A New Stochastic Algorithm Inspired on Genetic Algorithms to Estimate Signals with Finite Rate of Innovation from Noisy Samples

Aitor Erdozain, Pedro M. Crespo

CEIT and Tecnun (University of Navarra)
Manuel de Lardizábal 15, 20018 San Sebastián, Spain

Abstract

In early 2000, it was shown that it is possible to develop exact sampling schemes for a large class of parametric non-bandlimited noiseless signals, namely, certain signals of finite rate of innovation. In particular, signals \( x(t) \) that are linear combinations of a finite number of Diracs per unit of time can be acquired by linear filtering followed by uniform sampling. However, when noise is present, many of the early proposed schemes can become ill-conditioned. Recently, a novel stochastic algorithm based on Gibbs sampling was proposed by Tan & Goyal (IEEE Trans. on Sign. Proc., vol. 56, N. 10, pp. 5135) to recover the filtered signal \( z(t) \) of \( x(t) \) by observing noisy samples of \( z(t) \). In the present paper, by blending together concepts of evolutionary algorithms with those of Gibbs sampling, a novel stochastic algorithm which substantially improves the results in the cited reference is proposed.

Key words: Signals with finite rate of innovation; Genetic Algorithms; Gibbs sampling

1. Introduction

In the last decades, the development of the technology, in part, has been based on the digital world. The compact discs have occupied the place left by the vinyl records. However, the nature follows being the same, and the signals that can be found in it are inevitably continuous. To work with digital signals, the first step is usually the discretization of the continuous signals, both...
in time and amplitude [1, 2]. In the present paper we will focus on the time
discretization.

It is widely known that a continuous signal with a limited bandwidth can
be sampled at a constant rate with no loss of information. The theorem of
Whittaker et al. states that a signal \( x(t) \) with bandwidth \( B \) can be reconstructed
from its samples \( x(nT) \), with \( T \leq 1/2B \), as

\[
x(t) = \sum_{n=-\infty}^{\infty} x(nT) \text{sinc} \left( \frac{t}{T} - n \right).
\]

Therefore, the sampling and the posterior reconstruction of the original sig-
nal can be performed only when the bandwidth is limited. However, in this
paper the interest will be focused on signals with infinite bandwidth but with
a finite number of degrees of freedom per time period, known as signals with
finite rate of innovation (FRI signals), which can be found in Ultra-Wideband
and Global Positioning System applications for example [3]. Examples of FRI
signals are the streams of Diracs, the nonuniform splines and the piecewise polyno-
mials. Based on the Whittaker theorem [4, 5, 6], these kind of signals cannot
be sampled and reconstructed directly. However, in [7] it is proved that some
signals with finite rate of innovation can be reconstructed from the samples of
their filtered versions, i.e. these signals must be first convolved with a kernel
function, \( h(t) \), before being sampled. The idea is to convert the original signal
with finite rate of innovation \( x(t) \) into a bandlimited signal, sample it and then
from these samples recover the parameters \( \theta \) that define the original \( x(t) \). That
is,

\[
\begin{align*}
  z(t) &= x(t) * h(t) \\
  z_n &= z(nT) \\
  \theta &= T(\{z_n\})
\end{align*}
\]

where \( T \) and \( T \) denote the sampling period and the corresponding reconstruction
transformation needed to recover \( x(t) \) from the set of samples \( \{z_n\} \), respectively.

In [7], the maximum sampling period and the reconstruction transformations
are described for periodic and finite-length \( x(t) \) signals. The reconstruction
methods are based on finding the zeros of the annihilating filters of the signals
of interest. However, these methods have a drawback: when noise is added to
the signal or samples, the error in the reconstructed signal is magnified. To
overcome this problem, in [8] an alternative method is presented based on the
matrix pencil scheme [9, 10], named Subspace-based algorithm. In the particular
case of periodic signals formed by weighted sums of Diracs, the authors in [11]
introduced a new approach that also manages to improve the results achieved
by the method described in [7]. They base their reconstruction method on
annihilating filters and combine the total least-squares approach with the Cad-
zow’s iterative denoising algorithm [12] to estimate the time instants. All these
methods employ a \text{sinc} or a Gaussian kernel function \( h(t) \), although the recon-
struction can be also performed for a wider set of kernels as long as these \( h(t) \)
functions satisfy certain conditions defined in [13].
The algorithms proposed in [7, 8, 11] are deterministic. By contrast, the authors in [14] proposed a stochastic algorithm based on the Gibbs Sampling [15, 16, 17] suitable for the reconstruction of finite-time signals formed by weighted sums of Diracs, that is, for signals of the type

\[ x(t) = \sum_{k=1}^{K} c_k \delta(t - t_k) \]

with \( c_k \) and \( t_k \) being the parameters defining the finite rate of innovation signal \( x(t) \). In presence of noise, this algorithm renders a lower distortion error \( \mathcal{E}_T \) in the signal \( z(t) \) than the annihilating filters method in [7], where

\[
\mathcal{E}_T = \frac{\int | \sum_{k=1}^{K} \hat{c}_k h(t - \hat{t}_k) - z(t) |^2 dt}{\int |z(t)|^2 dt}
\]

In the above expression \( \hat{c}_k \) and \( \hat{t}_k \) are the estimated amplitudes and time instants of the pulses. The main advantage of this algorithm when compared to the previously mentioned methods is that, although in [14] the algorithm is designed for Gaussian kernels, it can be employed in principle for any kernel function not required to satisfy the conditions defined in [13]. In addition, the algorithm can be modified to also include the reconstruction of periodic signals.

Notice that the actual information generated by the source is contained in the time instants and amplitudes of the Dirac pulses of \( x(t) \). Consequently, if the goal is to retrieve this information, the \( c_k \) and \( t_k \) values should be estimated from the noisy samples of \( z(t) \) (the reason for filtering the original signal \( x(t) \) generated by the source is to allow a uniform sampling of the signal). Unfortunately, when one is interested in estimating the actual values of \( c_k \) and \( t_k \) rather than in the overall filtered signal \( z(t) \), the algorithm proposed in [14] may fail to converge to the actual values of the pulse time instants \( t_k \). The aim of this paper is to introduce an alternative algorithm by mixing concepts of Evolutionary Algorithms [18, 19, 20] with those of the Gibbs sampling method, which substantially improves the results obtained in [14]. Furthermore, for low SNR values and small number of samples, the proposed algorithm also improves the results yield by the subspace-based [8] or the total least-squares [11] algorithms.

The rest of the paper is organized as follows. In Section 2, the problem considered is stated and the method given in [14] for its solution is reviewed. It is shown by numerical examples that its reconstruction algorithm may fail to converge to the actual parameters. Section 3 introduces the proposed reconstruction scheme based on genetic algorithms that overcomes these convergence problems. Section 4 presents simulation results including comparisons with other algorithms in the literature. A computational complexity analysis of the proposed algorithm is also addressed in this section. Finally, the main conclusions of this work are presented in Section 5.
2. Statement of the problem and the Gibbs Sampling algorithm

Let the original signal \( x(t) \) be

\[
x(t) = \sum_{k=1}^{K} c_k \delta(t - t_k),
\]

that is, a weighted sum of \( K \) Dirac pulses. This signal is first filtered by a Gaussian low-pass filter (kernel)

\[
h(t) = \exp\left(-\frac{t^2}{2\sigma_h^2}\right)
\]

and then sampled at rate \( 1/T \). As it will be explained in Section IV, the selection of \( \sigma_h^2 \) will depend on the separations between the time instants \( t_k \), the smaller they are, the smaller \( \sigma_h^2 \) should be.

Regarding the sampling time \( T \), one would like to choose \( T \) as large as possible but small enough so that several non-negligible samples are produced when sampling the kernel. In this sense, a good compromise is achieved when selecting \( T = \sigma_h \).

The corresponding samples are given by

\[
z[n] = h(t) \ast x(t) \big|_{t=nT} = \sum_{k=1}^{K} c_k \cdot \exp\left(-\frac{(nT - t_k)^2}{2\sigma_h^2}\right)
\]

and the observation samples \( \{y[n]\}_{n=0}^{N-1} \) by

\[
y[n] = z[n] + e[n] \quad n = 0, 1, \ldots, N - 1
\]

where \( e[n] \) is a sequence of i.i.d Gaussian random variables with zero mean and variance \( \sigma_e^2 \). Therefore, the overall acquisition process from \( x(t) \) to \( \{y[n]\}_{n=0}^{N-1} \) can be represented by the model \( \mathcal{M} \)

\[
\mathcal{M} : \quad y[n] = \sum_{k=1}^{K} c_k \cdot \exp\left(-\frac{(nT - t_k)^2}{2\sigma_h^2}\right) + e[n] \quad n = 0, 1, \ldots, N - 1. \quad (3)
\]

In the sequel, we will use the following vector notation:

\[
\mathbf{Y} = [y[0], y[1], \ldots, y[N-1]]^T,
\]

\[
\mathbf{c} = [c_1, c_2, \ldots, c_K]^T,
\]

\[
\mathbf{t} = [t_1, t_2, \ldots, t_K]^T,
\]

and \( \mathbf{\theta} = \{\mathbf{c}, \mathbf{t}\} \) to denote the complete set of decision variables. In here, \( \mathbf{c} \) and \( \mathbf{t} \) are considered to be random vectors with a joint probability distribution \( p(\mathbf{c}, \mathbf{t}) \).

The problem we want to look at can be summarized as follows: Given a realization \( \mathbf{y} \) of \( \mathbf{Y} \) and the model \( \mathcal{M} \), find the Minimum Mean-Square Error (MMSE) estimate of \( \mathbf{\theta} = \{\mathbf{c}, \mathbf{t}\} \), i.e.,
\[ \hat{\theta}_{\text{MMSE}} = E(\theta | Y = y, \mathcal{M}) = \int \theta \cdot p(\theta | y, \mathcal{M}) d\theta \]

In [14], an approximation of \( \hat{\theta}_{\text{MMSE}} \) was found by generating a sequence of samples \( \{\theta_i\}_{i=1}^I \), drawn from the joint conditional probability density \( p(\theta | y, \mathcal{M}) \), and computing its average, i.e.,

\[ \hat{\theta}_{\text{MMSE}} \approx \frac{1}{I} \sum_{i=1}^I \theta_i. \] (4)

For sampling the \( p(\theta | y, \mathcal{M}) \) the Gibbs algorithm was used, under the assumption that the priors of \( \theta = \{c, t\} \) were uniformly distributed, that is, \( p(c) \) was a constant in some volume of \( \mathbb{R}^{2K} \).

2.1. Gibbs Sampling (GS)

Gibbs sampling is based on the Hammersley-Clifford theorem [21], which states that given the data \( y \), the conditional densities \( p_i(\theta_i | \theta_{\{j \neq i\}}, y, \mathcal{M}) \) contain sufficient information to produce samples from the joint density \( p(\theta | y, \mathcal{M}) \). Therefore, in the Gibbs sampling one only considers univariate conditional distributions — the distributions when all the random variables but one are assigned fixed values. Such conditional distributions are far easier to simulate than complex joint distributions and usually have simpler forms. In this way, \( n \) samples are drawn sequentially and iteratively from \( n \) univariate conditional probabilities instead of generating a single \( n \)-dimensional vector from the full joint distribution in a single step. The algorithm starts with an initial value \( \theta^{(0)} = \{c^{(0)}, t^{(0)}\} \) and obtains a new value \( \theta^{(i)} \) by setting \( i = 1 \) and computing

\[
\begin{align*}
    c_1^{(i)} &\sim p(c_1 | c_2^{(i-1)}, c_3^{(i-1)}, \ldots, c_K^{(i-1)}, t^{(i-1)}, y) \\
    c_2^{(i)} &\sim p(c_2 | c_1^{(i)}, c_3^{(i-1)}, \ldots, c_K^{(i-1)}, t^{(i-1)}, y) \\
    \vdots &\sim \vdots \\
    c_K^{(i)} &\sim p(c_K | c_1^{(i)}, c_2^{(i)}, \ldots, c_{K-1}^{(i)}, t^{(i-1)}, y) \\
    t_1^{(i)} &\sim p(t_1 | c_1^{(i)}, c_2^{(i)}, \ldots, c_K^{(i)}, t^{(i-1)}, y) \\
    \vdots &\sim \vdots \\
    t_K^{(i)} &\sim p(t_K | c_1^{(i)}, t_1^{(i)}, t_2^{(i)}, \ldots, t_{K-1}^{(i)}, y).
\end{align*}
\] (5)

Repeating this process \( L \) times, we generate a Gibbs sequence of length \( L \), which is a realization of a Markov chain process [22]. Following a sufficient burn-in period of \( I_b \) iterations \( (I_b < L) \), the sequence approaches its stationary distribution, independently of the starting value \( \theta^{(0)} \). By construction, this stationary distribution is the target distribution we are trying to simulate. Therefore, the subset points \( \theta^{(j)}, I_b < j \leq L - I_b \), are taken as our simulated draws.
from the full joint distribution \( p(\theta | y, \mathcal{M}) \). The average value in expression (4) is now obtained as

\[
\hat{\theta} = \frac{1}{I} \sum_{i=I_b+1}^{I} \theta^{(i)}
\]

with \( I = L - I_b \).

Explicit expressions for the univariate conditional probability density functions in (5) can be easily obtained from \( p(y | \theta, \mathcal{M}) \) [14], since

\[
p(\theta_l | \theta_{-l}, y, \mathcal{M}) = p(\theta_l | \theta_{-l}, y) \propto p(y | \theta) = \prod_{n=0}^{N-1} \mathcal{N}(y[n], \sum_{k=1}^{K} c_k \cdot \exp \left( -\frac{(nT - t_k)^2}{2\sigma^2_h} \right), \sigma^2_e)
\]

where \( \theta_{-l} \) denotes the set of parameters \( \theta \) excluding \( \theta_l \), and \( \mathcal{N}(x, \mu, \sigma^2) \) denotes the normal distribution of the random variable \( x \) with mean \( \mu \) and variance \( \sigma^2 \).

Although it is guaranteed that the Gibbs sequence will reach stationarity, it can happen that for some of the parameters in \( \theta \), the burn-in period required to reach stationarity is orders of magnitude larger than the ones required for the rest of the parameters, making the procedure unpractical. Unfortunately, as shown in the next simulation example, this is the case for the time instants \( \{t_k\} \) in our problem. Some of the \( t_k \) may need very large \( I_b \) (computationally unfeasible) to render reliable estimates.

It should be mentioned that the distortion measure considered by the authors in [14] to evaluate their proposed Gibbs algorithm was the normalized quadratic error (1) between the filtered signal \( \hat{z}(t) \) and the corresponding reconstructed signal \( z(t) \). For this global distortion measure and for practical values of \( I_b \), their results were satisfactory. However, it fails when the goal is to estimate the actual values of \( \{t_k\} \) and \( \{c_k\} \) in the original signal \( x(t) \) (2).

### 2.2. Simulation Example

We randomly generated 1000 realizations of weighted sums of Dirac pulses as in (2). The signal \( x(t) \) in each realization last 14 seconds and the amplitude weights \( c = \{c_1, \ldots, c_K\} \) were modeled as independent and identically distributed (i.i.d) random variables with an uniform distribution \( U[3, 10] \). Regarding the \( \{t_k\}^K_{k=1} \), the gaps between pulses followed an exponential distribution with mean one (second) and with a minimum spacing of 0.5 seconds, i.e., \( t_{k+1} = t_k + 0.5 + v \), where \( t_0 = 0 \) and \( v \) is an exponentially distributed random variable with parameter 1. Notice that the number of pulses \( K \) in \( x(t) \) will depend on the particular realization considered, and its value is such that \( t_K \leq 14 < t_{K+1} \). The observed sequence is

\[
y[n] = \sum_{k=1}^{K} c_k \cdot \exp \left( -\frac{(nT - t_k)^2}{2\sigma^2_h} \right) + e[n] \quad n = 0, 1, \ldots, N - 1
\]
where \( N = 100 \) (\( \sigma_h = T = 0.1414 \) sec.), and the variance of the noise \( e[n] \) was set to 0.25, giving an average Signal-to-Noise Ratio (SNR) of 27.7dB.

The following distortion measure was used for the estimated time instants \( \{\hat{t}_k\}_{k=1}^K \):

\[
\varepsilon = \sqrt{\frac{1}{K} \sum_{k=1}^{K} (\hat{t}_k - t_k)^2}.
\]

The total number of iterations was set to \( I_b + I = 1000 \), with 980 of them belonging to the burn-in period (\( I_b = 980 \)).

Fig. 1 shows a simulation result for a particular realization of \( x(t) \). The amplitudes \( \{c_k\}_{k=1}^K \) and time instants \( \{t_k\}_{k=1}^K \) corresponding to this realization of \( x(t) \) are given in Table 1. This realization has been chosen since clearly highlights the drawback of the algorithm proposed in [14]. The dashed lines show the true time instants \( \{t_k\}_{k=1}^K \), while the solid curves show the Gibbs sequence of these parameters as a function of the number of iteration. It can be seen that the MMSE estimator of \( t_4 \) and \( t_5 \) will not converge to the true time instants. In particular, for the first 200 iterations shown in this plot, the estimates of \( t_4 \) and \( t_5 \) are close to \( t_1 \) most of the time.

<table>
<thead>
<tr>
<th>( k )</th>
<th>( t_k )</th>
<th>( c_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.19s</td>
<td>8.81</td>
</tr>
<tr>
<td>2</td>
<td>4.38s</td>
<td>8.27</td>
</tr>
<tr>
<td>3</td>
<td>5.61s</td>
<td>6.72</td>
</tr>
<tr>
<td>4</td>
<td>10.02s</td>
<td>5.03</td>
</tr>
<tr>
<td>5</td>
<td>11.15s</td>
<td>7.88</td>
</tr>
</tbody>
</table>

Table 1: Time instants and amplitudes of the \( x(t) \) used in the example of the Fig. 1.

Fig. 2.2(a) plots the distortion error \( \varepsilon \) in expression (6) obtained for each of the 1000 realizations of \( x(t) \). In 310 out of 1000 realizations the error \( \varepsilon > 0.2 \), which means that in these realizations at least one of the time estimates differs considerably from its true value. In Fig. 2.2(b) the distortion measure \( E_T \) (see expression (1)) defined in [14] for the same 1000 realizations is shown. Observe that the algorithm in [14] performs well when the goal is to reconstruct the filtered signal \( z(t) \), however, it fails when trying to recover the positions of the time instants.

In the next section, we propose a novel algorithm aimed to reduce this distortion error \( \varepsilon \).

3. Proposed Algorithm

We begin by a brief review of Genetic Algorithms (GA) [18, 19, 20], since some of their concepts are used in the algorithm presented in Section 3.2.
3.1. Genetic Algorithms

A genetic algorithm (GA) is a search technique suitable for finding exact or approximate solutions in optimization and search problems. Genetic algorithms are categorized as global search heuristics and are a particular class of evolutionary algorithms (also known as evolutionary computation). They rely on techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover or recombination.

The GAs operate as follows. They start by randomly generating a set of
possible solutions, named individuals (or chromosomes). This set is known as the first generation or population. Each of the individuals is evaluated by a fitness function that gives an idea of their optimality. This fitness function is such that the sought solution of the problem achieves the highest fitness value in the set of possible solutions. Some operators inspired on genetics are then applied to these individuals, producing new individuals that form the next generation. This procedure is repeated successively, substituting old generations by new ones. It has been widely tested that this procedure gives the optimal solution in a great variety of problems.

We now define two particular GA operators, later used in the proposed algorithm of Section 3.2:

- **Elitism**: The individual with the highest fitness value in the generation is directly copied into the next generation.

- **Mutation**: An individual is replaced by a new randomly generated individual.

Although GA could be directly applied to minimize the error $||\hat{z}(\theta) - y||_2$, it would take a lot of time to find the solution. Instead, we define a genetic algorithm that, together with GA operators, employs the Gibbs sampling to generate the new generations.

### 3.2. The Gibbs-GA Algorithm

As stated in the previous section, the use of the Gibbs algorithm to solve the sought reconstruction problem could run into convergency problems. This is likely to happen when in the iterative process the samples drawn from the conditional probability distributions in (5) get stuck near a local maximum of the joint probability distribution.

To avoid local optima and improve in this way the convergence speed of the algorithm, we introduce the Gibbs sampling procedure into the framework of a genetic algorithm, the rational being that GAs bring mechanisms to escape local optima in optimization problems. The function to be maximized by the GA is $p(t, c|y, M)$, where the maximization is with respect to $\theta = (c, t)$. To that end, populations with just one chromosome (individual), representing the set of amplitudes and time instants of the $x(t)$ signal, are defined, and the following fitness function is chosen

$$f = -\sum_{n=0}^{N-1} \left[ y[n] - \sum_{k=1}^{K} c_k \exp \left( -\frac{(nT-t_k)^2}{2\sigma_h^2} \right) \right]^2.$$  

Notice that this function is related to

$$\log [p(c, t|y, M)] =$$

$$\log C - \frac{1}{2\sigma_t^2} \sum_{n=0}^{N-1} \left[ y[n] - \sum_{k=1}^{K} c_k \exp \left( -\frac{(nT-t_k)^2}{2\sigma_h^2} \right) \right]^2 + \log [p(c, t)] \quad (7)$$
where $C$ is a constant.

We are now ready to describe the proposed algorithm. It consists of two phases and the required burn-in period $I_b$ is split up into two sets of iterations $I_e$ and $I_m$.

**Phase I**

1. Like in the Gibbs sampling, randomly choose $\theta^{(0)} = \{c^{(0)}, t^{(0)}\}$ (first population). Set $i = 1$
2. Sample $\theta^{(i)} = \{c^{(i)}, t^{(i)}\}$ from the conditional pdf’s in (5).
3. The individual for the next generation is chosen as follows:
   - if $f(\theta^{(i)}) > f(\theta^{(i-1)})$
     - Choose $\theta^{(i)}$
   - else
     - Set $\theta^{(i)} = \theta^{(i-1)}$ (Elitism)
4. if $i < I_e$,
   - Set $i = i + 1$
   - Go to step 2.
5. else
   - Stop and keep the last sample $\theta^{(I_e)}$.

**Phase II**

1. Start with $\theta^{(I_e)} = \{c_{I_e}^1, \ldots, c_{I_e}^K, t_{I_e}^1, \ldots, t_{I_e}^K\}$ obtained in phase I, set $i = I_e + 1$ and copy $\theta^{(i+1)} = \theta^{(I_e)}$.
2. Define the fitness value $\hat{f}$ as $\hat{f} = f(c^{(i)}, t^{(i)})$.
3. Initialize the index $j=1$.
4. Generate a random value $\hat{t}_j \sim U[0, t_{\text{max}}]$ (Mutation).
5. Introduce $\hat{t}_j$ in $t^{(i)}$ in substitution of $t_j^{(i)}$.
6. Simulate the amplitudes $\{\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_K\}$ from the following conditional distributions:

   $\tilde{c}_1 \sim p(c_1 | c_2, c_3, \ldots, c_K, t^{(i)}, y)$
   $\tilde{c}_2 \sim p(c_2 | c_1, c_3, \ldots, c_K, t^{(i)}, y)$
   $\vdots \sim \vdots$
   $\tilde{c}_K \sim p(c_K | \tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_{K-1}, t^{(i)}, y)$. 

Notice that the concept of elitism is used in step 3. Consequently, one guarantees that the sequence of samples $\{\theta^{(j)}\}_{j=1}^{I_e}$ (Gibbs sequence) will make the corresponding $p(c, t | y, M)$ sequence monotonically increasing.

In the next phase the concept of mutation is introduced. The idea is to introduce into the iterative process new randomly generated samples of the time instants (to escape local maxima), that otherwise, the Gibbs sampling would likely not produce due to their very low probability of occurrence (5).
7. Calculate the fitness value \( f(\tilde{c}_1, \ldots, \tilde{c}_K, t^{(i)}) \).

8. if \( f(\tilde{c}_1, \ldots, \tilde{c}_K, t^{(i)}) > \bar{f} \):
   \[ c^{(i)} = \{\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_K\} \]
   \[ \bar{f} = f(\tilde{c}_1, \ldots, \tilde{c}_K, t^{(i)}) \]
   else
   Restore the \( t_j^{(i)} \) to the value removed in step 5. (Elitism)

9. if \( j < K \),
   \[ \theta^{(i)} = (c^{(i)}, t^{(i)}) \]
   Set \( j = j + 1 \) and go to step 4.
   else
   \[ \theta^{(i+1)} = (c^{(i)}, t^{(i)}) \] and follow in step 10.

10. if \( i < l_e + l_m \),
    Set \( i = i + 1 \) and go to step 2.
    else,
    - if Version 1: The estimation is \( \hat{\theta} = \theta^{l_e+l_m} \).

11. Perform the standard Gibbs sampling process, i.e., without mutation and elitism, for \( I \) iterations taking \( \theta^{l_e+l_m} \) as the initial value, and then compute the MMSE estimation \( \hat{\theta}_{MMSE} \) from expression (4).

In Table 2, the GA parameters of the proposed algorithm are resumed.

<table>
<thead>
<tr>
<th>Encoding scheme</th>
<th>Real-number encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>1 individual</td>
</tr>
<tr>
<td>Fitness function</td>
<td>( f = -\sum_{n=0}^{N-1} y[n] - \sum_{k=1}^{K} c_k \exp \left( \frac{- (nT-t_k)^2}{2\sigma_k^2} \right) )</td>
</tr>
<tr>
<td>Reproduction type</td>
<td>Asexual</td>
</tr>
<tr>
<td>Probability of mutation</td>
<td>0 in phase I and 1 in phase II</td>
</tr>
<tr>
<td>Elitism</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 2: The employed GA parameters.

Notice that this algorithm has two versions, differentiated at step 10. In next section both versions will be explained. Fig. 3 shows the flowchart that summarizes the proposed algorithm. It represents the version 1 of the algorithm, but it is straightforward to modify it for version 2 adding a module for the standard Gibbs sampling at the end.

Observe that the time instants mutations in step 4 are drawn from a uniform distribution. This is true except for the first iteration \( i = I_e + 1 \), in which the following procedure has been adopted: based on the estimated values \( \{\theta^{(I_e+1)}\} \), the samples of the filtered signal \( \hat{z}(nT) \) are reconstructed and the difference
between \( \hat{z}(nT) \) and the received signal \( y(nT) \) computed. The new value for the mutated time instant is the value where this difference is maximum, that is,

\[
\tilde{t}_j = T \cdot \max_{n \in \{0, \ldots, N-1\}} \{|y(nT)) - \hat{z}(nT)|\}. \tag{8}
\]

Finally, observe that the pulse amplitudes estimates \( \hat{c} \) are also obtained from the proposed modified Gibbs sampling. The resulting estimates are sufficiently accurate to be used as priors in the conditioned marginal probability distributions. However, as proposed in [14], and due to the large variance of the probability distributions \( p(c_i|c_{-i}, t, y) \), a better estimation can be achieved by using the method of linear least squares to find \( \hat{c} \) from \( \hat{t} \). This procedure can also be applied in our algorithm, that is, after calculating the time instants estimates, the method of linear least squares could be employed to estimate the amplitudes of the pulses.

### 4. Simulation Results

As in the previous example, 1000 realizations of \( x(t) \) were randomly generated using the same parameters defined in Section 2.2. However, in this case \( I_b \) was split into \( I_m = 980 \).

The proposed algorithm was first tested in two slightly different versions, labeled as version 1 and 2. In version 1, the chromosome obtained in the last iteration of the genetic algorithm is taken as the estimate for the parameters \( \theta \), which theoretically maximize the joint probability density function \( p(c, t|y, M) \).

In contrast, in version 2 an additional step (step 11) is required, where a sequence of \( I \) samples \( \{\theta_i\}_{i=1}^I \) is obtained by performing \( I \) iterations of the standard Gibbs sampling (i.e., without mutation and elitism) followed by the computation of the MMSE estimate of \( \theta \) (see expression (4)). The initial value taken by the standard Gibbs algorithm is \( \theta_{I_m+I_b} \), obtained at the previous step 10. Notice that the use of the genetic algorithm to find \( \theta_{I_m+I_b} \) guarantees that this initial value will be close to the absolute maximum of the joint probability distribution. Thus, Gibbs sampling will most likely have reached its stationary state, rendering accurate estimates.

Based on these two versions, Fig. 4 shows the resulting error \( \varepsilon \) (6) for the different realizations of \( x(t) \). A large error peak in the graph means that at
least one of the estimated time instants is far from its true value. Observe that version 1 produces a lower number of error peaks than version 2. In particular, with version 1, only 15 of the 1000 realizations have \( \varepsilon > 0.2 \), whereas with version 2 this happens 32 times out of 1000. A possible explanation why version 2 produces more error peaks is that, even though the Gibbs sampling in step 11 has reached stationarity, atypical samples with low probability of occurrence could still be drawn (from the probability tails). Therefore, if few samples \( I \) are used, these atypical samples could distort the corresponding MMSE estimate.

On the other hand, it is of interest to assess the performance of both versions of the algorithm when the range of the error is small. This is done in Fig. 5 where plots of their error difference \( \varepsilon_1 - \varepsilon_2 \) are shown. Notice that in this case, as opposed to what happens for large errors (peaks in the graph), version 2 outperforms version 1. Therefore, to improve the overall performance of the algorithm, the following strategy has been adopted. If the difference between the estimates in both versions is not large (\( |\varepsilon_1 - \varepsilon_2| < 0.05 \)), we take as valid the estimates of the second version. Otherwise, we take as estimates the ones from the first version of the algorithm.

![Graph showing error measurements for different realizations using the proposed algorithm in its two versions.](image)

**Figure 4:** The error measurement for different realizations using the proposed algorithm in its two versions.

Based on this modified algorithm, Fig. 6(a) plots the resulting time average
error $\varepsilon$ versus the SNR (the same SNR value is considered in all the realizations), for observation block lengths $N = 50, 100$ and $200$ respectively. In this plot, realizations where their timing error was higher than 0.2 (i.e., $\varepsilon > 0.2$) were discarded, since we are interested in realizations where the algorithm has converged.

The average error for the corresponding amplitude estimates is given by

$$\varepsilon_c = \sqrt{\frac{1}{K} \sum_{k=1}^{K} |\hat{c}_k - c_k|^2}$$

and it is plotted in Fig. 6(b). As before, this average has been calculated over the realizations with $\varepsilon < 0.2$.

In order to see the convergency behavior of the algorithm, Fig. 7(a) plots the evolution of the time instants for a particular realization of $x(t)$ (when $I_e = 20$ and $I_m = 180$). The dark lines represent the evolution of $t^i$ versus the number of iteration, while the gray lines represent the actual time instants. Observe that at iteration 21, the last time instant which has not yet converged jumps close to its true value. This iteration corresponds to the first iteration where mutation is performed using expression (8). Fig. 7(b) shows the evolution of the corresponding fitness measure (the horizontal line is the fitness value of the true solution). Observe that due to elitism, the fitness sequence is monotonically increasing.

Finally, Table 3 shows the performance of the algorithm as a function of the number of iterations in Phase II of the algorithm without considering step
11, that is, by looking at the intermediate estimates $\{\theta^{I_m+I_e}\}_{i=1}^{I_e}$. Recall from Section 3.2 that the first $I_e$ iterations correspond to Phase I, and the next $I_m$ to the steps 1 to 10 of Phase II. The shown results were obtained by running the previous simulation with a $SNR = 15$dB, $N = 100$, $I_e = 20$, and $I_m = 980$. The first column corresponds to the number of iteration, the second column in the table shows the number of realizations yielding $\varepsilon > 0.2$ (out of 1000), whereas the third and four show the timing and amplitude average errors, respectively. As before, the realizations with $\varepsilon > 0.2$ were not included when computing these average errors. Also shown in Table 3 are the average errors corresponding to the final estimates produced by allowing the Gibbs-GA algorithm to end by itself. That is, either by $\theta^{I_e+I_m}$ (version 1) or by $\theta_{MMSE}$ computed at step 11 (version 2).

Notice from Table 3 that, as the number of iteration $i$ increases, the accuracy of the algorithm keeps improving.
4.1. Comparison with other algorithms

We next compare the proposed Gibbs-GA algorithm with the following reconstruction methods found in the literature:

1. The Gibbs sampling method (Gibbs) in [14] (stochastic and Iterative)
2. The subspace-based algorithm described in [8] (deterministic and non-iterative).

The comparison is done for the two operating regions $\varepsilon \leq 0.2$ and $\varepsilon > 0.2$. In the region $\varepsilon \leq 0.2$ the objective is to compare the timing errors $\varepsilon$, whereas in the region $\varepsilon > 0.2$ the objective is to compare the number of realization in error, that is, the number of realizations yielding $\varepsilon > 0.2$ (out of 1000). Notice that these realizations are not included when computing the average errors in the region $\varepsilon < 0.2$.

For the region $\varepsilon \leq 0.2$, Fig. 8 shows the average timing errors $\varepsilon$ as a function of the SNR for $N = 50$ and 100. Notice that in this region, the Gibbs sampling method in [14] has not been included in the figure since in the low error region both algorithms Gibbs and Gibbs-GA coincide (if the error is low, the stationary state has been reached in both cases). The figure shows that the proposed algorithm outperforms the algorithm based on the total least-squares method in all the considered cases. It can be also observed that the reconstruction error with the proposed algorithm is lower than the error achieved by the subspace-based algorithm for low SNR and small $N$. In conclusion, in very noisy scenarios where $N$ is limited to low values, the Gibbs-GA algorithm renders lower average estimation errors than the other two referred methods. The results for the errors in the amplitudes are not shown because, in all the algorithms, the method of linear least square is proposed to estimate the amplitudes, and so, the average $\varepsilon$ and the average $\varepsilon_c$ are related.

Similarly, for the operational region $\varepsilon > 0.2$, Fig. 9 shows the number of realization yielding $\varepsilon > 0.2$ as a function of the SNR and for $N = 50$ and 100. Only the Gibbs and Linear Least Square methods are shown in conjunction with the proposed algorithm.

<table>
<thead>
<tr>
<th>i</th>
<th>Number of errors ($\varepsilon &gt; 0.2$)</th>
<th>avg($\varepsilon$)</th>
<th>avg($\varepsilon_c$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 ($I_e$)</td>
<td>580</td>
<td>0.0210</td>
<td>0.388</td>
</tr>
<tr>
<td>21</td>
<td>283</td>
<td>0.0220</td>
<td>0.483</td>
</tr>
<tr>
<td>50</td>
<td>98</td>
<td>0.0220</td>
<td>0.385</td>
</tr>
<tr>
<td>100</td>
<td>57</td>
<td>0.0209</td>
<td>0.376</td>
</tr>
<tr>
<td>200</td>
<td>38</td>
<td>0.0187</td>
<td>0.361</td>
</tr>
<tr>
<td>500</td>
<td>30</td>
<td>0.0157</td>
<td>0.342</td>
</tr>
<tr>
<td>1000 ($I_e + I_m$)</td>
<td>16</td>
<td>0.0139</td>
<td>0.332</td>
</tr>
<tr>
<td>Final estimates</td>
<td>16</td>
<td>0.0124</td>
<td>0.279</td>
</tr>
</tbody>
</table>

Table 3: Error results for different iterations.
with our proposed method due to their iterative nature. The subspace-based algorithm described in [8] is non-iterative and consequently does not run into convergency error problems (i.e., $\varepsilon$ was always less than 0.2). From this figure it can be observed that the Gibbs-GA clearly outperforms the Gibbs sampling approach, corroborating the initial intuition that the use of GA concepts would reduce the convergence problems in the standard Gibbs sampling method. In addition, for low SNR values, the proposed algorithm also outperforms the total least-squares approach presented in [11].

It can be also observed that in the standard Gibbs sampling the number of errors grows with the SNR, the reason being that when $\sigma_e$ decreases, the shape of joint probability distribution at the local maxima gets narrower, making it more difficult for this algorithm to escape from these local maxima.

Figure 8: The average error for different methods (a) $N=50$ and (b) $N=100$.

Figure 9: The number of realizations with high error ($\varepsilon > 0.2$) for different algorithms (a) $N=50$ and (b) $N=100$. 
Up to now, we have considered in all the realizations of \( x(t) \) a minimum spacing between pulses of 0.5 seconds, that is, the time instants, \( t_k \), were generated as (see Section 2.2)

\[
t_k = t_{k-1} + t_{\text{min}} + v, \quad k = 1, \ldots K + 1, \text{ and } t_0 = 0
\]

where \( t_{\text{min}} = 0.5 \) seconds and \( v \) is an exponentially distributed random variable with parameter 1. The \( K \) value, for a given realization, is such that \( t_K \leq 14 < t_{K+1} \), where 14 seconds is the length of the \( x(t) \) realizations. In order to analyze what happens when \( t_{\text{min}} \) is lower, a set of simulations were carried out for different \( t_{\text{min}} \). For each \( t_{\text{min}} \), 1000 realizations were again obtained with SNR=15dB and \( N = 100 \). To do a fair comparison, the total duration of the realizations was adjusted so that they had the same average number of pulses as in the previous simulations (\( t_{\text{min}} = 0.5s \)).

The results are shown in Table 4, where the number of errors and the average errors are calculated as before. Notice that the algorithm performance substantially degrades as \( t_{\text{min}} \) decreases. The problem with low time spacings arises because the resolution of two Dirac pulses after being filtered by a kernel with a fix bandwidth, diminish as the separation between deltas decreases. However, the problem can be solved by increasing the bandwidth of the kernel function. The downside of this is that when the bandwidth increases, the \( \sigma_h \) in the kernel decreases, and since the sampling period \( T \) must be proportional to \( \sigma_h \) to keep the reconstruction accuracy, the number of required samples will have to increase also.

<table>
<thead>
<tr>
<th>( t_{\text{min}} )</th>
<th>No. errors</th>
<th>( \text{avg}(\varepsilon) )</th>
<th>( \text{avg}(\varepsilon_c) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02s</td>
<td>415</td>
<td>0.0162</td>
<td>1.0330</td>
</tr>
<tr>
<td>0.05s</td>
<td>373</td>
<td>0.0162</td>
<td>0.8783</td>
</tr>
<tr>
<td>0.1s</td>
<td>240</td>
<td>0.0162</td>
<td>0.5975</td>
</tr>
<tr>
<td>0.2s</td>
<td>81</td>
<td>0.0133</td>
<td>0.4688</td>
</tr>
<tr>
<td>0.5s</td>
<td>16</td>
<td>0.0124</td>
<td>0.2790</td>
</tr>
</tbody>
</table>

Table 4: Results for different values of \( t_{\text{min}} \).

4.2. Computational Complexity

We proceed to analyze the computational complexity of the proposed algorithm. The complexity of the Gibbs-GA algorithm is measured by the amount of operations required in its two phases. Here, by operation it is meant either a sum, a product or an exponential performed by the algorithm. In what follows, we keep the same notation as in previous sections, that is, \( K \) denotes the number of pulses of the signal (2), and \( N \) the number of samples.

In phase I, the most computationally intensive step is the calculation of the probability density functions \((5)\) derived in [14]. Each probability distribution requires \( \mathcal{O}(KN)^1 \) operations, and as there are \( 2K \) distributions, each iteration in

---

1By \( \mathcal{O}(n) \) operations it is meant that the number of operations grows linearly with \( n \).
phase I will require $\mathcal{O}(K^2N)$ operations. Regarding phase II, the computation of a fitness value involves $\mathcal{O}(KN)$ operations and this computation is repeated $K$ times in every iteration, so, it takes again $\mathcal{O}(K^2N)$ operations. Since phase I and II of the algorithm have $I_e$ and $I_m + I$ iterations, respectively, we conclude that the algorithm needs $\mathcal{O}((I_e + I_m + I)K^2N)$ operations or that the algorithm has an order of $(I_e + I_m + I)K^2N$ operations complexity.

5. Conclusions

This paper addresses the problem of reconstructing a signal with Finite Rate of Innovation from its noisy samples. In particular, this paper focuses on signals formed by streams of Dirac pulses. The authors in [14] were the first to propose an stochastic algorithm based on Gibbs sampling to cope with a similar problem. Although their algorithm works satisfactorily when the goal is to reconstruct the original noiseless samples, it may fail when trying to recover the parameters of the Dirac pulses, i.e., the time instants and amplitudes.

In this paper, a new stochastic algorithm which blends together concepts of genetic algorithms with those of Gibbs sampling is proposed. Simulation results show that this algorithm outperforms the approach in [14] when aiming to recover the parameters of the pulses. Furthermore, for low SNR and small number of samples $N$, the proposed Gibbs-GA algorithm improves the results obtained by the deterministic methods found in the literature, as the total least-squares method [11] and the subspace-based algorithm [8]. An additional advantage (see Section 1) of the proposed algorithm when compared to these deterministic reconstruction methods [7, 8, 11] is that it allows to work with kernel functions that do not have to satisfy the restricted conditions imposed to the deterministic algorithms [13], and it is useful for both periodic and non-periodic signals.

Finally, it should be mentioned that the Gibbs-GA, being stochastic, presents another inherent advantage when compared to deterministic methods: the algorithm allows, in a natural way, to use a priori information of the original signal to improve estimation. For instance, the probability distribution of the Dirac pulses could be considered in the fitness function. This advantage has not been exploit in this paper, but we believe it could be an interesting future line of research.

References


