutation rate should be kept low enough to assure the overall convergence of the algorithm.

4) PBIL Algorithm

Based on the above, we can outline the algorithm in its

simplest form.

- The parameter space is encoded into a chromosome. Each bit position in the chromosome can be either 0 or 1. Gray coding is often used for the reasons explained earlier.
- A Probability Vector (PV) with the same number of positions is created. Each PV element can be any real number between 0 and 1. Initially, all PV elements are 0.5.
- A generation of points (chromosomes) is created based on PV. The value of the PV element describes the probability of a "1" appearing in this position in the chromosome. Initially ($PV(i)=0.5$) all chromosomes are completely random.
- The generation is evaluated with respect to the OF. The winner is declared.
- The PV is updated to take into account the genetic information of the winner (if the winner has a "1" in the position i , the $PV(i)$ is increased by the LR, if the winner has a "0", $PV(i)$ is decreased by the LR).
- Random mutations are applied to the PV according to the mutation rate.
- A new generation is created based on the new PV and evaluated for OF.
- The process is repeated until a stopping criterion is met. The winner is then declared the overall winner and returned as the solution.

5) Stopping Criterion

The PV starts as a string of elements set to 0.5. As the search progresses, they slowly drift towards either 0 or 1. The trend may change several times during the search, consequence of the variable gradient of the parameter space, noise and mutations. The PV will eventually converge to a string of zeros and ones, provided the search parameters are set correctly.

The best criterion for stopping the algorithm is to determine when this has happened (all or at least the majority of the PV elements are either 0 or 1). This will also mean that the genetic diversity is nonexistent and that there is no point in continuing to evaluate the same set of chromosomes.

Another method might be to limit the number of iterations, regardless of the PV value. This might prove useful if the process does not converge and can be used in conjunction with the above.

6) Possible Extensions of PBIL

The PBIL algorithm outlined above is really only the very basic version. It deals with a single PV and thus a single population. As such it suffers all the same shortcomings as the basic GA (loss of genetic diversity towards the end of the search, no guarantee of stability, an optimal solution may be lost due to the stochastic nature of the generations etc.)

Various enhancements have been proposed in the literature

- Multiple populations
- Time varying mutation rate
- Intelligent mutation
- Elitist selection

The problem to be optimized in this project is a fairly

simple one and does not justify the use of these enhancements with all the added complexity (although it might be interesting to experiment with them).

Only the intelligent mutation (instead of moving the PV randomly in a random direction, push it towards a less committed state. This means moving it away from 0 or 1 and towards 0.5) has been used for this project, due to its simple implementation. When compared with a more straightforward mutation strategy, there was no obvious gain.

V. OPTIMIZATION OF THE RX FILTER

A. The Model

Our QPSK channel was modeled for the data rate $R=10\text{MBps}$. In line with this data rate, the RRC signal was created to pass through zeros every 100ns. The energy of the RRC pulse is $8.5764e-8\text{Ws}$.

The pulse generator is followed by the smoothing filter (Figure 10). We have used a 7th order Butterworth filter with a corner frequency of 1.5 times the signaling rate $1/T$ (15MHz) and matched to 50Ω .

The RX filter was designed as a 7th order low pass structure with 4 capacitors and 3 inductors (Figure 11).

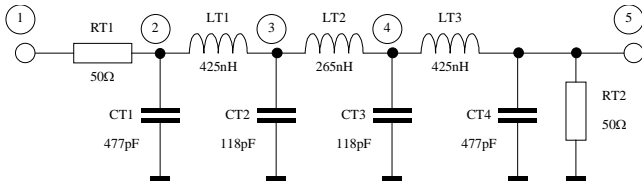


Figure 10: TX reconstruction filter

B. The Algorithm

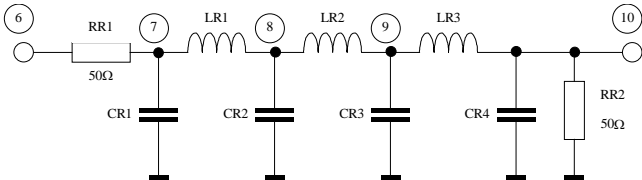


Figure 11: RX filter

1) General

The PBIL algorithm used was generally successful in finding an RX filter matched to the source. However, some care had to be taken in order to maximize the success of the search.

2) Noise Optimization

The filter has been optimized for the $E_b/N_0 = 10\text{dB}$. This value leads to $\text{BER} \approx 10^{-4}$ and consequently a reasonable objective function (OF). A lower E_b/N_0 would lead to a BER that is too high and hardly changes with the population (sensitivity problems). On the other hand, a higher E_b/N_0 would lead to a BER that is too low and is hard to evaluate (it would be lost in the round-off errors). This also corresponds to the area of interest in real communication channels (channel is unusable if $E_b/N_0 < 0\text{dB}$ and the $\text{BER} > 0.1$, while the RX filter is of lesser importance if $E_b/N_0 > 20\text{dB}$ and the

$\text{BER} < 10^{-10}$).

3) Order of the Filter

Several low pass and notch filter topologies have been tried. In the end, a low pass filter proved to be the best compromise between simplicity and performance.

Experiments with the filter order between 5 and 11 have been made. Filters of the 5th or lower order were inadequate (insufficient noise attenuation), while the 9th and higher, were too complex and did not contribute to the BER reduction in a significant way. A 7th order filter was chosen as the best compromise.

4) Scaling of the Components

This was the single most influential parameter responsible for the success or failure of the search. The PV values have to be related to the actual filter component values in order to allow a meaningful simulation of the channel and BER. The approach adopted here was to start with a reasonable range of components (0-10nF and 0-10μH). This gave a less than optimal solution with the components in the range $C < 1\text{nF}$ and $L < 2\mu\text{H}$. The scaling factors were then adjusted to put the expected component values roughly into the middle of the 16bit resolution and the search was repeated.

5) Stopping criterion

This is always a hard decision. When is the search over? Arguably, when there is no further improvement with the increasing number of iterations, the search should be terminated. We have chosen the value of the PV as the deciding factor.

The PV starts as a string of 0.5, then each position gradually evolves to either 0 or 1. It has been demonstrated that when the sum of all the distances of the vector elements from either 0 or 1 (whichever is closer) is 1 or less, the results do not change any more and the search is over. This was the stopping criterion in our search.

6) Mutations

Various mutation rates have been tried. A very low mutation rate or mutation shift had no effect and many runs were finishing in sub-optimal solutions. A high mutation would prevent the algorithm from converging. The mutation rate to affect 10% of runs and the mutation shift of 0.1 was chosen by experiment.

7) Analysis of Tolerances

The final outcome of the search, while leading to a proposed RX filter design, says nothing about the sensitivity of the design to the component tolerances. To this end, an additional run of simulations has been programmed. As capacitors can be found with 1% or better tolerances, the investigation was focussed on the inductors. A range of 10% inductors was assumed and the resulting BER calculated and plotted (Figure 15).

C. Results

The search has been repeated many times, with varying the parameters as described above. There were many successful outcomes and while they all result in roughly the same pulse response, frequency response and ultimate BER, they may

have different filter components. This implies that there are different RX filters with very similar responses. The difference between filters was most notable in the sensitivity of the BER versus component tolerances. One of the best filters in this respect was chosen here as the representative result.

The results presented here represent only one outcome of the search process.

1) Filter Circuit

The RX filter is a low-pass structure with components as in Figure 12. Input impedance and output impedance are matched to 50Ω . Capacitor values are as stated, inductor values are $\pm 10\%$.

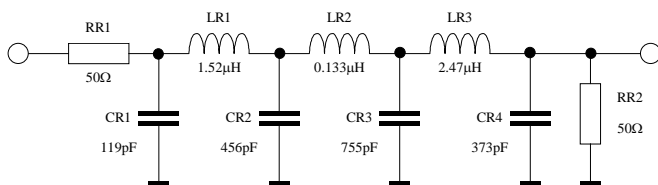


Figure 12: RX filter with component values

The tuning of the filter does not fall into any of the well known and documented categories. This is easy to understand.

The classic analogue filters are all distinguished by one or more of the performance criteria. Butterworth is the steepest filter with flat passband, Chebyshev is the steepest filter with a fixed amount of ripple in the passband (Chebyshev with zero ripple becomes Butterworth), elliptic filter allows ripple in the stopband as well as the passband, Bessel has no overshoot in time response, etc. The first three filters have a well-behaved frequency response and are suitable for filtering of signals with the information in the frequency domain, like audio signals. Bessel filter on the other hand, while it has a poor frequency response, is distinguished by no overshoot in time domain and is more suitable for signals with information in the time domain (video or other time domain signals).

We were not looking for a simple performance measure of this kind. Instead we wanted to find a filter leading to minimal BER, regardless of the frequency or time response. The optimal solution would actually be a RRC filter followed by an inverse Butterworth to compensate for the reconstruction filter on the transmit side. Only, such a filter can not be built with real passive components. First, such a filter would violate the causality principle (the RRC time response starts at $-\infty$ and goes on forever). In addition, the inverse Butterworth would require the frequency response to rise to infinity with frequency (stability problem). What we found is the closest thing that can be implemented as a 7th order low pass structure.

2) Time Response

The pulse shape after the RX filter can be found in Figure 13. There are several traces and they all correspond to $\pm 10\%$ tolerances of the inductor values.

The waveform is sampled at the moment of the positive peak for the maximum performance (this requires a synchronous detector circuit). A close inspection will confirm

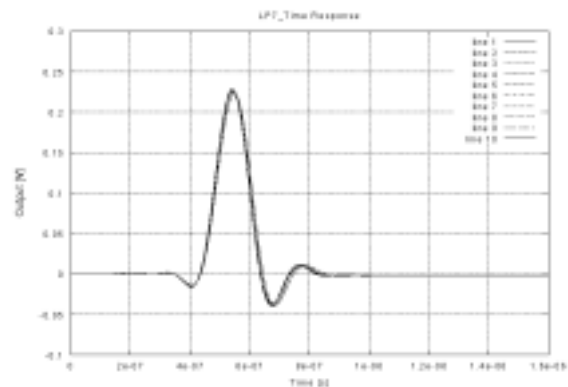


Figure 13: Pulse shape after the RX filter

that the waveform passes through zero (or very close to) every T seconds ($T=10^{-7}$ s). This causes minimal ISI and is in line with the initial requirements of this design.

3) Frequency Response

The frequency response of the RX filter itself is in Figure 14. There are several traces and they all correspond to $\pm 10\%$ tolerances of the inductor values.

The waveform closely follows the RRC shape up to $f=10\text{MHz}$. The spur between 20 and 30MHz at first seems unexpected. However, it consistently appears in all solutions, regardless of the order or the topology of the filter. It also appears in most sub-optimal solutions where the search got caught in a local minimum (these cases were observed during the development of the algorithm). The only explanation seems to be the compensation for the reconstruction filter used in the transmitter (15MHz butterworth described above and

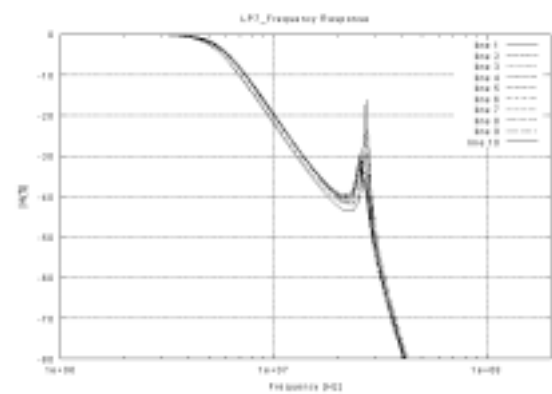


Figure 14: Frequency response of the RX filter

depicted in Figure 10).

4) The Noise Performance

The BER performance is depicted in Figure 15. There are three sets of lines:

- BER of the ideal matched filter (theoretical minimum, the lowest trace)
- BER of the RX filter with theoretical component values

(lowest of the group of traces)

- BER of the RX filter with $\pm 10\%$ tolerances for the inductor values (the highest group of lines)

This is the familiar "waterfall" graph. The E_b/N_0 range considered here is from -10dB to 20dB. Lower than -10dB values lead to a high BER (close to 0.5) and zero throughput, higher than 20dB values have negligible BER and are of no

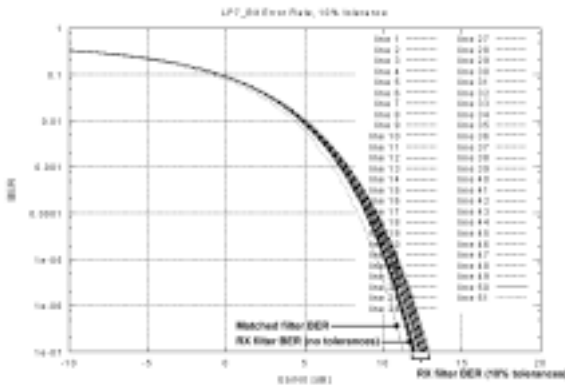


Figure 15: BER performance of the QPSK channel

interest here.

From Figure 15 we can read that the E_b/N_0 is generally 0.5dB worse than in case of a matched filter (and deteriorates by less than a further 1dB if 10% inductors are used). This is a good result considering the simplicity of the filter and loose tolerances of the components. Inductors with tighter tolerances (5%) are readily available and would slightly improve the BER. The performance can be further enhanced by handpicking the components (1% or less). However this is expensive and probably not necessary or beneficial. In reality, the channel is not exactly flat with infinite bandwidth. On top of this, the channel changes with time, leading to less than optimal performance even with ideal components. This again requires some kind of additional equalization (simple shift register multipliers), as mentioned in the Introduction.

VI. CONCLUSIONS

In most texts, the treatment of the digital communication channels with noise and ISI is generally divided into an ideally matched coherent detector case (without ISI) and a case with the ISI. The ISI analysis is generally limited to the worst case, when all interfering symbols add to act in opposition to the received symbol. This is then taken to narrow the opening of the eye-pattern and consequently reduce the noise immunity.

The focus and contribution of this paper is twofold:

- To treat the ISI as an interference of equally probable symbols from the past and the future. This requires a defined statistics of the data, in our case, equally probable symbols and no correlation between them, but does not limit the analysis to the worst case and allows for a more realistic estimate of the ISI impact on the BER

- To evaluate the impact of the combined noise and ISI on the final BER in the presence of an arbitrary RX filter. The noise level depends on the AWGN power in the channel and the frequency response of the RX filter, while the ISI depends on the time shape of the pulse after the RX filter, sampled at $t=kT$ intervals.

The BER of the channel could then be accurately evaluated under the above restrictions and its value used to stochastically search for an optimized RX filter. PBIL, a derivative-free algorithm was used here, but any other modern search algorithm would probably be just as successful.

This is an example of using theoretical analysis and reasoning together with a computer based search algorithm to find a practical solution to a problem in digital telecommunications.

REFERENCES

- [1] J.-S.R Jang, C.-T. Sun, E. Mizutani, Neuro-Fuzzy and Soft Computing, Prentice-Hall 1997
- [2] S. Baluja, Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning, School of Computer Science, Carnegie Mellon University, CMU-CS-94-163
- [3] R. Braun, Digital Communication Filter Design by Stochastic Optimization, Private communication
- [4] B. Sklar, Digital Communications, Prentice-Hall 1988
- [5] D. G. Childers, Probability and Random Process, Irwin 1997
- [6] L. W. Couch, Digital and Analog Communication Systems, Maxwell Macmillan 1990
- [7] A. J. Viterbi, J. K. Omura, Principles of Digital Communication and Coding, McGraw-Hill 1979
- [8] J. G. Proakis, Digital Communications, McGraw-Hill 1989